6-17-2020

High Performance Computing Education: Current Challenges and Future Directions

Rajendra K. Raj
Rochester Institute of Technology

Carol J. Romanowski
Rochester Institute of Technology

John Impagliazzo
Hofstra University

Sherif G. Aly
American University in Cairo

Brett A. Becker
University College Dublin

See next page for additional authors

Authors
Rajendra K. Raj, Carol J. Romanowski, John Impagliazzo, Sherif G. Aly, Brett A. Becker, Juan Chen, Sheikh Ghafoor, Nasser Giacaman, Steven I. Gordon, Cruz Izu, Shahram Rahimi, Michael P. Robson, and Neena Thota

High Performance Computing Education: Current Challenges and Future Directions

Rajendra K. Raj[*]
Rochester Institute of Technology
Rochester, NY, USA
rkr@cs.rit.edu

Carol J. Romanowski[†]
Rochester Institute of Technology
Rochester, NY, USA
cjr@cs.rit.edu

John Impagliazzo[†]
Hofstra University
Hempstead, NY, USA
john.impagliazzo@hofstra.edu

Sherif G. Aly
The American University in Cairo
Cairo, Egypt
sgamal@aucegypt.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

Juan Chen
National University of Defense Technology
Changsha, Hunan, China
juanchen@nudt.edu.cn

Sheikh Ghafoor
Tennessee Tech
Cookeville, TN, USA
sghafoor@tntech.edu

Nasser Giacaman
The University of Auckland
Auckland, New Zealand
n.giacaman@auckland.ac.nz

Steven I. Gordon
The Ohio State University
Columbus, OH, USA
gordon.1@osu.edu

Cruz Izu
The University of Adelaide
Adelaide, SA, Australia
cruz.izu@adelaide.edu.au

Shahram Rahimi
Mississippi State University
Mississippi State, MS, USA
rahimi@cse.msstate.edu

Michael P. Robson
Villanova University
Villanova, PA, USA
michael.robson@villanova.edu

Neena Thota
University of Massachusetts Amherst
Amherst, MA, USA
nthota@cs.umass.edu

ABSTRACT

High Performance Computing (HPC) is the ability to process data and perform complex calculations at extremely high speeds. Current HPC platforms can achieve calculations on the order of quadrillions of calculations per second, with quintillions on the horizon. The past three decades witnessed a vast increase in the use of HPC across different scientific, engineering, and business communities on problems such as sequencing the genome, predicting climate changes, designing modern aerodynamics, or establishing customer preferences. Although HPC has been well incorporated into science curricula such as bioinformatics, the same cannot be said for most computing programs. Computing educators are only now beginning to recognize the need for HPC Education (HPCEd).

Building on earlier work, this working group explored how HPCEd can make inroads into computing education, focusing on the undergraduate level. This paper presents the background of HPC and HPCEd, identifies several of the needed core HPC competencies for students, identifies the support needed by educators for HPCEd, and explores the symbiosis between HPCEd and computing education in contemporary areas such as artificial intelligence and data science, as well as how HPCEd can be applied to benefit diverse non-computing domains such as atmospheric science, biological sciences and critical infrastructure protection. Finally, the report makes several recommendations to improve and facilitate HPC education in the future.

[*]Working Group Leader
[†]Working Group Co-Leader

ITiCSE-WGR '20, June 17–18, 2020, Trondheim, Norway
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8293-9/20/06…$15.00
https://doi.org/10.1145/3437800.3439203

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **Computing education**;

KEYWORDS

ITiCSE working group; high performance computing; HPC education; high-performance computing curricula; contemporary computing education; computer science education.

ACM Reference Format:
Rajendra K. Raj, Carol J. Romanowski, John Impagliazzo, Sherif G. Aly, Brett A. Becker, Juan Chen, Sheikh Ghafoor, Nasser Giacaman, Steven I. Gordon, Cruz Izu, Shahram Rahimi, Michael P. Robson, and Neena Thota. 2020. High Performance Computing Education: Current Challenges and Future Directions. In *2020 ITiCSE Working Group Reports (ITiCSE-WGR ’20), June 17–18, 2020, Trondheim, Norway*. ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/3437800.3439203

1 INTRODUCTION

High Performance Computing (HPC) provides the ability to process data and perform complex calculations at quadrillions of calculations per second, orders of magnitude faster than ordinary high-speed computers [92]. HPC can be performed on dedicated supercomputers typically containing thousands of compute-nodes working together to complete one or more tasks in parallel. Quite recently, these were "virtual supercomputers" comprising many inexpensive commodity computers configured in parallel or distributed settings. For instance, the Hadoop ecosystem [10] is an open-source archetype that can be operated at low-cost to provide on-demand processing of big datasets that cannot fit on a single machine. Although Hadoop-based systems were considered HPC, Apache Spark [11] is now becoming popular as it provides improved HPC, showing that what is considered HPC is time-dependent.

HPC promises to revolutionize not just computing, but also every field of human endeavor [103]. As Gray noted [62], scientific discovery has moved from the empirical and theoretical paradigms through computational modeling/simulation to the fourth paradigm of data-intensive exploration, for which HPC is the keystone. Only HPC strategies [76] make it possible to have modern climate prediction, healthcare, structural designs, aerodynamic designs, management of natural resources, arts and entertainment, and social interaction, just to name a few.

Despite these dramatic impacts of HPC, computing educators are only now beginning to recognize the need for HPC Education (HPCEd). Moreover, even for educators working in this field, HPCEd has presented a variety of challenges as detailed in this report. This working group examines the background in this space, identifies HPC competencies, discusses the support needed for HPCEd, investigates how HPCEd can impact computing education, and makes recommendations for the future.

To set the stage for the report, the working group considers several definitions for HPC. *Inside HPC* defines it as: "the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation to solve large problems in science, engineering, or business." [68]. The European Commission defines it as: "thousands of processors working in parallel to analyse billions of pieces of data in real time, performing calculations thousands of times faster than a normal computer" [47]. Another definition is "the use of parallel processing for running advanced, large-scale application programs efficiently, reliably and very quickly on supercomputer systems" [109], which usually function at teraflops, or even petaflops, where a *flop* is a *floating point operation per second.*

Although HPC is sometimes viewed to be the same as supercomputing, this working group views supercomputing as a subset of high performance computing. Specifically, the group treats HPC as the platform technology concerned with programming for performance, where performance takes the broad meaning of speed (reducing time to solution), energy efficiency (doing more with less power), upscaling (handling larger problems such as simulating a wing and then a full plane, or a cell and then an organ) or high throughput (the ability to handle large volumes of data in near real-time, as required in the financial services industry, telecoms or satellite imagery) [69].

HPC is important in contemporary computing in diverse ways. One estimate shows that the HPC market is projected to be worth $44 billion in 2023 [67]. One supercomputer installed in Illinois, USA in 2013 costing $208 million [37] had a $1.08 billion direct impact on the Illinois economy and created nearly 6,000 full-time jobs [66]. Apart from this and other immediate economic factors, HPC is vital to the sustainability, growth and competitiveness of many industrial sectors including pharmaceuticals, energy, manufacturing, and many others. The 2020 Top500 list which tracks the 500 fastest computers on Earth found that 58% of machines were located in industry followed by 20% in research, 11% in academia, 7% in government and the rest in other sectors. HPC is also essential to energy, national laboratories, and defense, and impacts the lives of billions of people daily in ways they often do not realize. Dozens of the world’s fastest computers are used in weather forecasting [118] and household names such as Disney use a cluster of computers to create animation [119]. Many of the fastest computers on earth have contributed towards efforts to fight the current COVID-19 pandemic [9, 86, 98].

More specifically, HPC is essential for today’s "hot" computing areas such as Artificial Intelligence (AI), Cyberanalytics (CA), Data Science and Engineering (DSE) [102], and the Internet of Things (IoT). The symbiosis between AI and HPC is demonstrated by the HPC community’s support for the anticipated AI revolution and the concomitant expectations that AI will drive improvements in HPC hardware and algorithms [48]. Baz [14] makes a strong case, with examples, for HPC in IoT while Perrin et al. [100] make a similar case for HPC in data analytics. In fact, one could argue legitimately that deep learning would not be feasible without the availability of different HPC frameworks [110]. The use of HPC and AI for Cyberanalytics has also become commonplace [82]. It is worth noting that these application areas, coupled with hardware advances such as GPUs and other accelerators, have driven HPC from homogeneous towards heterogeneous platforms. This is important as it has been shown that algorithms for homogeneous platforms are not always optimal on heterogeneous ones [15], opening up promising new advances.

Given HPC’s importance, HPCEd is needed from undergraduate through to post-graduate levels, from computing to non-computing disciplines. However, there are many barriers inhibiting the adoption and integration of HPC into computing education. For instance, almost all HPC platforms and/or applications require an understanding of parallelism, an inherently difficult concept to grasp. Currently, a fairly substantial portion of HPC education seems ad hoc, and requires a more structured approach. HPCEd also requires appropriate infrastructures to support student learning, and guidance to faculty on how to incorporate HPC into their curricula. Finally, new approaches are needed to advance the education of interdisciplinary specialists [76].

This paper addresses these and other questions related to HPC education, building on earlier work [75, 76]. The next section discusses the needed background in HPC and HPCEd. Section 3 presents several of the core HPC competencies students should attain, and Section 4 explores the support needed by educators for HPCEd. Section 5 explores the symbiosis between HPCEd and computing education in hot areas such as Artificial Intelligence and Data Science, while Section 6 examines how HPCEd can benefit non-computing domains such as atmospheric science, biological sciences and critical infrastructure protection. We make a few recommendations in Section 7 that we hope will move HPCEd forward.

2 BACKGROUND

Modern scientific development depends on large-scale data processing and supercomputers, enabling the future growth of computational science [62]. Supercomputers are widely used in scientific and engineering applications such as weather forecasting, climate research, oil and gas exploration, and physical simulations [76], all of which require HPC. This section provides the history of HPC, contrasts HPC with parallel and distributed computing (PDC), discusses efforts in HPCEd, and examines challenges and opportunities in HPCEd.

2.1 A High-Level History of HPC

Using the working definition of HPC as the aggregation of computing power to yield much higher performance than can be extracted from a typical desktop system to solve large problems [68], we take a quick look at the historical developments in this space.

Introduced in 1954, the IBM 704 was "designed for higher speeds and larger capacities required by problems of increasing complexity and size which confront business, industry, government and science" [34]. Along with the IBM 704 arrived the earliest generations of Fortran [42], whose current generation remains a popular language for HPC programming [108]. The IBM 704 could be said to have been designed precisely for HPC.

In 1963, Seymour Cray and Control Data Corporation (CDC) replaced vacuum tubes with transistors to create the CDC 6600, the world's first actual supercomputer [65]. Cray is often viewed as the "Father of Supercomputing". CDC 6600 could handle 9 megaflops of processing power and was cooled by Freon. From the 1970s through the 1990s, vector processors dominated the HPC field by operating and processing large amounts of data simultaneously. Two notable examples are the Cray-1 and Cray-2 vector computers.

The early 1960s also led to the increased use of parallel computing with the advent of the first true multiprocessor system: the D825 [8, 45], arriving well after the IBM 704. Parallel systems were intimately intertwined with the support of HPC, as we now understand the term. Since the 1990s, massive parallel processors (MPP) became the norm. Examples of these machines include the Cray-T3D, the ASCI Red, the Earth Simulator, and the Blue Gene/L. MPP machines comprised a large number of connected computer nodes using customized high-speed interconnection networks. Such tightly coupled MPP structures became problematic due to proprietary hardware, proprietary software, and vendors going out of business [116]. Soon clusters replaced these MPP structures, as they provided open source software, as well as commodity off-the-shelf

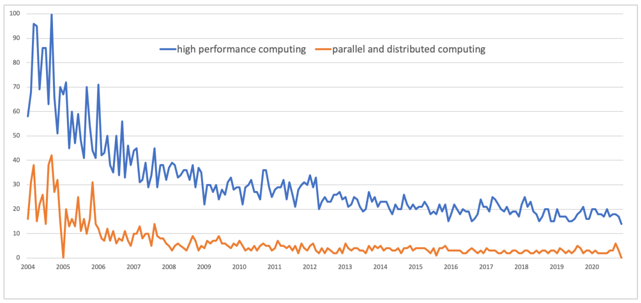

Figure 1: HPC versus PDC: Worldwide Google Trends

hardware. The Roadrunner became the first petaflops supercomputer, and the Tianhe-1 quickly followed as the first petaflops supercomputer with a CPU-GPU heterogeneous architecture [128]. Today's fastest supercomputer is Fugaku with a capability of 415.5 petaflops [120]. The next-generation exascale supercomputers deal with several system research challenges, including energy, efficiency, interconnect technology, memory technology, scalable system software, exascale algorithms, and resilience [80]. An exascale computer is one which has a performance of one or more exaflops. An exaflop is a thousand petaflops or a quintillion (10^{18}) double precision floating point operations per second. The first exascale system is predicted to debut around 2022 [121].

Distributed systems came into vogue in the 1970s, with the formalization of early networking protocols and client-server computing. The research into distributed computing theory and systems gained momentum in the 1980s, leading to distributed computing protocols, the Internet and the world-wide web in the 1990s, and cloud computing in the 21st century. Among other benefits, distributed systems offered yet another platform for realizing HPC. For example, the Google File System [52] and Hadoop [10] provided avenues to scale-up and scale-out the performance of a variety of applications.

The 21st century has been dominated by the fact that performance increases have hit a "power wall": increasing transistor and component density have resulted in heat dissipation, and therefore energy consumption, being the critical factor inhibiting performance advances in extreme-scale high performance computing [95]. This, coupled with other factors, have resulted in the proliferation of alternative architectures such as ARM, and heterogeneous architectures where general-purpose cores are augmented by specialized accelerators that offer superior performance per watt [43, 103]. Since 2013 the Green500 list, tracking the most power efficient supercomputers on Earth, has complimented the Top500 list [58].

2.2 Contrasting HPC and PDC

The combination of Parallel and Distributed Computing (PDC) represents one platform for achieving HPC. In fact, PDC has been so successful in achieving high performance that the two terms are often conflated. Viewed at the trends level in terms of worldwide Google search trends, as shown in Figure 1, it is fairly clear that the public at large views HPC as the broader concept as compared with PDC. In this subsection, we attempt to contrast PDC and

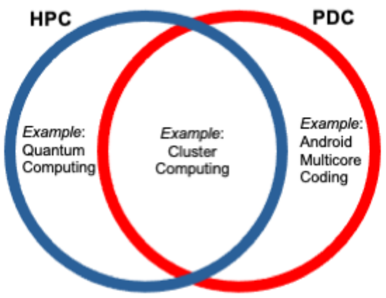
HPC
PDC
Example: Quantum Computing
Example: Cluster Computing
Example: Android Multicore Coding

Figure 2: Claim: HPC overlaps with but is not PDC

HPC to provide a foundation for the discussion of needed HPC competencies in Section 3.

We briefly define three core PDC concepts to facilitate our discussion; additional details about PDC can be found in several sources including the CDER PDC Curriculum [25].

(1) *Concurrency.* This term represents *conceptual separation* of a program's instructions into subsets of instructions (usually referred to as tasks) that may theoretically be executed simultaneously. While concurrency is a prerequisite for parallelism, concurrency does not require a physically parallel computing system. In other words, a program may execute its concurrent tasks on a single-core processor. Note that concurrency is not always used to improve performance. For instance, operating systems employ concurrency to give the appearance of simultaneous execution. Here we only consider concurrency when used to improve performance. For concurrency to deliver a performance improvement, an appropriate hardware environment, e.g. multicore or multiprocessor system, providing parallelism must be available. Concurrency is a property of the software.

(2) *Parallelism.* While concurrency allows a program's instructions to be decomposed into conceptually or logically separated tasks, parallelism is the actual execution of those tasks *literally at the same time*. Thus, parallelism implies a physically parallel computing system with multiple processing elements. Parallelism allows us to speed up the completion of programs composed by many concurrent tasks, or to execute multiple programs at the same time. Parallelism is a property of the hardware.

(3) *Distribution.* While parallelism refers to the temporal execution of a program's set of tasks, distributed computing refers to the spatial execution of those tasks—namely on physically distinct computing nodes separated by a network. A defining characteristic of distributed computing is the lack of a global clock, where the tasks execute "in their own world". Note that distributed computing is concurrent, but might not necessarily be parallel.

Figure 2 illustrates the relationship between the fields of HPC and PDC, providing one example to differentiate between the three subsets shown in the Venn diagram. We think of Quantum Computing as being primarily HPC-only, as issues of PDC is hidden away from the programmer and the user. Similarly, few people would think having multiple cores in a mobile phone leads to HPC, though it indubitably leads to PDC. In the intersection, cluster computing is both PDC and provides HPC. That is, while the fields have significant overlap, they are nonetheless different from each other. Due to this overlap, delineating between the two is not easy unless dealing with rather extreme cases. For example, running a large-scale simulation on a TOP500 supercomputer is obviously HPC, despite having many PDC concepts at work. On the other hand, operating system time-slicing on a multicore phone is clearly not HPC.

The goal of HPC typically is to massively reduce the wall-clock execution times of applications, programs, simulations, and such. However, there are cases where time reduction might not be the primary goal. For instance, some problems have memory requirements so large that they can only be tackled with HPC resources. In these situations, the primary motivation would be to select a platform that is capable of fully running the application regardless of speed, and thus the reduction of execution time becomes a secondary goal.

In this light, HPC refers to systems that are orders of magnitude more powerful than standard systems, e.g., mainstream desktops and mobile devices. Such HPC platforms can execute problems in minutes or days that would take standard computers months, years, or even longer. When considering computing history, this definition of HPC is a somewhat "moving target" in the context of PDC.

Is it possible to imagine an HPC world without PDC? Our working definition of HPC does not prohibit this possibility. For example, if some future (non-PDC) technology were to surpass typical computers in computing potential, HPC would effectively be possible without PDC. Quantum Computing, which promises computing power that is exponentially faster than any system today [94, 115], may be one such technology.

The working group acknowledges that modern-day HPC incorporates the broader application of concurrent, parallel, and distributed computing, where these approaches are used to improve the performance of achieving the lowest possible wall-clock execution time with the available resources. For the foreseeable future, PDC is an immediate precursor that enables this modern-day HPC. An important characteristic of HPC is that at any given time, it focuses on a single problem or a related set of simulations.

PDC represents the manner in which concurrency, the conceptual decomposition of the program, may occur in the form of a parallel or distributed program. Often a program is both parallel and distributed. Thus, a developer is able to achieve PDC on a

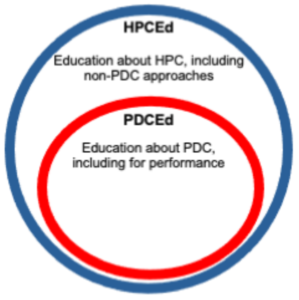
HPCEd
Education about HPC, including non-PDC approaches
PDCEd
Education about PDC, including for performance

Figure 3: Claim: Effective HPCEd covers all of PDCEd

standard mainstream system such as a multicore mobile phone or laptop, which is not necessarily HPC. Recall that HPC has a significantly more ambitious and purposeful objective: to achieve high performance improvement at a scale not possible on standard mainstream multicore systems. The difficulty in distinguishing between the two arises because modern-day HPC relies on PDC to achieve its goal, while at the same time PDC implies higher performance, which is the purpose of HPC.

2.3 Incorporating HPC into Computing Curricula

Over the past three decades, HPC has been increasingly used across the scientific, engineering, and business communities. However, the gap between HPC usage and HPCEd continues to increase because few computing students are educated in HPC competencies. Figure 3 illustrates the relationship of HPCEd in comparison to PDC Education (PDCEd), acknowledging that students need to at least learn PDC to learn HPC as there are elements of HPC that go beyond PDC. The HPCEd demands are additional to those of PDCEd, so it is imperative that HPCEd gear up to meet the contemporary needs of computing and engineering students.

Incorporating HPC early into the undergraduate curriculum has been under exploration since the 1990s, when Johnson et al. [72] proposed teaching parallel computing to first-year students by integrating parallel computation into a data structures course. Similar efforts include Lathrop [75, 76] and an ITiCSE 2010 working group report [22], which addressed questions such as what aspects of parallelism students should learn; how these concepts and practices could be incorporated into the computing curriculum; what resources might be needed; and what systemic obstacles exist to prevent adoption. HPC has been well-incorporated into many bioinformatics and computational biology curricula [99], but not uniformly or adequately integrated into computing programs as yet.

Some universities and many supercomputing centers worldwide promote broad involvement by offering some HPC curricula at undergraduate or graduate levels; for details, refer to Table 3 in Section 4. There has also been steady progress in the creation of educational materials and with the formation of a community of educators that can aid future efforts. In particular, the Curriculum Development and Education Resources (CDER) center [25] provides a detailed curriculum and parallel computing undergraduate courses [101].

The National University of Defense Technology (NUDT) in China has developed the Tianhe-2A system (ranking No.5 in the June 2020 TOP500 list), and also developed HPC undergraduate and graduate curricula [28–30], which promote broad involvement in parallel computing and supercomputing, even for non-computing and non-engineering majors [27]. One of the most important features of these HPC courses in NUDT is that students need to log in to the Tianhe-2A supercomputer system remotely to complete parallel programming assignments and debugging. Students are encouraged to use as many computing nodes as possible. The operation and practice on the actual supercomputer system is very helpful for students to understand some HPC problems, especially those HPC problems that are only exposed in relatively large-scale computing environments. The related academic research in supercomputing provides additional support for a robust curriculum leading to new possibilities for students worldwide.

Governmental and industrial alliances have also helped shape the HPC landscape. For instance, the European Technology Platform for High Performance Computing (ETP4HPC) is developing a strong ecosystem of technologies to drive the creation of leading edge HPC systems up to the exascale level [46]. The alliance has representation from over 15 different industrial partners, including IBM, Intel, and ARM. Not only does the initiative include research challenges, but it also covers training and education activities needed to create an appropriate professional skill set through the collaboration of the academic, scientific, and industrial communities. A notable challenge cited in the initiative however was that "The spectrum of deep knowledge and expertise required to develop competitive HPC infrastructures is extremely broad and goes far beyond traditional computer science" [46]. Needed competencies range from programming, processors, memory, and interconnection networks to cooling systems, heat reuse, and management systems. On another front, the Partnership for Advanced Computing in Europe (PRACE) [97] also includes a strong outreach program to help shape HPC education, and access to its supercomputing facilities.

Around 2005, a number of institutions called for the production of a workforce with expertise in computational science [96, 104, 112]. In response, there have been efforts at the undergraduate level to incorporate computational science into the curriculum. In the United States, the National Computational Science Institute [91] provided workshops for undergraduate faculty focused on introducing modeling and simulation in the undergraduate curriculum. That effort resulted in the introduction of computational science in many science disciplines [117]. Gordon, Carey, and Vakalis [56] formulated a set of undergraduate minor programs in computational science and produced a set of model competencies for such programs [64]. Since then, a significant number of undergraduate majors, minors, and concentrations in computational science have been created. Gordon and Cahill provide a recent survey of those programs and their challenges [57]. However, the slow dissemination of this program information to other institutions has not resulted in HPC integration into most undergraduate curricula.

2.4 HPCEd Challenges and Opportunities

Despite the initiatives listed in the previous section, the HPC expertise gap still exists [114] and HPCEd still faces multiple challenges. Few educators have the knowledge and skills to teach HPC, leading to limited educational opportunities for students [90]. In addition, HPC is applied in a large number of disciplines, making it more difficult for computing students to understand the different domains [90].

For a computing department that wants to adopt HPC coursework, just getting started poses serious difficulties [89]. More specifically, launching an HPC application is not a simple matter of creating a "hello world" program, as these applications contain multiple components that require different handling [130]. Therefore, HPC is obviously harder for domain experts who are not computing knowledgeable, requiring a serious learning curve [93].

Reflecting on the growing importance of parallel computing in undergraduate curricula, the CS2013 (the ACM/IEEE joint computer science) curricula report stated: "Previous curricular volumes had parallelism topics distributed across disparate Knowledge Areas (KAs) as electives. Given the increased importance of PDC, it seemed crucial to identify essential concepts in this area and to promote those topics to the core" [3, p. 29]. As a result, CS2013 introduced a new knowledge area in parallel and distributed computing [20]. Similarly, the CE2016 (the ACM/IEEE joint computer engineering) curricula report included several knowledge units in its recommendations such as "multi/many-core architectures, distributed system architectures, system architectural design and evaluation, and concurrent hardware and software design" [4].

The working group identified three major challenges in HPCEd that limit widespread adoption of HPC concepts in computing and engineering curricula. These can be characterized as *W-H-W*, stands for *What* to teach (Curricula), *How* to teach (Practical Environments), and *Who* teaches (Faculty) and learns (Students). We discuss each of these in the following subsections.

2.4.1 What To Teach? HPC plays an important role in a wide range of application fields, some within its own CS discipline, such as artificial intelligence and the Internet of things, and others across disciplines such as data science and bioinformatics. The diversity of domains makes teaching HPC difficult [90]. Experts in one domain lack necessary expertise in other HPC domains [76], which will greatly limit students' competency and versatility in the broader HPC field. Breaking the barriers between these various knowledge fields is crucial to produce multi-skilled talented graduates who can span multiple application fields. But developing and carrying out a uniform HPC curricular system is not feasible; we would not expect to build a complete HPC curricula across all possible fields. Competency-based learning can bridge the gap among different fields and is an effective way to mitigate this problem.

Increasing interdisciplinary knowledge and application ability is a very important educational goal but is difficult to achieve in the typical crowded computing curricula. In fact, most computing majors do not have enough cross-domain knowledge, which limits their development in the HPC field [61].

Interdisciplinary area for non-computing majors. Computational science creates advanced models that require a mixture of knowledge of mathematics, computing, and domain science and engineering expertise.

Many of the curriculum changes have occurred at the graduate level where the students have the appropriate disciplinary background to understand the systems being modeled and the challenge of advancing research in their own disciplines. It is not uncommon for graduate students to learn important skills and techniques through less formal means, such as workshops and training sessions provided by HPC research centers. Several graduate interdisciplinary programs in computational science have emerged in the physical sciences, materials science, atmospheric science, biological sciences, and earth science. Although somewhat unique, each program generally provides advanced work in the research challenges within the discipline, along with courses on the application of HPC to those challenges.

Scaling the number of students and programs that offer HPC expertise to undergraduate STEM students may require integration of the materials throughout the curriculum. The impetus for such an integration will require the expansion of the number of faculty using HPC in their research, extra training for additional faculty, and institutional recognition of the importance of those competencies to their graduates.

2.4.2 How To Teach? Ideally, universities would provide students with an immersive practical experience with real problems in large-scale system designs, software development, or HPC application optimization. Currently, the practice environment cannot satisfy such educational demands.

Lack of educational tools. Unlike the broader software engineering field, development effort estimations are much less established for the HPC community [126]. A long-standing concern in the HPC field is the ever-growing gap between hardware complexities and tools available to support developers, effectively resulting in a productivity bottleneck [79, 113]. Many of the issues surrounding the tools are general problems for the wider HPC community, particularly those that pertain to developer productivity: tools do not scale well, tools differ across platforms, and effective tools can sometimes lag behind the hardware by years [124]. More relevant to HPC education are additional problems imposed by tools that are difficult to learn and usually expensive for universities [124].

The broader computing education research community has long recognized the importance of tools for helping students learn programming [55]. In the space of introductory programming efforts, tools have been the dominant focus of researchers looking to support teaching and learning [16, 81]. Not only does this paper highlight the large number of purpose-built educational tools, but also includes numerous reviews dedicated purely to tools supporting learning [35, 107]. Educators continue to develop tools supporting introductory programming [81], but the same cannot be said for tools appropriate for the more difficult topic of HPC.

Limited practice environments. Unlike multicore parallelization, HPC presents unique problems of large-scale parallel systems such as scalability, parallel efficiency, heterogeneous computing, parallel storage systems and energy efficiency. HPC education should reproduce these unique problems; however, the current practice environment does not provide an immersive experience. Besides, effective practical exercises from real application cases are also essential for an immersive practical environment. Most of the real application cases cannot be directly used for teaching content because they tend to be too complex for student understanding. Students must be able to decompose real-world complex problems into smaller detailed forms so that they can understand how HPC can be applied.

2.4.3 Who Teaches and Who Learns? HPC remains a challenge for both computing students and instructors.

Student recruitment for HPC courses is a major problem in this field [76]. Students need to be convinced that the HPC field is important enough to justify taking on extra credit hours in their already challenging programs. As a result, most computing graduates do not have sufficient knowledge of parallel computer architecture elements such as an HPC interconnection network, different execution models for various architectural types (distributed, shared memory, and SIMD), common parallel algorithms, and high-performance analysis.

With regard to faculty, teaching loads are a concern. Typically, the expertise to teach HPC courses has been limited to one or a few faculty who must also maintain their teaching assignments for their departmental major courses or, in the case of research faculty, their grant work and overall research agenda. Because of the rapid development of HPC technologies and the limited availability of up-to-date textbook materials, the average faculty member lacks the vision and HPC knowledge to teach at the undergraduate level, but more will be able to do so if course materials are provided and readily available for use in an average lab.

Although academic HPC researchers have rich HPC engineering experience, they often focus on their own research or focus on graduate students, as they know the importance of producing many-faceted HPC talented graduates. Currently, these researchers are an untapped resource for undergraduate STEM education in HPC. How can institutions persuade such researchers to be involved in HPC course development, that is, developing course content design, in-class teaching, laboratory instruction, and thus reduce the burdens of other faculty?

3 HPC COMPETENCIES

Most computing curricular reports have described a discipline through knowledge areas, knowledge units, and learning outcomes, but do not generally provide guidance pertaining to skills or human behavior, especially as reflected in workplace environments. Recently, the Association for Computing Machinery (ACM) published two recommendations surrounding competencies in computing curricula [13, 50]. The recommendations, which included definitions of competency related to information technology and information systems, evolved independently, but their approaches are rather similar. Extending such efforts by creating competencies that incorporate the wide range of HPC related knowledge and skills across a variety of disciplines may serve as a critical guide to curriculum development. In this section, we look at HPC curricula using the competencies perspective.

3.1 Competency Areas

Competency areas relating to HPC have been developed as part of a number of endeavors, mainly from the computational science community, the ACM/IEEE computing education curriculum documents, and as an effort of government and industrial alliances.

3.1.1 The computational science community. The National Science Foundation (NSF) has supported grants to scientists and engineers focusing on the integration of computational and modeling skills into the curriculum. In addition, HPC centers funded by national and international agencies, national laboratories, and universities have provided training and education programs. As modeling and simulation have become an integral part of research in science and engineering, there has been a growing need to educate students about the concepts and tools required to create, implement, and validate models.

A number of projects have focused on the creation of educational modules to be used in classrooms and workshops [91]. A model, multi-institutional, interdisciplinary group of faculty have developed a set of competencies for undergraduate students in science and engineering [56]. Subsequently, those competencies were updated as a part of the education program for the XSEDE project [127]. The competencies are divided into seven major categories:

(1) Modeling and simulation
(2) Programming and algorithms
(3) Differential equations and discrete dynamical systems
(4) Numerical methods
(5) Optimization
(6) Parallel computing
(7) Scientific visualization

This mix of mathematics, computing principles, and domain science skills is intended to guide the development of undergraduate curricula. A number of institutions have used the competencies while building their programs, giving different subsets of the competencies greater or lesser importance depending on the particular field of study. The expectation is that all students will require modeling and simulation competencies and basic programming and mathematics knowledge, while those that go on to more advanced courses might require optimization and parallel computing skills to use HPC in their modeling efforts. Existing minor programs in computational science generally follow these major areas and also include experience with using science and engineering code on HPC systems to address domain level modeling challenges. For example, these challenges might include the use of computational chemistry code to model molecules or bioinformatics code for genomic analysis.

The introductory parallel computing competencies proposed for STEM (non-computing) majors include four major areas. With a grasp of these basic concepts, students will better understand the operating environments and underlying principles of scientific computing code that runs on HPC systems. These competencies described below were developed from the CS2013 [3] and CE2016 [4] curricula:

ST1. Demonstrating knowledge of parallel programming concepts and their relationships to different computer architectures.
ST2. Using parallel computing concepts to create parallel programs using MPI and OpenMP communications for a range of different problems.
ST3. Demonstrating knowledge of scalability and speed-up metrics for parallel programs.
ST4. Applying performance tools and profiling programs and using parallel libraries in parallel programs.

3.1.2 The ACM/IEEE Curricula. Updated periodically, these documents recognize that computing professionals must have skills related to the design and use of large-scale parallel and distributed computing systems, multicore and many-core systems, and their related programming and algorithmic requirements. The HPC-related skills embedded in the computer science curriculum document (CS2013) [3] are similar in many respects to the competencies constructed by the HPC community. The knowledge and skills listed in the document reflect a greater depth of understanding and HPC skills for computing students than those that were discussed for STEM majors. However, a number of the skills are listed as elective or as "Core-Tier 2" requirements, meaning that every program will cover 75-80% of these requirements, but may omit the rest as appropriate to that program. The CS2013 curriculum document includes seven major topics related to parallel and distributed computing:

CSC1. Introduction to modeling and simulation
CSC2. Multiprocessing and computer architectures
CSC3. Parallelism fundamentals
CSC4. Parallel decomposition
CSC5. Parallel algorithms, analysis, and programming
CSC6. Parallel architecture
CSC7. Parallel performance

Multiple HPC-related skills appear in the computer engineering document CE2016 [4]. However, there are further skills not listed in CE2016 that are also relevant to HPC. The skill sets in the report can be grouped into three major categories: a) performing algorithmic analysis of parallel algorithms, b) understanding parallelism from an architectural perspective, and c) understanding parallelism from a systems perspective, namely that of distributed systems. The relevant skills are:

CED1. Analyze the parallelism inherent in a simple sequential algorithm.
CED2. Explain why communication and coordination are critical to ensure correctness.
CED3. Calculate the speedup attainable in theory and explain factors limiting attainable speedup.
CED4. Explain limitations to scalability.
CED5. Discuss parallel algorithm structure and give examples.
CED6. Illustrate ways to manage algorithmic execution in multiple threads.
CED7. Select appropriate methods for measuring the performance of multithreaded algorithms.
CED8. Explain the impact of granularity and levels of parallelism in distributed systems, including threads, thread-level parallelism and multithreading.
CED9. Describe how the client-server model works in a decentralized fashion.
CED10. Articulate current programming techniques, models, frameworks, and languages for distributed, parallel processing.
CED11. Describe how programs are partitioned for execution on multi-/many-core processors.

Table 1 shows the relationships between these three perspectives.

Table 1: Relationships between perspectives of HPC skills

STEM Major Skills	CS 2013 [3]	CE 2016 [4]
ST1 Knowledge of PP and Architecture	CSC2 CSC3,4 CSC6	CED2 CED5 CED8,9,10
ST2 Use concepts to create parallel programs	CSC4 CSC5	CED1 CED6 CED11
ST3 Knowledge of scalability and speed-up	CSC7 CSC8	CED3 CED4 CED7
ST4 Applying performance tools and profiling	CSC7	–

3.2 New Competency Model

The CC2020 project consists of a fifty-member task force representing twenty countries and six continents. The project seeks to summarize the current state of curricular guidelines for undergraduate academic programs in computing [13], to guide a transition from knowledge-based learning to competency-based learning, and to set pathways for the future of computing education.

The CC2020 project specifies competency composed of three dimensions (knowledge, skills, dispositions) observed within the performance of a task. This construct leads to the following definition.

Competency = [Knowledge + Skills + Dispositions] in Task

Taken from the CC2020 draft report, Figure 4 illustrates a competency structure of knowledge, skills, and dispositions that are observable in the accomplishment of a task where the task prescribes purpose within a work context [125].

Knowledge is the "know-what" dimension of competency as a factual understanding. An element of knowledge designates a core concept essential to a competency. Table A.3 identifies 36 knowledge elements for computing organized into six categories as they appear in the CC2020 report. The knowledge categories include humans and organizations, systems modeling, software systems architecture, software development, software fundamentals, and hardware.

Skills refer to the capability and strategy for applying knowledge to actively accomplish a task. Hence, skills express the application of knowledge and are the "know-how" dimension of competency. Table A.2 describes six skill levels as they appear in the CC2020 report. The skills include remembering, understanding, applying, analyzing, evaluating, and creating.

Dispositions frame the "know-why" dimension of competency and prescribe a requisite character or quality in task performance. Dispositions shape the discernment of skillful engagement of "know-what" and "know-how." They encompass socio-emotional skills, behaviors, and attitudes that characterize the inclination to carry out tasks. The set of dispositions is an essential characteristic of a well-structured competency, and it has an intricate involvement in statements related to workplace or academic activities. Table A.4 describes 11 dispositions as they appear in the CC2020 report. The dispositions related to meta-cognitive awareness include being proactive, self-directed, passionate, purpose-driven, professional, responsible, adaptable, collaborative, responsive, meticulous, and inventive. They also include how we work with others to achieve common goal or solution.

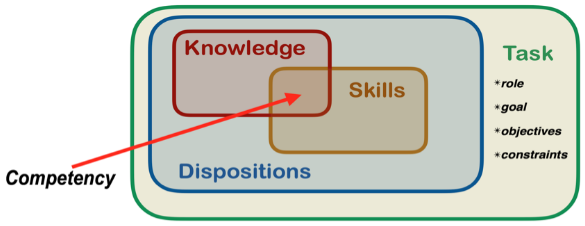

Figure 4: Competency Model (from CC2020 Report [13])

Task is the construct that frames the skilled application of knowledge and makes dispositions concrete. A task provides the setting to manifest dispositions, where individuals moderate their choices, actions, and effort necessary to pursue and succeed in an efficient and effective manner.

Two curricula reports have already used this new perspective to describe their scope:

- **The Information Technology IT2017 Report** diverged from the knowledge areas, knowledge units, and learning outcomes approach. This report [50] adopted competency-based learning, particularly since almost all graduates enter the industry or government workplace. It created the canonical triad that Competency = Knowledge + Skills + Dispositions taken in context. The CC2020 definition mentioned earlier uses a similar approach of knowledge, skills, and dispositions used in the IT2017 report. This canonical model of competency avoided the knowledge lens and uses competency as the centerpiece of learning for information technology.
- **Master of Science for Information Systems MSIS2016 Report**: Similar to the IT report, the MSIS2016 report [122] identified a set of graduate competencies. In this context, the term "competency" refers to a graduate's ability to use knowledge, skills, and attitudes to perform specified tasks successfully. The report represented competency as "a dynamic combination of cognitive and meta-cognitive skills, demonstration of knowledge and understanding, interpersonal, intellectual and practical skills, and ethical values" [78]. The MSIS2016 report specified competency areas as the highest-level categorization that included competency categories, and these categories specify the actual competencies.

3.3 Defining Competencies for HPC

As we have seen, the field of HPC is complex. From the perspective of HPC system composition, it involves computing subsystems, interconnection network subsystems, storage subsystems, and other components; from the perspective of multi-layer structure (bottom to top), it involves high performance processor design, architecture design, system software, resource management systems, parallel algorithm design, and cross-domain parallel application development. From the perspective of HPC application fields, it involves broad areas such as computational fluid dynamics and bio-medicine, as well as meteorological and ocean forecasting. This breadth and complexity increases the difficulty of including necessary elements in both computing and non-computing education.

Because of a lack of relevant HPC knowledge, non-computing majors usually struggle to understand concepts such as scalability, concurrency, parallel I/O, and reliability. Students tend to have a low level of understanding, focusing instead on optimizing HPC applications. Non-computing majors can better understand challenging HPC notions by connecting their domain concepts with the corresponding HPC concepts.

Competency-based HPCEd, in contrast to knowledge-based approaches, focuses on cultivating students' ability to comprehensively apply knowledge and skills in multiple fields as well as to communicate with other professionals. For example, a reduction in communication overhead can encompass the perspectives of interconnection network design, MPI interface design, and communication method selection. This ability to apply comprehensive, multi-faceted knowledge can be transferred to students through concrete examples in the teaching process.

At the same time, HPCEd needs to provide students with a powerful and real experimental environment, so that students can immerse themselves in the experimental setting. Optimizing HPC applications is a complex and iterative process, and due to its interdisciplinary link, requires students, instructors, and researchers to communicate in depth, accentuating the need for dispositions within human settings.

3.3.1 *HPC Competency Structure.*

Generating a list of HPC competencies follows the same structure as previously mentioned for regular computing areas. Skills, dispositions, and tasks remain the same; the only component that changes is knowledge. Therefore, we will refer to the most recent knowledge areas developed by the Computing Curriculum Development and Educational Resources (CDER) center at Georgia State University [25]. Table 2 provides a summary of these knowledge elements related to HPC. The elements reflect the experience of the authors and those attributed to the CDER project.

Based on the knowledge elements found in Table 2, the authors have created a list of basic HPC sample competencies for both computing and non-computing, and STEM curricula. We focused on a sample of basic parallel and distributed computing skills for both non-computing and computing majors. The key competencies are associated with understanding parallel processing concepts (such as processes, threads, and contention), some basic parallel computing algorithms, the measurement and affecting factors of scalability and "speed-up", and the use of tools that support parallel programming development and profiling. For computing majors, those same competencies are also required, but these students should also have a greater depth of knowledge and experience with a wider array of parallel algorithms, the key features of different parallel computing architectures, common concurrency problems, and parallel design patterns [84].

3.3.2 *Sample Basic Competencies.*

The following sample competencies apply to both non-computing and computing majors.

- Collaborate with colleagues and present a project to create a parallel program that addresses a critical question in a subject domain. In this example, the skills relate to parallel programming, the knowledge is related to the domain being modeled, and the disposition is collaboration with colleagues.
- Develop a framework for a commercial organization to compute the speed-up associated with converting a serial to a parallel program and its potential benefits toward improving the efficiency of an HPC system.
- Compare performance metrics between desktop applications and HPC-supported applications for a government agency.
- Exploit factors that mainly affect the scalability of a parallel program or client's HPC application, from both the architecture and the parallel algorithm perspectives.
- Develop a framework for a commercial enterprise to learn various commands or operations in a large-scale parallel

Table 2: HPC Knowledge Elements

Performance and Evaluation	Models and Applications	Programming	Middleware Tools	Architecture
Scalability	Model abstraction	Problem decomposition	Job scheduling	Shared memory
Speed-up	Model verification	Multi-threading	Load balancing	Distributed memory
Fault tolerance	Model validation	Parallel algorithms	Parallel I/O	Heterogeneous architectures
Performance monitoring	Domain application	Concurrency issues	Network throughput	Multicore / Many core
		Data management		Interconnect
		Vector processing		Pipelining
		Message passing		

computing system associated with using resource assignment strategies to achieve better job scheduling.

These examples focus on the HPC knowledge (as listed in Table 2) skills, and dispositions relating to parallel computing. The skills are reflected in the action words "compare", "develop", or "create". The phrases such as "commercial organization," "government agency", "client," and "commercial enterprise" are the implied dispositions that require human attributes to achieve a successful task.

3.3.3 Sample Competencies for Computing and STEM Majors. The following sample competencies are intended for computing and STEM majors.

- Design, implement, and assure the correctness of a high performance system using knowledge of thread implementation, synchronization, prioritization, and communication as well as a contemporary programming language for deployment in an environment that may occasionally require real-time performance. In this example, the skill includes the ability to design, implement and assure. The knowledge pertains to the implementation, of synchronization of threads, their prioritization, and communication, while the disposition applies to the context of the environment that requires the real-time performance.
- Analyze and compare different network topologies including but not limited to semantics, scalability, and other limitations for designing and building large-scale computing systems in organizations that undertake scientific research.
- Collaborate with colleagues and researchers on a project creating a parallel program that solves a problem with high efficiency by comprehensively using parallel architecture, HPC interconnection network, and parallel compilation technology, resource scheduling.
- Develop a framework for a commercial entity to calculate the parallel efficiency associated with converting a serial to a parallel program on homogeneous and heterogeneous architectures (such as CPU-GPU, CPU-accelerator) and the potential challenges of developing faster next-generation HPC systems.

Each program seeking to integrate HPC into their curriculum can use this approach to define the full range of basic, intermediate, and advanced HPC competencies that they find relevant to their curriculum. Such efforts should begin with the selection of the knowledge and skills that are important to their discipline and program. The knowledge and skills can then be combined with one or more dispositions to define the HPC competencies that should be integrated into the program. In turn, those competencies can be used to insert the appropriate modules into existing courses or to develop new courses that will achieve curricular goals.

4 SUPPORTING HPCED

As described in Section 2.4, a major challenge for faculty who are not conversant with HPC but are interested in including HPC in the average computing class is locating instructional resources ready for classroom use. In this section, we provide information and links to educational and infrastructure resources that cover HPC competencies.

4.1 Educational Resources

Although a range of resources have been developed for teaching HPC and PDC concepts (textbooks, reading materials, slides, syllabi, lab assignments, etc.), the community needs more instructional resources at different levels and covering different applications. More importantly, there is also a need for systemic categorization, cataloging, and evaluations of the existing resources with respect to quality, maturity, completeness, usability, and adoptability.

Brown [22] organized educational resources into three major categories: for classroom instruction, providing access to hardware platforms for hands-on HPC/PDC exercises, and software tools, libraries and games for aiding the instruction of HPC/PDC. Following Brown, we categorize the available resources into two categories: educational and infrastructure. We focus on resources that are available free of charge and are appropriate for undergraduate teaching, since course offerings are few and major gaps in faculty knowledge exist at this level.

Instructional resources help faculty to teach HPC/PDC concepts and students to learn those concepts. Lesson plans, handouts, lab modules, and games fall under this category. Table 3 provides a list of online resources along with a brief description and their weblink for easy access. Several universities worldwide also publicly share their course details and assignments on their websites.

4.2 Infrastructure Resources for HPCEd

Teaching parallel and high performance computing requires some hardware or software infrastructure; at minimum a multicore system and some parallel software is needed. This section describes different ways to meet hardware and software requirements.

Table 3: HPCEd Instructional Resources.

Name	Program/Institution Name	Resource Description and Web Link
AWS Educate	Amazon Web Services	Online learning modules on cloud computing and virtual cloud lab for students and faculty. https://aws.amazon.com/education/awseducate
BWPEP	Blue Waters Petascale Education Program	Materials to support the teaching and use of parallel and high-performance scientific computing in undergraduate and graduate classrooms and workshops. http://shodor.org/petascale
CDER	Center for Parallel and Distributed Computing Curriculum Development and Educational Resources	PDC core curricula, an early adopters program, and a book project on Topics in Parallel and Distributed Computing Enhancing the Undergraduate Curriculum. https://tcpp.cs.gsu.edu/curriculum/?q=node/21183
CSERD	Computational Science Education Reference Desk	Curriculum materials and workshops for faculty funded by the National Science foundation (plus internship and fellowship for students) http://shodor.org/cserd
CSinParallel	CSinParallel is supported by a grant from NSF-TUES	A searchable database of undergraduate teaching modules and links to platform setup, libraries, and packages for teaching parallel computing. https://csinparallel.org/index.html
HPCU	High Performance Computing University	Virtual organization that shares educational and training materials for computational modeling and digital humanities and social sciences. http://www.hpcuniversity.org
iPDC	Integrating Parallel and Distributed Computing Modules	Lesson plans for plugged and unplugged activities to teach PDC. https://www.csc.tntech.edu/pdcincs/index.php/ipdc-modules
NICS	National Institute for Computational Science	Education, Outreach and Training (EOT) program provides collaboration opportunities and resources for institutions and training events for students. https://www.nics.tennessee.edu/eot/hpc-training
OSCER	University of Oklahoma's Supercomputing Center for Education & Research	Provides a workshops's serie titled "Supercomputing in Plain English" with 11 recorded videoconference presentations. http://www.oscer.ou.edu/education.php
SIGHPC	Special Interest Group on High Performance Computing	Educational and training resources for HPC, Computational Science Systems (general and domain specific), system professionals, and pre-university (K-12) practitioners. https://sighpceducation.acm.org/resources.html

Table 4: HPCEd Hardware Resources.

Name	Program/Institution Name	Resource Description and Web Link
AREN	Alabama Research and Education Network	Access is free to all educators throughout the state of Alabama. https://www.asc.edu
Chameleon	NSF-funded testbed for Computer Science experimentation	Available to Computer Science researchers and students. Deeply-reconfigurable, with capabilities for networking, distributed and cluster computing, and security. https://www.chameleoncloud.org/
CDER	Center for Parallel and Distributed Computing Curriculum Development and Educational Resources	CDER cluster freely available for faculty and students. Offers heterogeneous clusters maintained at Georgia State University. https://tcpp.cs.gsu.edu/curriculum/?q=node/21615
DiaGrid	Distributed research computing grid	Provides free web-based access to high performance and high throughput computing to users. https://diagrid.org
JetStream	JetStream is led by the Indiana University Pervasive Technology Institute (PTI)	It provides high performance computing resources to education communities. https://jetstream-cloud.org
XSEDE	Extreme Science and Engineering Discovery Environment	Educational portal that allocates storage, and cloud facilities to educators on request. These resources include computing and visualizations. https://www.xsede.org/web/xup/allocations/education

4.2.1 Hardware. The overwhelming majority of modern devices, including smart phones, include multiple processing cores. Therefore, beginners can use standard resources such as laptops, desktop PCs, and general purpose lab computers.

Additionally, a variety of local smaller scale options exist, such as low power dedicated boards like the Raspberry Pi or Arduino. An orthogonal concern is the availability of programmable graphics processing units (GPUs) for use in HPC.

These general purpose resources alone are often not enough to teach HPC/PDC effectively to its full extent. Dedicated hardware resources such as HPC clusters, high end servers, or other such platforms are needed to replicate the conditions of HPC working environments and implement large projects. These hardware resources also include parallel IaaS provided by cloud providers such as Amazon AWS, Google Cloud, Microsoft Azure, etc. The advantage of cloud resources is that they are globally available and require minimal maintenance but they can be costly, especially for large or long running classes. However, they are reasonable tools for short duration courses such as workshops and tutorials.

Another option, available on some campuses, is a local cluster, often bought via hardware investment through an academic unit and incurring ongoing maintenance and access fees. This model has several benefits including consistent usage and performance, and gives students an opportunity to use and access modern supercomputers. However, limited access is the primary drawback. Larger consortiums exist for institutions lacking their own hardware infrastructure. National research organizations, such as NSF, also make time available for students and researchers. See Table 4 for a list of free hardware resources for educators.

4.2.2 Software. In addition to hardware resources, a mix of software infrastructure is required to support basic parallel education.

Parallelism exists or can be supported in a variety of modern languages from Java to Python. Java supports a native threading abstraction as well as several libraries which can be used for distributed computing, e.g., MPI support. Python also supports a variety of high performance libraries, both at the shared and distributed memory levels. Julia is another useful parallel programming language that support the three main features for concurrent and parallel programming: asynchronous tasks, multi-threading, and distributed computing [73].

For performance oriented work, the Message Passing Interface (MPI) is commonly used to write code distributed across multiple computer nodes. One of MPI's strengths is its relative simplicity as a C and Fortran library with a simple send/receive model. It is tuned for performance on a wide range of modern machines. However, it forces the programmer to reason directly about the hardware they are using. For advanced projects other distributed alternatives exist, such as Charm++ or Chapel, but these are less widely used in production scale applications.

In addition to, alongside, or instead of these options, students can also use languages which only parallelize their code on a single system. The most common, OpenMP, is often taught (and programmed) alongside MPI. OpenMP is a lightweight compiler-based tool which is used to annotate loops by the programmer, which in turn generate parallel code via a supporting compiler. started is lower as OpenMP is not a separate library. This class of languages also includes GPU programming languages such as CUDA and OpenCL/OpenACC, which are more specialized.

Students will also require a standard set of development tools including an editor and compiler (with support for the above libraries). Also helpful would be a package manager (HomeBrew, MacPorts, Cygwin) or for HPC applications, Spack [51].

Performance-oriented lessons would also benefit from the use of profiling tools, which could range from OS tools such as *gprof* to specialized parallel profilers that exist as part of the language or library. Finally, beyond rudimentary debugging, a parallel debugger such as Alline DDT makes debugging possible for complicated code.

4.3 Overview of Educational Papers

Although there are no journals specifically directed towards educational research in HPC and PDC, some journals have special issues on these topics and others publish education research in these areas. Table 5 lists two special issues that we classify as 'dedicated' to HPC/PDC research as well as a 'general' computing education research journal, JCSC, which has practitioner reports of HPC education. The HPCEd coverage is stronger in HPC-related conferences that have a workshop or a sub-topic on education.

Note that other journals such as The Journal of Supercomputing, Supercomputing Frontiers and Innovations, and the IEEE Transactions on Parallel and Distributed Systems publish technical research work in HPC and PDC, and they are referred to for technical content. SIGCSE also has a list of journals and conferences that cover computing education, which occasionally publish HPC/PDC related research work.

4.3.1 Relevant Papers. To identify key HPCEd papers for this review, we searched the ACM digital library for papers with "High Performance Computing" in the abstract, either "teaching" or "learning" in the content, and that have been mapped to content area CSS 2012- Computing Education. This search resulted in a list of 71 publications that were then checked for relevant examples and approaches. These papers fall into three broad categories: hardware, software, and pedagogical/logistical support.

Table 5: Where to find HPC Education Papers

Venue	Description
JPDC Special Issue	Journal of Parallel and Distributed Computing Special issue: Keeping up with Technology: Teaching Parallel, Distributed and High Performance Computing November 2019.
TOCE Special issue	Transaction on Computer Education: special issue on concurrent and parallel programming January 2013.
J. of Computing Sciences in Colleges	Conference proceedings for the regional conferences sponsored by the Consortium for Computing Sciences in Colleges (CCSC).
EduHPC	Workshop held in conjunction with the SC Conference (2018 onward).
EduHiPC	Targeted towards educators in Asia (2018 onward).
EduPAR	Targeted towards PDC pedagogy and curricula (2011 onward).
Euro-EDUPAR	Targeted towards educators in Europe (2015 onward).

Due to time limitations we could not carry out a systematic review. Instead, three authors read the list and identified a subset of paper that best illustrates ideas and experiences integrating HPC into the undergraduate curriculum. That subset has been listed in Appendix B.3 for interested readers.

5 HPC OPPORTUNITIES IN COMPUTING

This section examines how incorporating HPC into a computing curriculum can result in several benefits to contemporary computing education. In particular, HPC is now being used more intensively by educators and researchers in artificial intelligence, data science and analysis, and internet of things among others. The list will continue to expand as more educators and scientists are introduced to the possibilities of using HPC, bringing their own unique understanding of how it can be used in their fields within computing. Section 6 explores HPC opportunities in non-computing fields.

5.1 Low-fidelity HPC Integration

The interdisciplinary nature of HPC allows for numerous educational opportunities. Here, we provide examples of "low-fidelity" HPC integration into otherwise non-HPC courses. The idea here is likened to *low-fidelity prototyping* [106], where software designers focus on depicting concepts without significant time and financial investment in the development process. Such low-fidelity prototyping also requires little programming skill, allowing the implementor to focus on fundamental design [106]. Thus, this approach provides an opportunity for course designers with little HPC expertise to inject fundamental HPC concepts—in the form of low-fidelity HPC components—into non-HPC courses. By identifying examples of such low-effort HPC integration, the way is paved for instructors who are non-expert in HPC (or time-restricted) to gradually introduce its elements into their courses.

5.1.1 HPC in courses. HPC—like any other fast-paced, advanced and continuously evolving computing area—is rich with several exploratory and research opportunities, which lends itself well to meeting learning outcomes of broader Project-Based Learning (PBL), such as capstone and research courses. PBL has been shown not only to help students understand a subject area better, but also to provide motivation [19]. PBL is common in engineering education, allowing students to apply knowledge in a self-directed manner [88]. For example, HPC project topics have been integrated into a PDC, though not necessarily HPC, coursework for software engineering and computer systems engineering students [53, 54]. In this course, instructors teach the fundamentals of parallel computing and students can then self-learn and explore a wider variety of topics not formally taught by the instructors; these topics could include HPC. Students subsequently relay core concepts they learned in their individual projects through oral presentations in class.

5.1.2 Broader Efforts. Several efforts have sought to integrate parallel programming concepts into courses that are not necessarily dedicated to HPC, giving rise to valuable lessons for other efforts looking to build on these. Despite the serious challenges of teaching parallelism to undergraduate students early in their course progression, there are benefits in exposing students to such parallel models early. By merely letting students acknowledge the existence of parallelism, they become aware of parallelism and can apply it in later courses should they need it. In addition, students are typically thrilled to learn a real-world and relevant "hot topic" [59]. Even without practical opportunities, a breadth-first approach in the form of paper discussions allows students to appreciate the wide-ranging relevance of parallelism [105].

5.1.3 Educational Tools. Although few educational tools exist for teaching "strictly HPC" content, several do exist for fundamental concepts that are relevant to parallel computing. *PDC Unplugged* provides a large collection of activities, categorized into various course and curricula views, to help instructors find relevant material for their PDC courses [83]. ParallelAR uses an analogy to visualize fundamental parallel scheduling policies, focusing on concepts without requiring students to program [2]. Also recognizing the importance of visualization, TSGL is a thread-safe graphics library that helps instructors and students visualize the underlying execution of their parallel programs [6]. Block-based programming extensions are also starting to emerge, for both parallel programming [49] and distributed programming [21].

5.2 Data Science

Data Science is an interdisciplinary area that applies methods and algorithms to structured or unstructured data to extract information and hopefully knowledge [77]. Discovering knowledge and patterns in a complex dataset is a computationally intensive task. This is especially the case when dealing with large-size data with many attributes (possibly big data), or unstructured data with hidden interconnectivity [41]. Consequently, HPC is needed to provide at least part of the solution to the computational challenge of data science.

5.2.1 HPC in courses. This pivotal need for HPC in Data Science has led to the introduction of High Performance Data Analytics (HPDA) as a new sub-discipline of data science by Pacific Northwest National Laboratory in 2013 [24]. Researchers active in HPDA have been exploring, evaluating, and demonstrating the application of HPC technologies to data analytics challenges. On the educational front, the ubiquity of HPC in data science applications means that many data science programs include material on Hadoop, Mapreduce, Spark, NoSQL, and other concepts in their coursework. The Park City report specifically mentions these topics in its curriculum guidelines for undergraduate data science programs [40].

5.2.2 Broader Efforts. Many data science programs are situated in computing-focused departments or colleges. Computing-heavy curricula are more likely to present HPC concepts, while business and statistics-oriented programs are not. Given the importance of HPC in this field, this educational gap could be filled by the low fidelity approach discussed earlier, with more advanced courses available to the interested student.

5.2.3 Educational Tools. From the tools and education aspects, HPC programming interfaces such as MPI, OpenMP, PGAS (OpenSHMEM), Spark, Hadoop, and their software stacks are intensively utilized in data science [10]. These parallel and distributed APIs have generally met data scientists' requirements in terms of high computational performance, while Big Data frameworks such as Spark have performed likewise in terms of high-level programming, resiliency and I/O handling [12]. For less computing-intensive data science programs, plug-and-play options that require little technical intervention are needed to reach educational goals.

5.3 Artificial Intelligence

Artificial Intelligence is the science and engineering of making intelligent machines, where the intelligence is the computational part of achieving goals such as learning, search, optimization, and decision support, among others [85]. AI algorithms have been around since the late 1950s, many of which could not be efficiently implemented until the availability of sufficient computational capacity needed to deal with high computational complexity. Hence, it comes as no surprise that the AI community has increasingly used HPC infrastructures to achieve the vision of AI to build machines and applications that help humans make better and more informed complex decisions [129]. Note also the contribution of AI methods in providing solutions to different elements of HPC systems such as load balancing, job scheduling, runtime prediction, and optimization of resource utilization [70].

5.3.1 HPC in courses. Figure 5 illustrates the sharp increase in the amount of computing usage in AI training algorithms since 2012, which reflects the availability of more powerful HPC platforms to run more complex algorithms [7]. The increasing computational power has led to better AI search algorithms, which in turn has led to the development of intelligent applications such as AlphaGo, an AI search program and the first computer program to defeat a Go world champion and in fact, the first program to defeat any professional human Go player [31].

In general, AI search algorithms are classified among the most computationally heavy algorithms. To understand how computationally demanding these algorithms are, we can consider the game of chess. There are about 10^{81} atoms in the universe, while the lower bound of the game tree complexity of chess is 10^{120} (Shannon number [111]). Because of this level of complexity, it took until 1997 for Deep Blue to beat Garry Kasparov (chess world champion), and until 2016 for AlphaGo to beat Lee Sedol, 18-time world champion [31]. These types of algorithms cannot be discussed in class without teaching students their parallel implementations.

5.3.2 Broader Efforts. There are many popular use cases for AI in HPC. Some of these AI-based applications are image recognition and classification, trading strategies, predictive maintenance, object identification and detection, patient data processing, image query, geospatial feature detection, social media content management and distribution, cyber defense and operation, unstructured content-based bots, sensor data fusion and analysis, among others [74]. These applications for HPC and AI provide opportunities for HPC concepts to be taught in targeted non-computing courses such as economics and engineering.

5.3.3 Educational Tools. AI tools for computing courses are many, from NLP programs like the Natural Language Toolkit (nltk) [18] to deep learning's TensorFlow [1] and Keras [32]. For non-computing classes such as economics, business, or engineering, tools that do not require extensive modification or setup are needed.

5.4 Other Computing Areas

There are several other areas of computing where HPCEd would be useful.

For example, as the massive number of interconnected devices capable of sensing and actuation—the Internet of Things (IoT)–become increasingly commonplace, they provide tremendous opportunities for the creation of new services with tangible benefit to society. However, the two-layered architectures of systems based on cloud/IoT devices are failing to support real-time communication and data processing needs by these billions of devices to meet quality of service expectations. Hence, as the drive to push processing more towards resource constrained fog/edge devices, a larger need arises for developing and utilizing high performance computing competencies. At the least, fog/edge devices will need to contribute to computationally intensive data quality activities, privacy preservation, mobility support, the preservation of network semantics, and more importantly, probably making use of intensive machine learning [17].

Additionally, data quality activities alone may include the normalization, filtering, and aggregation of data. Massive data will have to be put in common formats, homogenized to preserve semantics, and serialized for efficient transmission. Furthermore, filtration activities will work on eliminating redundant or faulty data to relieve subsequent layers from the burden of traffic and processing. Finally, data will probably be fused for more abstracted meanings. All such activities could be relying on rather intensive algorithms that possibly involve spatio-temporal analysis, statistical modeling, and machine learning [17]. Such time-critical data must be processed as fast as possible for further usage in IoT systems. Furthermore, integration of IoT and HPC systems is needed to hide away resource complexity related to context awareness, actuation, sensing, and the involved heterogeneity of devices [38, 39].

Another example comes from the onset of the COVID-19 pandemic, as it has stimulated HPC application and research, from medical forensics to social media behaviors and others. The COVID-19 High Performance Computing Consortium [36] engages industry and academic members to provide research funding for projects using machines that range from small computer clusters to large supercomputers. Researchers have used 17 parameters on a supercomputer to analyze 63 million COVID-19 messages from 13 million Twitter customers over a six-month period [60]; the goal here was to determine a message's degree of relevance to the pandemic, the emotional level of an author, and the intensity of fear in the communication. The authors then presented a statistical analysis of their research.

HPCEd can build on these tools to educate the next generation of computing professionals. A recent publication suggests the use of the current COVID-19 as a pandemic supercomputing teaching tool for reacting to and addressing future pandemics [63].

6 HPC OPPORTUNITIES IN OTHER DOMAINS

In Section 5, we explored the impact of HPCEd within Computing, and we do the same in this section for other disciplines. Today, HPC resources are a critical part of many science and engineering programs and disciplines. These resources play an increasingly important role in preparing students for many careers in both industry and academic fields. As our ability to collect big data in different disciplines increases, the need to be able to effectively analyse the data also increases.

Modeling and simulation using HPC has become a critical part of both research and applications across a wide range of disciplines. This includes virtually every field in physical and biological sciences and engineering as well as business, finance, and the social sciences. The examples in this section illustrate that a combination of domain and HPC computing knowledge is required in order to effectively develop and apply those models.

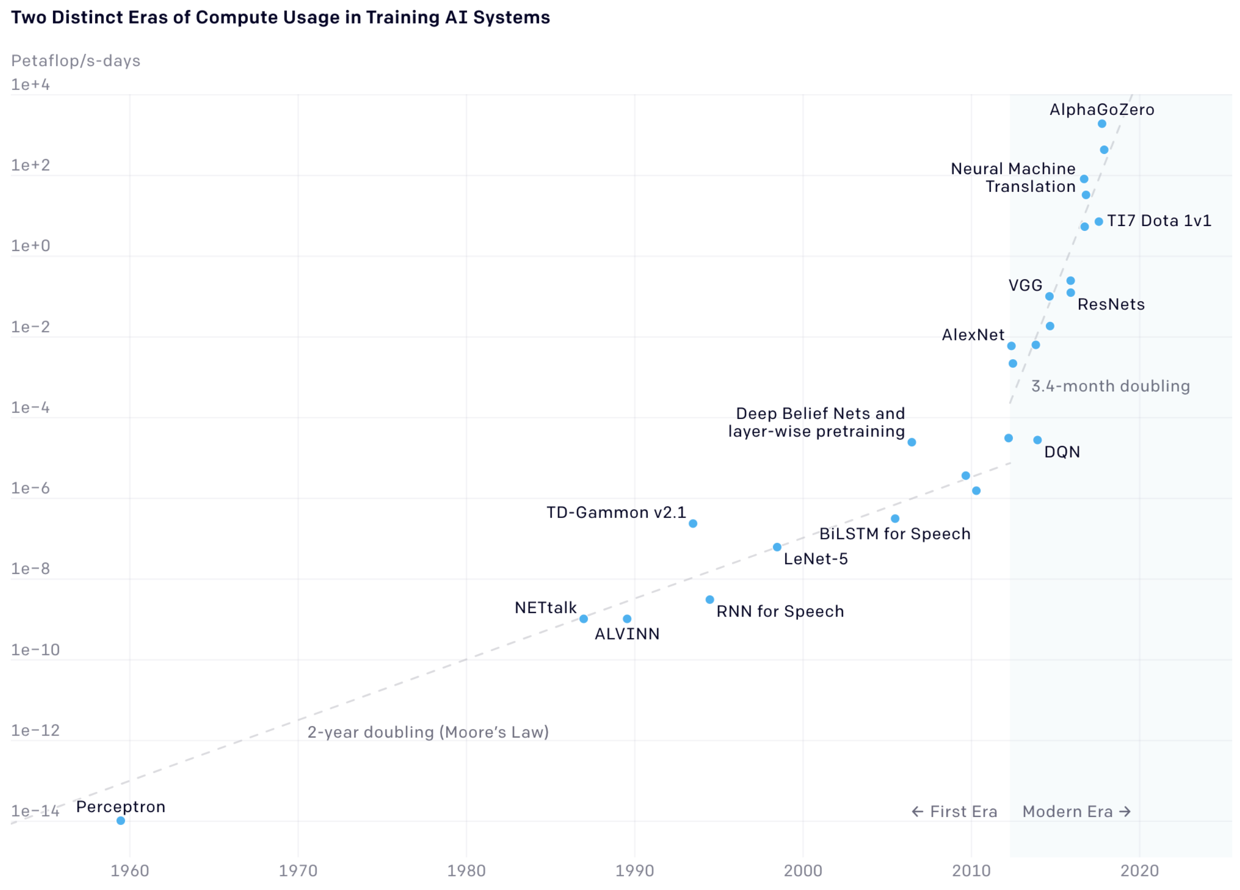

Figure 5: Two distinct eras of computational usage for AI [7].

6.1 Atmospheric Science

6.1.1 Domain Description. Atmospheric science research's contribution to understanding the earth's atmosphere is crucial as climate change and pollution continue to affect global weather patterns. Models in atmospheric science divide the atmosphere into a three-dimensional grid to calculate how matter and energy interact. Mathematical equations represent the physics of matter and energy transfer in each of the cells and pass the resulting quantities to neighboring cells.

6.1.2 Where HPC helps. HPC is required to scale the models to represent atmospheric dynamics at local, continental, or global scales. Current models take advantage of multiple processors to perform the calculations for each of the grid cells and then pass the results for neighboring cells for each time period.

Knowledge of many of the HPC concepts summarized in Section 3 is required to effectively run these models on HPC systems. Those concepts include a basic understanding of parallel computing principles, the algorithms that are used to make the calculations, how the model will scale on HPC systems, and the management of large- scale input and output datasets. Deeper expertise in parallel algorithms, parallel input-output, shared memory, and other HPC knowledge elements is required for researchers seeking to improve the scope, accuracy, and performance of models such as Weather Research and Forecasting (WRF).

6.1.3 Example application. One example of such a model is the WRF model developed through a collaborative partnership with the National Center for Atmospheric Research [123]. Users of the model must have knowledge of the underlying domain and the mathematical representation of the physical processes being simulated. In addition, they must understand the required underlying data input structure and the impact of model parameter selection on model accuracy and results. Thus, the combination of domain and HPC knowledge is critical for atmospheric scientists to build large-scale models on HPC systems.

6.2 Meteorology

6.2.1 Domain Description. Meteorology is a branch of atmospheric science with a particular focus on weather forecasting. As global warming contributes to severe and highly impacting weather around the globe, the need for more accurate simulation and prediction of weather conditions emerges. To name one, forest fires have been impacting millions of people in North American, European, and Australian cities over the past couple of years. According to Accuweather, the cost of forest fires in California alone in 2019 amounted to about US$80 billion [71].

6.2.2 Where HPC helps. The National Oceanic and Atmospheric Administration (NOAA) published its 2015-2020 high performance computing strategic plan and cites the need for major investment needed in the domain of HPC. NOAA's Warn on Forecast system is expected to push exascale computing by 2023. Expanding the area covered by a particular forecast significantly multiplies the necessary computation power needed to achieve forecast accuracy [123].

6.2.3 Example application. Forest file pattern prediction is seen as an extremely complex problem that involves notable uncertainty, and must be delivered under the strictest real time constraints [23]. Since 2017, HPC's ability to rapidly provide on-demand analysis of massive data has been a critical part of fighting wildfires [87].

6.3 Critical Infrastructure Protection

6.3.1 Domain Description. Critical infrastructure refers to the systems and processes necessary to maintain a functioning society. The US Department of Homeland Security identified sixteen critical infrastructure sectors, including commercial facilities, finance, critical manufacturing, food and agriculture, dams, and transportation systems [33].

6.3.2 Where HPC helps. Computing, and specifically HPC, underpins all sectors, providing the capability to rapidly analyze data and develop realistic models. In the transportation sector, HPC has been used to design complex discrete event simulation models to analyze congestion and other issues in an urban environment [26].

6.3.3 Example application. Especially in the area of disaster management, the situational awareness necessary for coherent, swift response to emergency events requires analytic capability at all levels of government. For example, in the area of emergency services, efficient response to disruptive events requires fusing data in real time from multiple streaming sources, and analyzing existing models, potential scenarios, and established emergency response plans to recommend a course of action. The fused data provides emergency managers with information needed to support rational decision making in the face of uncertainty and stress.

6.4 Biological Sciences

6.4.1 Domain Description. Biological sciences range from molecular levels to generations-long view of the world through evolution and pandemic tracking. Study of these sciences impacts fields such as health care, ecology, wildlife management, and marine environments.

6.4.2 Where HPC helps. The partnership between biology and computer science, and specifically high performance computing, goes back decades, leading to crucial advances in both fields. HPC is often referred to as the computational microscope [44] and is an essential tool for practically all modern biologists.

6.4.3 Example application. A particular subfield that is well suited to leverage HPC platforms is molecular dynamics. Such applications are often computationally intensive and require a relatively small amount of data. Additionally, they employ straightforward physics, suitable to diverse HPC platforms such as GPUs, and utilize algorithms that are reasonable to prototype, explain, test, and so on. Other major users of high performance computing resources in the biological sciences include genome assembly platforms. These applications primarily consist of long parallel pipelines, are throughput oriented, and are able to leverage accelerators, e.g., FPGAs, and cloud technologies. There are other diverse biological fields and applications, from epidemic modeling to the statistics of phylogeny generation in evolutionary fields.

7 RECOMMENDATIONS

The working group's overarching goal was to explore the current state of HPC Ed and recommend ways to improve HPCEd, based on the issues that we explored in this paper. In short, our recommendations stem from the earlier sections.

7.1 Building a Professional HPC Educator Community

Attracting HPC researchers with good communication skills to the HPC education community can greatly reduce the pressure on instructors and allow the content of HPC education to be updated as the state of the art progresses. HPC researchers can also benefit from the HPC education community. For example, researchers can decompose the latest scientific research questions and post them. The questions can be followed by more students worldwide, and solved; research topics can attract the attention of more outstanding students all over the world, allowing their own research to expand in new ways. In short, we need to use the HPC education community to build a bridge between HPC researchers and students.

7.2 Developing an Understanding of HPC

Disagreements and misunderstandings of the ways PDC and HPC fit together are common. There are many definitions and notations in each field, especially as relates to what each domain entails, their programming, tools, theoretical aspects, and supporting architectures. It is imperative to build a common understanding in the community about each domain, and probably a recommended set of curricula for basic, intermediate, and advanced level studies in each of those domains.

7.3 Building HPC Competencies

Educational programs are strongly encouraged to develop a set of competencies needed for their programs to enable high performance computing education. Recent efforts in the academic community have proposed competencies as a core set of knowledge, skills, and dispositions that need to be attained. While many programs focus on the knowledge and skills alone, the dispositions that prescribe a requisite character or quality in task performance are very much lacking in the specification of academic programs. Programs need to be alert to providing students with an experiential environment for students to immerse themselves in a hands-on setting.

7.4 Building Infrastructures for HPC Education

General purpose resources are not enough to teach either PDC or HPC to the extent needed to achieve high competence in the areas. Given that it is very difficult to replicate real HPC settings using working environments available at most university campuses, students and educators need easier access to modern infrastructure for HPC learning and experimentation. This access could be provided by raising awareness about the available infrastructures and how to build them at low cost. One possible approach to accomplish low-cost construction would be to use "Budget Beowulf Clusters" that may not be truly HPC, but help students understand the concepts of PDC [5].

7.5 Providing HPC Experiences in Contemporary Computing

HPC spans multitudes of other fields, notably AI, Data Science, IoT, and Big Data. Domain knowledge is an important part of HPC education so that students can gain practical experience on how to make use of HPC in solving real life problems. Finally, we note that not all aspects of HPC education can be provided formally in the classroom, as there will always be a need for students to learn on the job, such as an internship or co-op, to solve a new class of problems that require HPC.

ACKNOWLEDGMENTS

Chen acknowledges support provided by the 2019 Hunan Province Higher Education Teaching Reform Research Foundation of China (Teaching Practice of Training High-Performance Computing Talents Relying on High-level Scientific Research). Raj acknowledges support provided by the National Science Foundation under Awards 1433736, 1922169, and 2021287.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Marin Abernethy, Oliver Sinnen, Joel Adams, Giuseppe De Ruvo, and Nasser Giacaman. 2018. ParallelAR: An augmented reality app and instructional approach for learning parallel programming scheduling concepts. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, Vancouver, Canada, 324–331.

[3] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. *Computer Science Curricula 2013*. Technical Report. ACM Press and IEEE Computer Society Press. https://doi.org/10.1145/2534860 Accessed: November 14, 2017.

[4] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2016. *Computer Engineering Curricula 2016*. Technical Report. ACM Press and IEEE Computer Society Press. https://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf

[5] Joel C. Adams, Jacob Caswell, Suzanne J. Matthews, Charles Peck, Elizabeth Shoop, and David Toth. 2015. Budget Beowulfs: A Showcase of Inexpensive Clusters for Teaching PDC. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE ’15)*. Association for Computing Machinery, New York, NY, USA, 344–345. https://doi.org/10.1145/2676723.2677317

[6] Joel C. Adams, Patrick A. Crain, Christopher P. Dilley, Christiaan D. Hazlett, Elizabeth R. Koning, Serita M. Nelesen, Javin B. Unger, and Mark B. Vande Stel. 2018. TSGL: A tool for visualizing multithreaded behavior. *J. Parallel and Distrib. Comput.* 118 (2018), 233–246.

[7] Dario Amodei and Danny Hernandez. 2018. OpenAI. https://openai.com/blog/ai-and-compute/

[8] James P. Anderson, Samuel A. Hoffman, Joseph Shifman, and Robert J. Williams. 1962. D825 - a Multiple-Computer System for Command amd Control. In *Proceedings of the December 4-6, 1962, Fall Joint Computer Conference*. Association for Computing Machinery, New York, NY, USA, 86–96. https://doi.org/10.1145/1461518.1461527

[9] Scottie Andrew. 2020. The world’s fastest supercomputer identified chemicals that could stop coronavirus from spreading, a crucial step toward a treatment. https://edition.cnn.com/2020/03/19/us/fastest-supercomputer-coronavirus-scn-trnd/index.html

[10] Apache Software Foundation. 2019. Apache Hadoop. https://hadoop.apache.org.

[11] Apache Software Foundation. 2020. Apache Spark: a unified analytics engine for large-scale data processing. https://spark.apache.org/ Accessed: November 4, 2020.

[12] HamidReza Asaadi, Dounia Khaldi, and Barbara Chapman. 2016. A comparative survey of the HPC and big data paradigms: Analysis and experiments. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, Taipei, 423–432.

[13] Association for Computing Machinery and the IEEE Computer Society. 2020. *Computing Curricula 2020 (CC2020) Paradigms for Future Computing Curricula*. Technical Report. ACM/IEEE. Draft of November 2020, http://www.cc2020.net/.

[14] Didier El Baz. 2014. IoT and the Need for High Performance Computing. In *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*. IEEE, Beijing, China, 6. https://doi.org/10.1109/IIKI.2014.8

[15] Olivier Beaumont, Brett A. Becker, Ashley DeFlumere, Lionel Eyraud-Dubois, Thomas Lambert, and Alexey Lastovetsky. 2019. Recent Advances in Matrix Partitioning for Parallel Computing on Heterogeneous Platforms. *IEEE Transactions on Parallel and Distributed Systems* 30, 1 (2019), 218–229.

[16] Brett A. Becker and Keith Quille. 2019. 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE ’19)*. Association for Computing Machinery, New York, NY, USA, 338–344. https://doi.org/10.1145/3287324.3287432

[17] Paolo Bellavista, Javier Berrocal, Antonio Corradi, Sajal Das, Luca Foschini, and Alessandro Zanni. 2018. A Survey on fog computing for the Internet of Things. *Pervasive and Mobile Computing* 52 (12 2018). https://doi.org/10.1016/j.pmcj.2018.12.007

[18] Steven Bird, Evan Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O’Reilly Media Inc, Sebastopol, California.

[19] Phyllis C. Blumenfeld, Elliot Soloway, Ronald W. Marx, Joseph S. Krajcik, Mark Guzdial, and Annemarie Palincsar. 1991. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational psychologist* 26, 3-4 (1991), 369–398.

[20] André B. Bondi. 2000. Characteristics of Scalability and Their Impact on Performance. In *Proceedings of the 2nd International Workshop on Software and Performance (WOSP ’00)*. ACM, New York, NY, USA, 195–203. https://doi.org/10.1145/350391.350432

[21] Brian Broll, Ákos Lédeczi, Hamid Zare, Dung Nguyen Do, János Sallai, Péter Völgyesi, Miklós Maróti, Lesa Brown, and Chris Vanags. 2018. A visual programming environment for introducing distributed computing to secondary education. *J. Parallel and Distrib. Comput.* 118 (2018), 189–200.

[22] Richard Brown, Elizabeth Shoop, Joel Adams, Curtis Clifton, Mark Gardner, Michael Haupt, and Peter Hinsbeeck. 2010. Strategies for Preparing Computer Science Students for the Multicore World. In *Proceedings of the 2010 ITiCSE Working Group Reports (ITiCSE-WGR ’10)*. Association for Computing Machinery, New York, NY, USA, 97–115. https://doi.org/10.1145/1971681.1971689

[23] C. Brun, T. Artes, A. Cencerrado, T. Margalef, and A. Cortés. 2017. A High Performance Computing Framework for Continental-Scale Forest Fire Spread Prediction. *Procedia Computer Science* 108 (2017), 1712–1721.

[24] V. G. Castellana, M. Minutoli, S. Bhatt, K. Agarwal, A. Bleeker, J. Feo, D. Chavarría-Miranda, and D. Haglin. 2017. High-Performance Data Analytics Beyond the Relational and Graph Data Models with GEMS. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, Lake Buena Vista, FL, 1029–1038.

[25] CDER Center. 2020. NSFIEEE-TCPP Curriculum Initiative. https://tcpp.cs.gsu.edu/curriculum/?q=node/21183.

[26] C. Chan, B. Wang, J. Bachan, and J. Macfarlane. 2018. Mobiliti: Scalable Transportation Simulation Using High-Performance Parallel Computing. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Maui, Hawaii, 634–641.

[27] Juan Chen, Brett Becker, Youwen Ouyang, and Li Shen. 2020. What Influences Students’ Understanding of Scalability Issues in Parallel Computing?. In *Proceedings of Workshop on HPC Education and Training for Emerging Technologies (HETET2020), in conjunction with ISC2020*. ACM, Digital Event.

[28] Juan Chen, Yingjun Cao, linlin Du, Youwen Ouyang, and Li Shen. 2019. Improve Student Performance Using Moderated Two-Stage Projects. In *Proceedings of ACM Global Computing Education Conference (CompEd2019)*. ACM, New York, NY, USA, 201–207. https://doi.org/10.1145/3300115.3309524

[29] Juan Chen, John Impagliazzo, and Li Shen. 2020. High-Performance Computing and Engineering Educational Development and Practice. In *Proceedings of the 50th Frontiers in Education 2020 (FIE2020)*. IEEE, Uppsala, Sweden.

[30] Juan Chen, Li Shen, Jianping Yin, and Chunyuan Zhang. 2018. Design of Practical Experiences to Improve Student Understanding of Efficiency and Scalability Issues in High Performance Computing (Poster). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE2018)*. ACM, New York, NY, USA, 1090–1090. https://doi.org/10.1145/3159450.3162239

[31] Jim X Chen. 2016. The evolution of computing: AlphaGo. *Computing in Science & Engineering* 18, 4 (2016), 4–7.

[32] François Chollet et al. 2015. Keras. https://keras.io.

[33] CISA. 2020. Critical Infrastructure Sectors. https://www.cisa.gov/critical-infrastructure-sectors,.

[34] International Business Machines Corporation. 1955. IBM 704 Manual of Operation.

[35] Natalie J. Coull and Ishbel M.M. Duncan. 2011. Emergent requirements for supporting introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences* 10, 1 (2011), 78–85.

[36] COVID-19 High Performance Computing (HPC). 2020. COVID-19 High Performance Computing Consortium. https://covid19-hpc-consortium.org/

[37] Cray. 2012. Enabling Scientific Breakthroughs at the Petascale. https://www.cray.com/enabling-scientific-breakthroughs-petascale.

[38] L. de Souza Cimino, J. E. E. d. Resende, L. H. Moreira Silva, S. Queiroz Souza Rocha, M. de Oliveira Correia, G. S. Monteiro, G. N. de Souza Fernandes, S. G. Moreira Almeida, A. L. Barroso Almeida, A. L. Lins de Aquino, and J. de Castro Lima. 2017. IoT and HPC Integration: Revision and Perspectives. In *2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, Curitiba, PR, Brazil, 132–139.

[39] Leonardo de Souza Cimino, José Estevão Eugênio de Resende, Lucas Henrique Moreira Silva, Samuel Queiroz Souza Rocha, Matheus de Oliveira Correia, Guilherme Souza Monteiro, Gabriel Natã de Souza Fernandes, Renan da Silva Moreira, Junior Guilherme de Silva, Matheus Inácio Batista Santos, Andre Luiz Lins Aquino, André Luís Barroso Almeida, and Joubert de Castro Lima. 2019. A middleware solution for integrating and exploring IoT and HPC capabilities. *Software: Practice and Experience* 49, 4 (2019), 584–616. https://doi.org/10.1002/spe.2630 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2630

[40] Richard D De Veaux, Mahesh Agarwal, Maia Averett, Benjamin S Baumer, Andrew Bray, Thomas C Bressoud, Lance Bryant, Lei Z Cheng, Amanda Francis, Robert Gould, et al. 2017. Curriculum guidelines for undergraduate programs in data science. *Annual Review of Statistics and Its Application* 4 (2017), 15–30.

[41] Vasant Dhar. 2013. Data science and prediction. *Commun. ACM* 56, 12 (2013), 64–73.

[42] Applied Science Division and Programming Research Department. 1956. *Fortran automatic coding system for the IBM 704: Programmer's reference manual.* Technical Report. International Business Machines Corporation.

[43] Jack Dongarra and Alexey L Lastovetsky. 2009. *High Performance Heterogeneous Computing*. Vol. 78. John Wiley & Sons, Hoboken, NJ.

[44] Ron O Dror, Robert M Dirks, JP Grossman, Huafeng Xu, and David E Shaw. 2012. Biomolecular simulation: a computational microscope for molecular biology. *Annual review of biophysics* 41 (2012), 429–452.

[45] Philip Enslow. 1977. Multiprocessor Organization—a Survey. *ACM Comput. Surv.* 9, 1 (1977), 103–129. https://doi.org/10.1145/356683.356688

[46] ETP4HPC Association. 2020. European Technology Platform for High Performance Computing. https://www.etp4hpc.eu/, accessed June 14, 2020.

[47] European Commission. 2019. High-Performance Computing. https://ec.europa.eu/digital-single-market/en/high-performance-computing/.

[48] Rob Farber. 2017. *AI-HPC is Happening Now*. Technical Report. insideHPC, USA. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/the-convergence-of-ai-and-hpc.pdf.

[49] Annette Feng, Mark Gardner, and Wu-chun Feng. 2017. Parallel programming with pictures is a Snap! *J. Parallel and Distrib. Comput.* 105 (2017), 150–162.

[50] Association for Computing Machinery (ACM) IEEE Computer Society (IEEE-CS). 2017. Information Technology Curricula 2017 Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology. https://doi.org/10.1145/3173161

[51] Todd Gamblin, Matthew LeGendre, Michael R. Collette, Gregory L. Lee, Adam Moody, Bronis R. de Supinski, and Scott Futral. 2015. The Spack Package Manager: Bringing Order to HPC Software Chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15)*. Association for Computing Machinery, New York, NY, USA, Article 40, 12 pages. https://doi.org/10.1145/2807591.2807623

[52] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google File System. *SIGOPS Oper. Syst. Rev.* 37, 5 (Oct. 2003), 29–43. https://doi.org/10.1145/1165389.945450

[53] Nasser Giacaman and Oliver Sinnen. 2014. EA: Research-infused teaching of parallel programming concepts for undergraduate Software Engineering students. In *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*. IEEE, New York, 1099–1105.

[54] Nasser Giacaman and Oliver Sinnen. 2018. Preparing the software engineer for a modern multi-core world. *J. Parallel and Distrib. Comput.* 118 (2018), 247–263.

[55] Mercedes Gómez-Albarrán. 2005. The teaching and learning of programming: a survey of supporting software tools. *Comput. J.* 48, 2 (2005), 130–144.

[56] S. Gordon, K. Carey, and Vakalis. 2008. A shared, interinstitutional undergraduate minor program in computational science. *Computing in Science and Engineering* 10, 5 (2008), 12–16.

[57] Steven I. Gordon and Katherine Cahill. 2020. The state of undergraduate computational science programs. *Journal of Computational Science Education* 11, 2 (2020), 7–11. https://doi.org/10.22369/issn.2153-4136/11/2/2

[58] Green500 list. 2020. https://www.top500.org/lists/green500/

[59] Thomas R Gross. 2011. Breadth in depth: a 1st year introduction to parallel programming. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, New York, 435–440.

[60] Raj Kumar Gupta, Ajay Vishwanath, and Yinping Yang. 2020. COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes. Cornell University Press. https://arxiv.org/abs/2007.06954

[61] Peter J. Hawrylak, John Hale, and Mauricio Papa. 2017. Undergraduate Educational Pathways for Developing a High-Performance Computing Workforce. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. ACM, New Orleans, 1–4.

[62] Tony Hey, Stewart Tansley, and Kristin M. Tolle (Eds.). 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington. http://research.microsoft.com/en-us/collaboration/fourthparadigm/

[63] Brian T. Horowitz. 2020. How Supercomputing Can Deliver Key COVID-19 Research. Dice, DHI Group. https://insights.dice.com/2020/06/29/how-supercomputing-can-deliver-key-covid-19-research/

[64] HPC University Educators. 2020. Minor Program in Computational Science Competency/Topic Overview. http://hpcuniversity.org/educators/undergradCompetencies/.

[65] Cray Inc. 2020. History: Seymour Cray. https://www.cray.com/company/history/seymour-cray.

[66] InsideHPC. 2017. NCSA Blue Waters Report Shows Economic Benefits of HPC. https://insidehpc.com/hpc-basic-training/what-is-hpc/.

[67] InsideHPC. 2019. HPC Market Five-Year Forecast bumps up to $44 Billion Worldwide. https://insidehpc.com/2019/06/hpc-market-five-year-forecast-bumps-up-to-44-billion-worldwide/.

[68] InsideHPC. 2020. What is high-performance computing? https://insidehpc.com/hpc-basic-training/what-is-hpc/.

[69] Irish Centre for High-End Computing. 2020. About High-Performance computing (HPC). https://www.ichec.ie/news/press-corner/about-high-performance-computing-hpc.

[70] Nigar Ismayilova and Elviz Ismayilov. 2018. Convergence of HPC and AI: Two Directions of Connection. *Azerbaijan Journal of High Performance Computing* 1, 2 (2018), 179–184.

[71] John Roach. 2019. California wildfires will cost tens of billions, AccuWeather estimates. https://www.accuweather.com/en/weather-news/california-wildfires-will-cost-tens-of-billions-accuweather-estimates/612548,.

[72] Donald Johnson, David Kotz, and Fillia Makedon. 1994. Teaching Parallel Computing to Freshmen. In *Proceedings of the Conference on Parallel Computing for Undergraduates*, Chris Nevison (Ed.). Colgate University, Colgate University, Hamilton, NY, 7. http://www.cs.dartmouth.edu/~dfk/research/johnson-freshmen/index.html

[73] JuliaLang.org. 2020. The Julia Programming Language. https://julialang.org/ Accessed: November 4, 2020.

[74] Aditya Kaul and Clint Wheelock. 2017. *Use Cases for Artificial Intelligence in High-Performance Computing*. Technical Report. CRAY Inc.

[75] Scott Lathrop. 2016. A Call to Action to Prepare the High-Performance Computing Workforce. *Computing in Science Engineering* 18, 6 (Nov 2016), 80–83. https://doi.org/10.1109/MCSE.2016.101

[76] Scott Lathrop and Thomas Murphy. 2008. High-Performance Computing Education. *Computing in Science Engineering* 10, 5 (Sep. 2008), 9–11. https://doi.org/10.1109/MCSE.2008.132

[77] Christophe Ley and Stéphane PA Bordas. 2018. What makes data science different? A discussion involving statistics 2. 0 and computational sciences. *International Journal of Data Science and Analytics* 6, 3 (2018), 167–175.

[78] Jenneke Lockoff, Bas Wegewijs, Katja Durkin, Robert Wagenaar, Julia Gonzales, Ann Katherine Isaacs, Luigi F. Donà dalle Rose, and Mary Gobbi. 2010. *A Tuning guide to formulating degree programme profiles: Including programme competencies and programme learning outcomes.* Technical Report. DOE ASCAC Subcommittee, Bilbao, Spain. http://www.core-project.eu/documents/Tuning_Guide_Publicada_CoRe.pdf.

[79] Eugene Loh, Michael L. Van De Vanter, and Lawrence G. Votta. 2005. Can software engineering solve the HPCS problem?. In *Proceedings of the second*

international workshop on Software engineering for high performance computing system applications. ACM, St. Louis, Missouri, 27–31.

[80] Robert Lucas et al. 2014. *Top Ten Exascale Research Challenges*. Technical Report. DOE ASCAC Subcommittee, USA. https://science.osti.gov/-/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf.

[81] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, New York, 55–106.

[82] Raffael Marty. 2013. Cyber Security: How Visual Analytics Unlock Insight. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. Association for Computing Machinery, New York, NY, USA, 1139. https://doi.org/10.1145/2487575.2491132

[83] Suzanne Matthews. 2020. PDCunplugged: A Free Repository of Unplugged Parallel & Distributed Computing Activities. In *EduPar 2020: 10th Workshop on Parallel and Distributed Computing Education*. IEEE, New Orleans, 284–291. https://doi.org/10.1109/IPDPSW50202.2020.00060

[84] Timothy G Mattson, Beverly Sanders, and Berna Massingill. 2004. *Patterns for parallel programming*. Pearson Education, Upper Saddle River, NJ.

[85] John McCarthy and Patrick J Hayes. 1981. Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*. Elsevier, New York, 431–450.

[86] Justin McCurry. 2020. Fugaku, world's fastest supercomputer, searches for coronavirus treatment. https://www.theguardian.com/world/2020/jun/23/fugaku-worlds-fastest-supercomputer-searches-for-coronavirus-treatment

[87] Michael Feldman. 2017. HPC Modeling Used to Help Fight California Wildfires. https://www.top500.org/news/hpc-modeling-used-to-help-fight-california-wildfires/,.

[88] Julie E Mills, David F Treagust, et al. 2003. Engineering education - Is problem-based or project-based learning the answer? *Australasian journal of engineering education* 3, 2 (2003), 2–16.

[89] Julia Mullen, Chansup Byun, Vijay Gadepally, Siddharth Samsi, Albert Reuther, and Jeremy Kepner. 2017. Learning by doing, High Performance Computing education in the MOOC era. *J. Parallel and Distrib. Comput.* 105 (2017), 105–115.

[90] Julia Mullen, Weronika Filinger, Lauren Milechin, and David Henty. 2018. The impact of MOOC methodology on the scalability, accessibility and development of HPC education and training. In *Fifth SC Workshop on Best Pracices for HPC Training and Education at SC18*. IEEE Computer Society and ACM, Dallas, 7.

[91] National Computational Science Institute (NCSI). 2020. National Computational Science Institute. http://www.computationalscience.org/.

[92] NetApp, Inc. 2019. What Is High-Performance Computing? https://www.netapp.com/us/info/what-is-high-performance-computing.aspx.

[93] Marco A. S. Netto, Rodrigo N. Calheiros, Eduardo R. Rodrigues, Renato L. F. Cunha, and Rajkumar Buyya. 2018. HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 1–29.

[94] Jeremy L. O'brien. 2007. Optical quantum computing. *Science* 318, 5856 (2007), 1567–1570.

[95] Kenneth O'Brien, Ilia Pietri, Ravi Reddy, Alexey Lastovetsky, and Rizos Sakellariou. 2017. A Survey of Power and Energy Predictive Models in HPC systems and Applications. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–38.

[96] Oden, Tinsley J. et.al. 2006. Simulation Based-Engineering Science: Report of the NSF Blue Ribbon Panel on Simulation-Based Engineering Science. https://www.nsf.gov/pubs/reports/sbes_final_report.pdf.

[97] Partnership for Advanced Computing in Europe. 2020. Enabling High-Impact Scientific Discovery and Engineering Research & Development. https://prace-ri.eu/, accessed June 14, 2020.

[98] Oliver Peckham. 2020. Global Supercomputing Is Mobilizing Against COVID-19. https://www.hpcwire.com/2020/03/12/global-supercomputing-is-mobilizing-against-covid-19/

[99] Esteban Perez-Wohlfeil, Oscar Torreno, Louisa J. Bellis, Pedro L. Fernandes, Brane Leskosek, and Oswaldo Trelles. 2018. Training Bioinformaticians in High Performance Computing. *Heliyon* 4, 12 (2018), 18. https://doi.org/10.1016/j.heliyon.2018.e01057

[100] Dimitri Perrin, Marija Bezbradica, Martin Crane, Heather J. Ruskin, and Christophe Duhamel. 2012. High-Performance Computing for Data Analytics. In *2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*. IEEE, New York, 234–242. https://doi.org/10.1109/DS-RT.2012.41

[101] Sushil K. Prasad, Almadena Chtchelkanova, Frank Dehne, Mohamed Gouda, Anshul Gupta, Joseph Jaja, Krishna Kant, Anita La Salle, Richard LeBlanc, Andrew Lumsdaine, David Padua, Manish Parashar, Viktor Prasanna, Yves Robert, Arnold Rosenberg, Sartaj Sahni, Behrooz Shirazi, Alan Sussman, Chip Weems, and Jie Wu. 2012. NSF/IEEE-TCPP Curriculum Working Group. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates. Technical Report, IEEE-TCPP. http://tcpp.cs.gsu.edu/curriculum/?q=system/files/NSF-TCPP-curriculum-version1.pdf.

[102] Rajendra K. Raj, Allen Parrish, John Impagliazzo, Carol J. Romanowski, Sherif G. Aly, Casey C. Bennett, Karen C. Davis, Andrew McGettrick, Teresa Susana Mendes Pereira, and Lovisa Sundin. 2019. An Empirical Approach to Understanding Data Science and Engineering Education. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 73–87. https://doi.org/10.1145/3344429.3372503

[103] Daniel A Reed and Jack Dongarra. 2015. Exascale Computing and Big Data. *Commun. ACM* 58, 7 (2015), 56–68.

[104] Reed, D., Bajcsy, R., Fernandez, M., Griffiths, J., Mott, R., Dongarra, J., Johnson, C.,Inouye, A., Miner, W., Matzke, M., and Ponick, T. 2005. Computational Science: Ensuring America's Competitiveness. https://www.nitrd.gov/Pitac/reports/20050609_computational/computational.pdf.

[105] Suzanne Rivoire. 2010. A breadth-first course in multicore and manycore programming. In *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, Milwaukee, Wisconsin, 214–218.

[106] Jim Rudd, Ken Stern, and Scott Isensee. 1996. Low vs. high-fidelity prototyping debate. *interactions* 3, 1 (1996), 76–85.

[107] Daisuke Saito, Ayana Sasaki, Hironori Washizaki, Yoshiaki Fukazawa, and Yusuke Muto. 2017. Program learning for beginners: survey and taxonomy of programming learning tools. In *2017 IEEE 9th International Conference on Engineering Education (ICEED)*. IEEE, Kanazawa, Japan, 137–142.

[108] Miriam Schmidberger and Bernd Bruegge. 2012. Need of software engineering methods for high performance computing applications. In *2012 11th International Symposium on Parallel and Distributed Computing*. IEEE, Munich, Germany, 40–46.

[109] SearchDataCenter. 2020. Definition: high-performance computing (HPC). https://searchdatacenter.techtarget.com/definition/high-performance-computing-HPC.

[110] Shayan Shams, Richard Platania, Kisung Lee, and Seung-Jong Park. 2017. Evaluation of Deep Learning Frameworks Over Different HPC Architectures. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, New York, 1389–1396. https://doi.org/10.1109/ICDCS.2017.259

[111] Claude E. Shannon. 1950. XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41, 314 (1950), 256–275. https://doi.org/10.1080/14786445008521796 arXiv:https://doi.org/10.1080/14786445008521796

[112] SIAM Working Group on CSE Undergraduate Education. 2006. Undergraduate Computational Science and Engineering Education. https://www.siam.org/Portals/0/Publications/Reports/CSE_Report.pdf?ver=2018-03-16-161618-620.

[113] Susan Squires, WG Tichy, and Lawrence Votta. 2005. What do programmers of parallel machines need? A survey. In *Second Workshop on Productivity and Performance in High-End Computing (P-PHEC)*. IEEE, San Francisco, 8.

[114] Susan Squires, M. Van De Vanter, and L. Votta. 2006. Yes, There Is an Expertise Gap In HPC Applications Development. In *Third Workshop on Productivity and Performance in High-End Computing (PPHEC06)*. IEEE, Austin, Texas, 6.

[115] Andrew Steane. 1998. Quantum computing. *Reports on Progress in Physics* 61, 2 (1998), 117.

[116] Thomas Sterling, Matthew Anderson, and Maciej Brodowicz. 2017. A Survey: Runtime Software Systems for High Performance Computing. *Supercomput. Front. Innov.: Int. J.* 4, 1 (March 2017), 48–68. https://doi.org/10.14529/jsfi170103

[117] Jill Tomley and Mary Searcy. 2009. Computational Science: Not Just for Researchers Anymore. *The International Journal of Science in Society* 1, 3 (2009), 27–42.

[118] Top500. 2017. TOP500 Meanderings: Supercomputers for Weather Forecasting Have Come a Long Way. https://www.top500.org/news/top500-meanderings-supercomputers-for-weather-forecasting-have-come-a-long-way/.

[119] Top500. 2017. Walt Disney Feature Animation. https://www.top500.org/site/49252.

[120] TOP500 list. 2020. http://www.top500.org.

[121] TOP500.org. 2020. Top500 Performance Development. https://www.top500.org/statistics/perfdevel/

[122] Heikki Topi, Helena Karsten, Sue A. Brown, João Alvaro Carvalho, Brian Donnellan, Jun Shen, Bernard C. Y. Tan, and Mark F. Thouin. 2017. *MSIS 2016: Global Competency Model for Graduate Degree Programs in Information Systems*. Technical Report. Association for Computing Machinery, New York, NY, USA.

[123] US National Oceanic and Atmospheric Administration. 2015. High Performance Computing Strategic Plan, 2015-2020 (Final Draft). https://www.noaa.gov/sites/default/files/atoms/files/HPCStrategy_Final_Draft_080913.pdf.

[124] Michael L. Van De Vanter, DE Post, and Mary E. Zosel. 2005. HPC needs a tool strategy. In *Proceedings of the second international workshop on Software engineering for high performance computing system applications*. ACM, St. Louis, Missouri, 55–59.

[125] Les Waguespack, Heikki Topi, Stephen Frezza, Jeffry Babb, Linda Marshall, Shingo Takada, Gerrit Veer, and Arnold Pears. 2019. Adopting Competency Mindful of Professionalism in Baccalaureate Computing Curricula. In *2019 Proceedings of the EDSIG Conference*. ISCAP (Information Systems and Academic Professionals), Cleveland Ohio, 17.

[126] S. Wienke, J. Miller, M. Schulz, and M. S. Müller. 2016. Development Effort Estimation in HPC. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, Salt Lake City, Utah, 107–118.

[127] XSEDE Project. 2020. XSEDE Educator Programs. https://www.xsede.org/community-engagement/educator-programs.

[128] Xuejun Yang, Xiangke Liao, Kai Lu, Qingfeng Hu, Junqiang Song, and Jinshu Su. 2011. The TianHe-1A Supercomputer: Its Hardware and Software. *Journal of Computer Science and Technology* 26 (2011), 344–351.

[129] Gangman Yi and Vincenzo Loia. 2019. High-performance computing systems and applications for AI. *The Journal of Supercomputing* 75, 8 (2019), 4248–4251.

[130] Il-Chul Yoon, Alan Sussman, and Adam Porter. 2005. And Away We Go: Understanding the Complexity of Launching Complex HPC Applications. In *Proceedings of the Second International Workshop on Software Engineering for High Performance Computing System Applications (SE-HPCS '05)*. Association for Computing Machinery, New York, NY, USA, 45–49. https://doi.org/10.1145/1145319.1145333

A ELEMENTS FROM CC2020

The CC2020 draft report [13] specifies a better practice for listing the needed knowledge, skills, and dispositions to create needed curricular competencies. It provides example high-level vocabulary for knowledge, skills, and competencies based on wisdom elicited from the computing community, as shown in Tables A.1, A.2, A.3 and A.4. Most importantly, the dispositions present themselves as a breakthrough contribution to the definition of needed competencies, and are very much related to activities in the workplace and academia.

Table A.1: Sample Knowledge Elements from CC2020 [13]

Humans and Organizations	Systems Modeling	Software Systems Architecture	Software Development	Software Fundamentals	Hardware
Social Issues User Experience Security Policy IS Management Enterprise Architecture Project Management	Security Issues Systems Analysis Requirements Analysis Data Management	Virtual Systems Embedded Systems Integrated Systems Intelligent Systems Internet of Things Computer Networks Platform Technologies Parallel Computing Security Technology	Software Quality Software Verification Software Process Software Design Software Modeling Platform Development Software Development	Graphics and Visualization Operating Systems Algorithms Programming Languages Software Development Systems Fundamentals	Architecture and Organization Digital Design Circuits / Electronics Signal Processing

Table A.2: Sample Professional and Foundational Skills' Elements from CC2020 [13]

Analytical and Critical Thinking Collaboration and Teamwork Ethical and Intercultural Perspectives Mathematics and Statistics Multi-Task Prioritization and Management Oral Communication and Presentation Problem Solving and Trouble Shooting	Project and Task Organization and Planning Quality Assurance / Control Relationship Management Research and Self-Starter/Learner Time Management Written Communication

Table A.3: Sample Levels of Cognitive Skills' Elements from CC2020 [13]

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Exhibit memory of previously learned materials by recalling facts, terms, basic concepts and answers	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions	Solve problems to new situations by applying acquired knowledge, facts, techniques, and rules in a different way	Examine and break information into parts by identifying motives or causes; make inferences and find evidence to support solutions	Present and defend opinions by making judgements about information, validity of ideas, or quality of material	Compile information together in a different way by combining elements in a new pattern or proposing alternative solutions

Table A.4: Sample Disposition Elements from CC2020 [13]

Element	Elaboration	Element	Elaboration
Proactive	With initiative, self-starter, independent	Adaptable:	Flexible; agile, adjust in response to change
Self-directed	Self-motivated, determination, independent	Collaborative:	Team player, willing to work with others
Passionate	Conviction, strong commitment, compelling	Responsive:	Respectful; react quickly and positively
Purpose-driven	Goal driven, achieve goals, business acumen	Meticulous	Attentive to detail; thoroughness, accurate
Professional	Professionalism, discretion, ethical, astute	Inventive:	Exploratory. Look beyond simple solutions
Responsible	Use judgement, discretion, act appropriately		

B HPC EDUCATION RESOURCES

This appendix includes three tables with relevant HPCEd resources for educators as follows:

- Table B.1 contains a list of upper undergraduate and postgraduate courses at different institutions worldwide.
- Table B.2 provides a list of textbooks useful for HPCEd.
- Table B.3 lists a sample set of papers that illustrate ideas and experiences integrating HPC into the undergraduate curriculum. The list of papers is not meant to be exhaustive or complete, merely representative of different HPC approaches.

Table B.1: HPC University courses and other graduate modules

Institution	Description and web link
ANU, Australia	Australian National University offers a course on High Performance Scientific Computing (COMP6464) that provides an introduction to High Performance Computing with an orientation towards applications in science and engineering. https://programsandcourses.anu.edu.au/course/comp6464
Barcelona Supercomputing Centre	Offers a MSc in High Performance Computing. https://www.bsc.es/education/graduate/master-programme
Ben-Gurion University	Introduction to parallel processing - undergraduate course materials licensed under a Creative Commons http://tel-zur.net/teaching/bgu/pp/schedule.html
Carnegie Mellon University	Course 15-418/15-618 contains lectures and exercises on Parallel Computer Architecture and Programming topics. http://www.cs.cmu.edu/~418/
EPCC (formerly the Edinburgh Parallel Computing Centre)	offers HPC training as well as MSc and PhD programmes in HPC. https://www.epcc.ed.ac.uk/education-training
Intel partnerships	Intel has partnered with a number of US universities to established parallel computing centers, lecture material, and hands-on lab modules for undergraduate and graduate courses. George Washington University: https://ipcc.seas.gwu.edu/lectures/ Georgia Tech University: https://www.cc.gatech.edu/~echow/ipcc/hpc-course/ University of Oregon: http://ipcc.cs.uoregon.edu/curriculum.html
Irish Centre for High-End Computing	Offers graduate modules on Scientific Programming Concepts and HPC & Parallel Programming https://www.ichec.ie/about/activities/academic-user-support
NUDT, China	China's National University of Defense Technology offers the HPC series of courses that provide an introduction to high-performance computing, parallel programming methodology, and parallel computing system for computing majors and non-computing majors. https://www.educoder.net/hpc-course/
PRACE (Partnership for Advanced Computing in Europe)	offers approximately 100 training events each year in collaboration with training centres in cooperation with 14 national HPC centres in Europe. They also run summer schools for students and postdoctoral researchers and offer online training. https://prace-ri.eu/training-support/training/
Trinity College, Dublin, Ireland	Offers an MSc in High Performance Computing. https://www.tcd.ie/courses/postgraduate/az/course.php?id=DPTMA-HPCO-1F09

Table B.2: A Selection of HPC Textbooks

Year	Textbook	Description
1994	**Introduction to parallel computing.** Kumar, Vipin, Ananth Grama, Anshul Gupta, and George Karypis. Vol. 110. Redwood City, CA: Benjamin/Cummings.	Introduction to Parallel Computing is a complete end-to-end source of information on parallel computing from introduction to architectures to programming paradigms to algorithms to programming standards. It covers traditional Computer Science algorithms (sorting, graph and matrix algorithms), scientific computing algorithms (FFT, sparse matrix computations, N-body methods), and data intensive algorithms (search, dynamic programming, data-mining). https://www.oreilly.com/library/view/introduction-to-parallel/0201648652/
2009	**High performance heterogeneous computing.** Dongarra, Jack, and Alexey L. Lastovetsky. Vol. 78. John Wiley & Sons	This book provides an overview of the ongoing academic research, development, and uses of heterogeneous parallel and distributed computing in the context of scientific computing. The book is organized in five distinct parts: (1) Heterogeneous Platforms: Taxonomy, Typical Uses, and Programming Issues (2) Performance Models of Heterogeneous Platforms and Design of Heterogeneous Algorithms (3) Performance: Implementation and Software (4) Applications (5) Future Trends https://www.wiley.com/en-us/High+Performance+Heterogeneous+Computing-p-9780470508190
2008	**Parallel computing on heterogeneous networks.** Lastovetsky, Alexey L. Vol. 24. John Wiley & Sons.	Provides a detailed introduction to parallel computing on heterogenous clusters, including new parallel computing approaches that make better use of the heterogeneous cluster architecture All concepts and algorithms are illustrated with working programs that can be compiled and executed on any cluster https://onlinelibrary.wiley.com/doi/book/10.1002/0471654167
2011	**An introduction to parallel programming. 1st edition** Pacheco, Peter. Elsevier.	Pacheco uses a tutorial approach to show students how to develop effective parallel programs with MPI, PThreads, and OpenMP. The first undergraduate text to directly address compiling and running parallel programs on the new multi-core and cluster architectures. User-friendly exercises teach students how to compile, run and modify example programs. https://www.elsevier.com/books/an-introduction-to-parallel-programming/pacheco/978-0-12-374260-5
2013	**Introduction to High Performance Scientific Computing.** Eijkhout, Victor. Lulu.com	Free. https://pages.tacc.utexas.edu/~eijkhout/istc/istc.html?source=techstories.org
2014	**Programming on Parallel Machines; GPU, Multicore, Clusters and More.** Matloff, Norm.	Free. http://heather.cs.ucdavis.edu/~matloff/158/PLN/ParProcBook.pdf
2017	**Parallel Programming in MPI and OpenMP.** Eijkhout, Victor. Lulu.com.	Free. www.tacc.utexas.edu/~eijkhout/istc/istc.html
2017	**Elements of Parallel Computing** Aubanel, Eric. Chapman and Hall/CRC.	Designed for introductory parallel computing courses , Elements of Parallel Computing presents the fundamental concepts of parallel computing not from the point of view of hardware, but from a more abstract view of algorithmic and implementation patterns. The first five chapters present core concepts in parallel computing. The second part presents three case studies that reinforce the concepts of the earlier chapters. The content of the book is language neutral, using pseudocode that represents common programming language models.
2020	**An introduction to parallel programming. 2nd edition** Pacheco, Peter and Malensek, Matthew. Elsevier.	The new edition includes coverage of accelerators via new content on GPU programming and heterogeneous programming. https://www.elsevier.com/books/an-introduction-to-parallel-programming/pacheco/978-0-12-804605-0

Table B.3: Papers of interest to introduce HPC at college at high school level

Year	Paper	Description
2007	Joseph Smith and Greg Wolffe. *Introducing AP computer science students to high-performance computing.* J. Comput. Sci. Coll. 23, 1 (October 2007), 70–76.	**Topic**: Early introduction to high performance. K-12 / University partnership to co-develop a 3-week module on HPC that was presented to students in an Advanced Placement computer science class.
2008	Homma Farian, Kirk M. Anne, and Matthew Haas.*Teaching high-performance computing in the undergraduate college CS curriculum.* J. Comput. Sci. Coll. 23, 3 (January 2008), 135–142.	**Topic**: Hands-on cluster projects. This paper illustrate how to move from a theorical coverage of distributed systems to a hands-on course in which students are given the opportunity to pull together various aspects of their Computer Science education including Operating Systems, Graphics, Data Structures, Data Communications, and Networking to explore the realm of HPC.
2012	Andrew J. Pounds. *The babyblas - an extended project for introducing undergraduates to the concepts of high performance and parallel scientific computing.* J. Comput. Sci. Coll. 28, 2 (December 2012), 153–159.	**Topic**: hands-on HPC by building a high-performance library. Introduced advanced Computational Science undergraduates to the principles of high performance computing (HPC) and parallel scientific computing. In the project students build their own miniature version of the Basic Linear Algebra Subroutines (BLAS).
2013	Omar Abuzaghleh, Kathleen Goldschmidt, Yasser Elleithy, and Jeongkyu Lee. *Implementing an affordable high-performance computing for teaching-oriented computer science curriculum.* ACM Trans. Comput. Educ. vol 13 no 1, 14 pages. DOI:https://doi.org/10.1145/2414446.2414449	**Topic**: affordable high-performance cluster system This work designed and implemented an affordable HP cluster system based on the PlayStation 3 (PS3) connected through gigabit ethernet. To evaluate this PS3 cluster, they conducted a benchmarking test, that is, matrix multiplication, with different numbers of synergistic processing elements (SPEs) and nodes. The implemented PS3Cluster syste mwas used for computer science courses, such as Parallel and Distributed Databases and Parallel Programming.
2014	A. Shafi, A. Akhtar, A. Javed and B. Carpenter, *Teaching Parallel Programming Using Java.* 2014 Workshop on Education for High Performance Computing, New Orleans, LA, 2014, pp. 56-63, DOI: 10.1109/EduHPC.2014.7.	**Topic**: Java support for parallel programming This paper illustrates how to use Java threads API to teach parallel programming techniques for shared memory systems, and MPJ Express – a Java MPI library – to cover distributed memory systems including clusters and network of computers. Course weekly schedule and sample assignments are provided.
2014	D. Valentine, *HPC/PDC Immunization in the Introductory Computer Science Sequence.* 2014 Workshop on Education for High Performance Computing, New Orleans, LA, 2014, pp. 9-14, DOI: 10.1109/EduHPC.2014.11.	**Topic**: early exposure to PDC/HPC topic This paper advocates for exposing students to PDC/HPC topics within a "traditional" CS1/CS2 environment, so that students are aware of the existence of parallelism, to facilitate a natural transition from serial to parallel.
2015	Maha Aziz, Heng Chi, Anant Tibrewal, Max Grossman, and Vivek Sarkar. 2015. *Auto-grading for parallel programs.* Workshop on Education for High-Performance Computing (EduHPC 2015). ACM, New York, pp. 1–8. DOI:https://doi.org/10.1145/2831425.2831427	**Topic**: Grading tools for parallel programming tasks This paper describes our work on extending Web-CAT to address the requirements of Rice University's introductory parallel programming course, thereby creating infrastructure that can be used for similar courses at other universities and in online courses.
2018	Ben Glick and Jens Mache. *Using jupyter notebooks to learn high-performance computing.* J. Comput. Sci. Coll. 34, 1 (October 2018), 180–188.	**Topic**: Jupyter notebooks support for HP This paper describe a set of 8 Jupyter chapter that cover: "What is HPC?", "Resource Managers and Schedulers," "Parallel Algorithms," "Workflow Management Systems and Tools," "Distributed Algorithms," "Parallel Algorithm Analysis," and "Benchmarking. Hands-on programming is supported by the python-based Parallel Scripting Library (Parsl)