High-Performance Computing and Engineering Educational Development and Practice

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

John Impagliazzo
School of Engineering and Applied Science
Hofstra University
Hempstead, New York, USA
john.impagliazzo@hofstra.edu

Li Shen
College of Computer
National University of Defense Technology
Changsha, China
lishen@nudt.edu.cn

***Abstract*—This Innovate Practice Full Paper presents HPC and engineering educational development and practice. Educators and researchers are witnessing the emergence of parallel and high-performance computing (HPC) in many computing and engineering environments worldwide. Single-core processing is slowly becoming obsolete while parallel solutions to application development are now replacing serial solutions to achieve higher performance, particularly in many scientific, engineering, and computing fields. Some universities or research institutes worldwide have developed and have become centers of supercomputing. Most of these centers, particularly the world acclaimed Tianhe-2 supercomputer developed by China's National University of Defense Technology (NUDT), often have their own HPC educational approaches as well as their own curricula at both undergraduate and graduate levels. This paper presents some high-level HPC educational concepts, educational framework designs, and strategies for cultivating HPC talented graduates. In this work, the authors show unique perspectives and practical experiences on student capability, oriented toward HPC curricular development. They also show a step-by-step practical system in which real platform, real application, and research-teaching integration modes can immerse students into an HPC world for better understanding. The authors show how researchers with rich computing and engineering experience can become deeply involved in course design, in-class teaching, and in laboratory instruction. They also discuss the effectiveness of HPC courses at the NUDT and they show how HPC can become a stable element in computing and engineering education.**

Keywords—high-performance computing, HPC curricular systems, HPC education, step-by-step practice systems, educational modes, HPC computing and engineering education

I. INTRODUCTION

Parallel and high-performance computing (HPC) are emerging in many computing and engineering environments worldwide. Currently, single-core processing is slowly becoming obsolete. Parallel solutions to application development are now replacing serial solutions to achieve higher performance, particularly in many scientific, engineering, and computing fields.

Some universities worldwide have developed and have become centers of supercomputing. Of interest is the accomplishment of the Tianhe-2 supercomputer developed by the National University of Defense Technology (NUDT) in China. Many centers for supercomputing often have their own high-performance computing curricula at undergraduate or graduate levels. These centers promote broad involvement in parallel and supercomputing; some even involve non-computing and non-engineering majors as exemplified at NUDT. Advancements in academia surrounding supercomputing research can form a strong basis for a robust curriculum and new possibilities for university students worldwide.

From its inception, high-performance computing has been a very interdisciplinary academic and research area. Researchers, faculty members, and students now use supercomputers to acquire new knowledge, to publish their research results, and to explore nascent areas in a variety of scientific, computing, and engineering fields. In some areas, research would stall or even come to a halt without HPC knowledge. Therefore, it is imperative that HPC teaching and learning start early and that it becomes a distinct computing and engineering element in the education profile of students.

However, high-performance computing technology develops very quickly. The gap between HPC technology development and HPC education is growing larger than before. Technical advancements outpace educational change and the "catch-up mode" becomes continuously challenging. Questions such as: How should educators respond to the latest parallel and distributed computing (PDC) and HPC research achievements? How do educators infuse this rapidly changing knowledge into PDC/HPC classes to make the computing and engineering undergraduate curricula prepare its students to serve better this evolving and developing field? This paper addresses these and other questions related to HPC education.

II. OBSTACLES TO HPC EDUCATION

Many obstacles exist regarding curricular issues, practical environments, and teaching modes for cultivating competent HPC students. An explanation of these challenges now follows.

A. HPC Curricula and Education Needs

Current HPC curricular systems cannot meet the educational needs to produce strong HPC graduates. High-performance computing has the characteristics of high system complexity, wide application fields, and comprehensiveness. There is an urgent need to produce many diverse and talented HPC professionals across multiple fields such as computers, applications, and system maintenance. However, the knowledge structures of these three fields are usually independent. Current methods of learning should break from this situation of having independent knowledge systems. Universities should develop stronger HPC curricular frameworks to accommodate the needs of different-types and different-levels of proficient HPC graduates and professionals.

B. Practical Environments and Educational Processes

The current educational environment has difficulty in supporting high-performance computing education processes. Students have difficulty in accessing real-world platforms and application cases; additionally, they lack practical training. The computing field needs to establish practical HPC environments, including different scales of parallel computing platforms, collections of practical exercises and experiments with multiple levels of difficulty, and step-by-step practical teaching modes.

Current teaching modes cannot achieve the educational goals of producing high-performance computing graduates. Existing teaching methods and evaluation strategies make it difficult to achieve the goal of cultivating innovative HPC graduates. In the face of high-complexity practical systems and wide-ranging application problems, even experienced teachers can hardly complete the tracking of the entire process of practical guidance and evaluation. The educational system needs the participation of experienced lecturers and teachers, especially HPC researchers for each stage of student study and practice in and out of classroom settings.

C. Addressing the Obstacles

In this work, the authors provide some unique perspectives, practice experiences, teaching effects, lessons on curricular development, practical environment setup and teaching mode innovation. They will also explain solutions and their effects through several core HPC courses. In particular, the authors show how HPC scientific research integrates with the HPC teaching process. For example, they show ways to customize real application problems for appropriate teaching situations in lectures and projects, and in laboratory classes.

The authors also show how researchers with rich computing and engineering experience have become involved in course design, in-class teaching, and in laboratory instruction. The innovation of complex HPC education contributes toward new teaching ideas, practical teaching case studies from current real HPC research problems, and distinctive laboratory instruction methods within a university's HPC computing and engineering educational environment. A discussion surrounding the effectiveness of HPC courses for an undergraduate curriculum can help show how universities might participate in this relatively novel experience.

III. Related Work

Many modern scientific developments depend on large-scale data processing and the exploitation of parallelism on supercomputer systems. Examples include computational simulations for scientific and engineering applications in the atmosphere, earth, and environmental realms [1], in addition to commercial applications such as data mining and transaction processing.

A. Perspectives on Parallel and HPC Education

Since the 1990s, people have explored including high-performance computing in the undergraduate computing curriculum. For example, Donald Johnson et al. [2] proposed developing HPC elements into a data structures course. Additionally, the Curriculum Development and Education Resources (CDER) center [3] has established curricular guidelines for early computing courses [4]. Notwithstanding, implementing HPC early in the computing curriculum remains a continuing challenge.

There has been an ongoing interest to include HPC elements in computing curricula. For example, the CS2013 curricula guidelines indicated that "the vastly increased importance of parallel and distributed computing, it seemed crucial to promote those topics to the core" [5, p 29]. Consequently, CS2013 proposed a new knowledge area in parallel and distributed computing [6]. As another example, the 2016 ACM/IEEE joint computer engineering curricula report (CE2016) has included four knowledge units titled multi/many-core architectures, distributed system architectures, system architectural design and evaluation, and concurrent hardware and software design [7] as part of its recommendations.

B. Educational Advances at NUDT

Many universities have focused on establishing centers of high-performance computing. Specifically, NUDT has developed and has become a world center for the Tianhe-2 supercomputer. As a global center for supercomputing, the university developed its own high-performance undergraduate curriculum. It promotes broad involvement in parallel and supercomputing, even to non-computing and non-engineering majors. Its advancements in academic research surrounding supercomputing form a strong basis for a robust curriculum offering new possibilities for its own students and for students throughout China and beyond. The university recognizes and acknowledges that high-performance computing education has been a challenge because HPC is a very interdisciplinary academic and research area.

IV. Basic Issues

With respect to the challenges mentioned, this paper focuses on three challenging issues, presented as the following three queries identified as Q1, Q2, and Q3.

A. The Problem of Relatively Independent Knowledge Systems in Various Fields – Q1

As explained in Section II, high-performance and parallel computing have the characteristics of high system complexity, wide application fields, and comprehensiveness and the field urgently needs many-faceted talented graduates that can span multiple fields such as computers, applications, and system maintenance. The cultivation of each type of talented graduates is often relatively independent. The situation leads to the problem that application domain experts lack HPC expertise; likewise, HPC experts lack application domain knowledge, while system maintenance personnel lack both application domain and computing knowledge.

The situation as stated above hinders the development of meaningful high-performance computing applications. The current situation that exists at universities presents a significant challenge to educational resources and curriculum development. Therefore, it is important to break the barriers or walls between different independent knowledge fields and to form a uniform HPC curricular system that satisfies the educational needs of compound talented graduates that span multiple application fields.

B. The Practice Environment Lags Behind the Need for HPC Education – Q2

Students have difficulty accessing real practice platforms and application cases; they also lack practical exercises. No matter whether it is a simulated parallel environment or a small-scale parallel computing system, students cannot reproduce the unique problems of large-scale parallel systems such as parallel scalability, large-scale fault tolerance, power management, and dynamic system maintenance.

The lack of real application cases for practical systems makes it difficult to educate and train students to understand the practical problems of high-performance computing applications. In addition, teachers cannot directly use real application cases as teaching content. Complex problems contained in the parallel and high-performance domains require decomposition into smaller and more detailed forms so students can understand the problem while faculty members can include them as part of their teaching portfolio.

C. The Problem of Inefficient Teaching Modes – Q3

Existing teaching and evaluation methods are also difficult to achieve. Active guidance to cultivate innovative graduates for high-performance computing applications should be a priority. In the face of high-complexity practical systems and wide-ranging application problems, even experienced teachers have difficulty completing the entire process of practical guidance and evaluation.

The situation begs the question of the necessity for experienced teachers and researchers to participate in this evolving HPC education field. The situation presents both a challenge and an opportunity for all educators working on HPC problems and HPC educational courseware. Improvement of HPC educational processes, tools, and curricula should be a goal for all educators interested in the HPC domains.

V. METHODOLOGY BY LAYER MODELING

Before presenting solutions to the above three problems, it is important to explain the principles of layer modeling and to show ways to connect hierarchical capability learning to course design useful for solving the three problems.

A. Parallel Computational Thinking

Hierarchical approaches of learning have been in commonly use. In 1956, Robert M. Gagné [8] proposed a hierarchy system of learning in terms of the degree of complexity of the mental processes involved. He ordered the eight basic types of learning he identified with an increasing complexity from the lower to the upper. In this way, behavioral and cognitive aspects are distinguishable and easier to learn. In this work, we use the hierarchical approach to make capacity development more specific.

Teachers can utilize parallel computational thinking as a goal of cultivating student capacities and use capacity learning to help curricular system design. They clarify the content of capacity development by using hierarchical methods. Teachers decompose parallel computing problems into four capability categories: model abstraction, parallel algorithm design, parallel programming, and parallelism evaluation and optimization. Teachers deliver such multi-layer capacity demand to their students in various courses [9].

The development of a multi-level framework for parallel computational thinking contains four layers for category capability. The first or top layer identifies with the ability to do problem decomposition and abstraction by transforming a real interdisciplinary problem into an abstract model. The second layer addresses the ability to do parallel algorithm design by a transformation from models to parallel algorithms. The third layer addresses the ability to do parallel programming capabilities through a transformation from parallel algorithms to parallel program design. The fourth or bottom layer identifies parallelism evaluation and optimization capability testing for computing efficiency, scalability, communication performance, and other elements followed by optimizing application parallelism.

B. Connecting Hierarchical Capability Learning to Course Design

Fig. 1 shows the relationship between hierarchical parallel computational thinking capability and interdisciplinary knowledge. The higher the level, the greater the requirement of a deeper understanding of specific issues and domain knowledge. Here, the focus is to transfer a continuous real-world problem into a discrete computer-solvable specific problem. The higher the level, the greater the requirement for abstraction ability; that is, it must be closer to the actual problem analysis and decomposition. The lower the level, the greater the requirement for hands-on computer practical ability; that is, the higher complexity involved, the greater is the requirement for comprehensive ability and closeness to computing knowledge. From a hierarchical perspective, domain-specific knowledge and computing knowledge are not independent entities; they partially overlap with each other. This hierarchical capability framework makes the dividing line and the common point of two knowledge domains clearer.

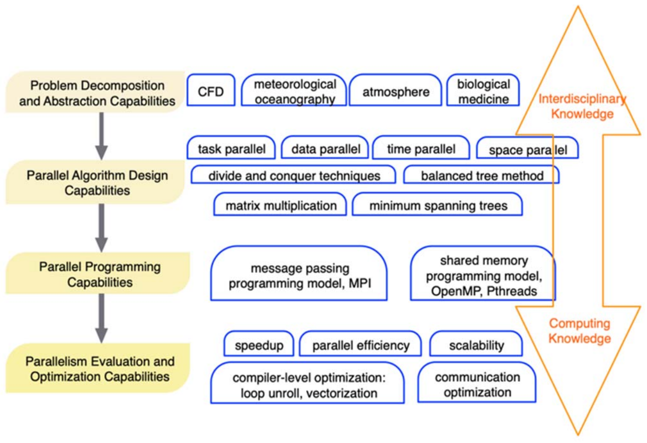
Problem Decomposition and Abstraction Capabilities
CFD
meteorological oceanography
atmosphere
biological medicine
Interdisciplinary Knowledge
task parallel
data parallel
time parallel
space parallel
Parallel Algorithm Design Capabilities
divide and conquer techniques
balanced tree method
matrix multiplication
minimum spanning trees
Parallel Programming Capabilities
message passing programming model, MPI
shared memory programming model, OpenMP, Pthreads
Computing Knowledge
speedup
parallel efficiency
scalability
Parallelism Evaluation and Optimization Capabilities
compiler-level optimization: loop unroll, vectorization
communication optimization

Fig 1. Relationship between hierarchical parallel computational thinking capability and interdisciplinary knowledge.

At the NUDT, the university has five core undergraduate courses for parallel computing and high-performance computing spanning first-year through fourth-year students. The breadth and depth increase incrementally. Through the five courses, it is possible to show how to develop a parallel computational thinking capacity for students. Fig. 2 shows how the five courses cover different layer capabilities where different depths associate with separate levels. On the average, 24 undergraduate students enrolled in the curricular courses. Of these, on the average 23 students completed the sequence. Upon completion, 11 graduates went to industry and 12 graduates continued in academia for a master's degree in HPC or a closely related discipline.

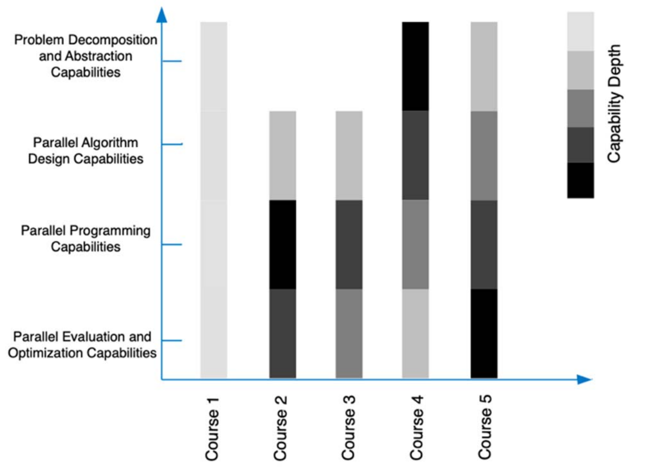

Fig 2. Five core HPC undergraduate courses with the goal of cultivating parallel computational thinking capabilities of students.

C. *Course 1: Introduction to the Supercomputer System - Discover High-performance Computing*

This first-year elective course aims to present a survey of high-performance computing history, its current status and future, using the Tianhe-2 supercomputer as an example. Teachers connect nearly all knowledge content with their own supercomputer experience, including parallel computer architecture, parallel programming model, parallel compiler technology, high-performance system interconnect technology, parallel storage system, and large-scale parallel applications. For example, the lecture of parallel compiler technology especially focuses on the machine's compiler tool development and optimization techniques, especially their experiences or interesting stories.

Instructors do not expect first-year students to understand all details and techniques. Instead, they hope students create a deep impression on the development process and experiences from developers. Instructors invite researchers who have rich development experiences to engage in student discussions. In this way, these students will learn more about the Tianhe-2 supercomputer and develop great pride of the university and become inspired wish to do HPC research work later. Leading first-year students into the HPC world and sparking their interests in HPC knowledge are the main goals of this course. Course-1 is the first of a series of high-performance computing courses, which includes sixteen class sessions.

D. *Course 2: Introduction to Parallel Programming*

This second-year required course is the first programming course in the high-performance computing course series. Students begin learning parallel computer systems from the perspective of parallel programming. By practicing the writing of parallel programs, students will further understand basic parallel computer architecture, parallel programming models, mainstream parallel programming interfaces and mechanisms, and several common parallel algorithm cases. Students must do many parallel programming homework lessons and projects by using message-passing interface (MPI), OpenMP and Pthreads. The proportion of lecture lessons and laboratory lessons is about 1:1. Projects are motivated not only by the lecturer, but also by several other professionals from the HPC research groups.

In the parallel programming course, the researchers with rich high-performance computing engineering experiences mainly involved in laboratory instruction and after-class project instruction. Three to five students form a group and receive instruction by a researcher or a doctoral student. The researchers (experimental teachers) identify the subject of the project and design the detailed title and requirements with the lecture teacher before assigning the projects. Researchers check the progress of each group. They evaluate the performance of each student and they also assign new tasks for the following week. At the mid-term and final-term of the course, all the researchers participate in all group presentations and evaluate their performances. The grades of each student's performance on each week feedback, mid-term presentation, and final-term presentation constitute their project grades. Earlier work [10] describes more details on ways to develop a parallel programming course.

E. *Course 3: Parallel Computing Systems I*

This third-year elective course for computing majors focuses on developing students' problem-solving abilities by parallel computational thinking. The main contents include parallel computer architecture, high-performance computing interconnection networks, basic concepts and principles of the parallel programming model, coding by using MPI and OpenMP, evaluation of scalability, and classic parallel application solutions. The content of this course has some overlap with the "Introduction to Parallel Programming" course. Specifically, the "Parallel Computing Systems I" course provides greater emphasis on parallel computer architecture, and parallel application evaluation and optimization, which covers more Level 2, Level 4 of the parallel computational thinking multi-layer structure, and partially covers Level 3. While the "Introduction to Parallel Programming" course focuses on parallel programming capability (Level 3).

Although this is an elective course for computing majors, most third-year students choose this course because they possibly need to do more practical applications in the future. The topics of projects cover a wide range. As an example, the "Parallel Computing Systems I" course typically includes eight projects covering the following four aspects: on-chip network design, high-performance interconnection network scheduling algorithm design and performance evaluation, parallel file system and job scheduling policy, and computational fluid dynamics (CFD) parallel application performance, evaluation and optimization.

F. *Course 4: Parallel Computing Systems II*

The design of this third-year elective course for non-computing majors has a reduction in depth of course content with customized applications according to the specialization of the students. Students who lack a background in fundamental computer architecture, systems and algorithms have some difficulties in understanding parallelism, scalability and other issues. Faculty members developed a special project design for third-year students of the College of Meteorological Oceanography. All the projects come from the student's major domains such as weather research and forecasting model. Instructors seek assistance from the professor and teaching assistants from this college. They work together to design the project, check the progress of projects, and instruct students in laboratory lessons.

G. Course 5: High-Performance Computing

This required fourth-year course aims to show the basic principles, the key challenges, and the latest research results in the HPC field. It is an enhanced version of the first-year course. The course presents many challenges and several instructors teach it. The course contains four principal topics: (a) high-performance microprocessor design and parallel computer architecture, (b) high-performance interconnection network design and optimization, (c) high-performance system software and optimization (including compiler optimization), and (d) large-scale parallel application development and evaluation.

VI. Solving Query Q1

The basic idea toward a solution of Q1 is to restructure and optimize the HPC curricular system with the goal of cultivating parallel computational thinking capabilities for students. The following discussion aims to explain ways to build an HPC curriculum using hierarchical capability modeling. The hierarchical structure helps to decompose abilities into smaller pieces. In this way, it is possible to reconstruct diminished capacities to accommodate demands for a course capability.

Hierarchical capability modeling seeks to transcend barriers between different independent knowledge fields and to form a uniform HPC curriculum that satisfies the educational needs to produce talented graduates who can span multiple application fields. The main steps are as follows.

Firstly, in the solution of complex engineering problems, the goal is to cultivate a problem-solving ability. It is necessary to clarify the content of each layer of the hierarchical parallel computational thinking ability and determine key points and difficulties for each layer. Secondly, it is crucial to clarify a relationship between layers in the layered model to reconstruct capability key points and knowledge contents for satisfying the learning needs of each field. Thirdly, according to the content of each field's knowledge, it is feasible to develop a series of customized capability-centered courses where 'customized' means to choose appropriate characteristics on each layer based on decided capability points. Fig. 3 shows a framework for the series of HPC courses covering from undergraduate to graduate studies at NUDT.

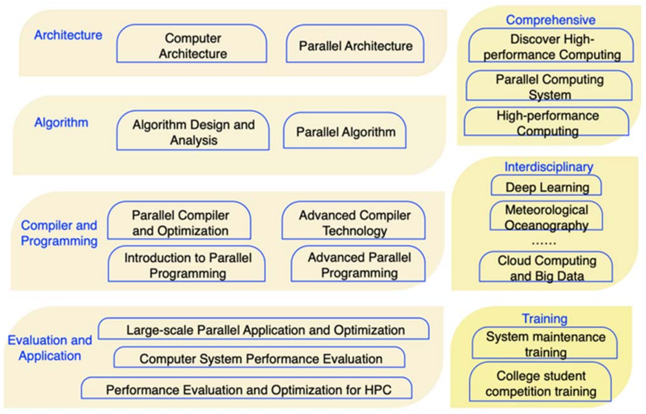
Architecture
Computer Architecture
Parallel Architecture
Algorithm
Algorithm Design and Analysis
Parallel Algorithm
Compiler and Programming
Parallel Compiler and Optimization
Advanced Compiler Technology
Introduction to Parallel Programming
Advanced Parallel Programming
Evaluation and Application
Large-scale Parallel Application and Optimization
Computer System Performance Evaluation
Performance Evaluation and Optimization for HPC
Comprehensive
Discover High-performance Computing
Parallel Computing System
High-performance Computing
Interdisciplinary
Deep Learning
Meteorological Oceanography
......
Cloud Computing and Big Data
Training
System maintenance training
College student competition training

Fig 3. HPC course series framework showing how courses cover interdisciplinary computing fields.

In Fig. 3, teachers restructure the contents of most courses by chosen characteristics; they also add a large proportion of practice exercises or projects. Students majoring in computing fields prefer four-layer courses at the left and comprehensive courses at the top right. Students majoring in interdisciplinary fields would choose some professional courses from the 'Interdisciplinary' box and some left lower-layer computing courses. Industry professionals can customize some training courses to learn more system management and operation skills and as well as maintenance techniques. These "training" courses are not suitable for college students and are more appropriate for faculty members or supercomputing center staff personnel. College students can achieve comprehensive system capability improvement through various competitions, special challenging courses, or other related activities. Fig. 3 does not cover all the possible courses. Different colleges or universities may have their own similar courses that cannot have a full presentation here. Since the HPC curriculum is very fluid and dynamic, it continuously evolves and improves.

VII. Solving Query Q2

Basic idea of a solution to Q2 is reliance on high-performance computing innovation practice based on an available supercomputing center. It is possible to build a real large-scale parallel computing environment that serves the teaching process. An HPC research team can provide real high-performance application problems to create step-by-step practice modes for HPC education.

A. Real Practice Platform and Real Teaching Cases

At NUDT, the Tianhe-2 supercomputer platform, as a real parallel computing environment, serves as a vehicle for practice exercises. At the same time, faculty members integrate existing resources to build an innovative HPC practice base shared by the entire campus with a peak performance of 100 teraflops. A scientific research team provides real application problems from typical high-performance computing fields, to form a series of comprehensive practical projects covering the computing and domain-oriented fields, which run through the entire curriculum practice system. On this basis, faculty members build a library of typical case study resources.

B. Step-by-step Practical Modes

According to the knowledge level of students and their gradual improvement in their ability, teachers can establish a step-by-step practical mode. This practical mode includes step-by-step parallel computing platforms and teaching practice exercises or projects. The incremental approach of the platform begins from a single multi-core computing node to a medium-scale parallel multi-node (such as with a high-performance computing innovation practice base cluster), and finally to a large-scale parallel multi-node machine such as the Tianhe-2 supercomputer. This step-by-step method of teaching practice exercises and projects first begins with basic verification experiments on classic problems of parallel programming, then it moves to medium-scale tests based on typical cases and benchmarks, and finally moves to large-scale or comprehensive typical application problem-solving.

C. Examples of Some Case Studies

Appendix A shows a partial snapshot for step-by-step teaching practice exercises or projects. For small-scale, medium-scale and large-scale processes, the number of computer nodes is about 1-8, 8-32, and 32-128, respectively.

For a parallel system with a 24-core per node, the number of processor cores is up to about 3,000 (6,000 with hyperthreading). Although 6,000 processor cores are far smaller than large-scale application demands in a real HPC world, this scale is enough for immersing students into an HPC world and is beneficial for their better understanding to some key problems such as scalability and parallel efficiency.

Faculty members can partition various problem types not only by the scales of used computer nodes, but also by the complexity of problems. The higher the layer, the more singular and simpler the problem. The bottom (large-scale) layer also includes some more complex problems such as on-chip interconnection network simulation and resource management scheduling.

VIII. Solving Query Q3

The basic idea toward solving Q3 is to establish a collaborative teaching mode consisting of a scientific research team and a teaching team. Scientific research tutors and lecturers jointly guide practical teaching activities such as application problem decomposition, practical processes, and ability achievement assessment to form an efficient high-performance computing cultivation mechanism to produce competent graduates.

A. Basic Methodology

It is important to strengthen group guidance and to stimulate discussion in class. Research mentors and lecturers participate in idea exchanges on systematic knowledge lectures and classroom discussions. Faculty members can often combine teaching methods such as group interactivity and two-stage teaching to help students understand the core concepts and difficult issues in parallel computing as well as improving students' four-level capability in parallel computational thinking. The communication and instruction of homework and projects are mainly in the classroom. Individual guidance occurs outside the classroom to solve difficult problems related to the interdisciplinary knowledge or comprehensive computer system knowledge. Teachers and students communicate individually or in small groups. They provide individualized learning in different professional fields and at different levels. Under the joint guidance of scientific research mentors and faculty members, students complete activities such as Q&A, consulting documents, experiments, in-depth discussions on interdisciplinary issues and writing academic research papers. For special problems, students contact their teachers who are engaged in related research and are available for individual discussions.

B. Methods to Attract HPC Researchers to HPC Classes

To make students develop a deeper understanding of course content, a senior HPC researcher or system developer teaches some topic in which s/he is proficient. Instructors convey their actual engineering experience to students through classes. They also take part in project design and evaluation sessions. However, teaching is usually very time-consuming. How can faculty members attract these busy researchers to join HPC classes voluntarily? Some of the following cases have been proven to be effective.

Method 1: Stimulate their passions. There are common demands for talented students.

These researchers (or system developers) are usually master tutors. Their own graduate students also require professional HPC education. Before redeveloping these HPC courses, the researchers had to cultivate their own graduate students because previous courses could not accommodate student needs. They agree with the Q1 issue (problems of relatively independent knowledge systems in various fields). Their research work is usually interdisciplinary and needs compound talented students who span multiple application fields. However, the education process is time-consuming and usually ineffective. So, researchers welcome this course and they volunteer their time to be part of course development.

Example: At NUDT the "Parallel Computing System I" and the "High-Performance Computing" courses have successfully attracted HPC researchers to join its classes. They cover five aspects of the courses to include high-performance microprocessor design, high-performance interconnection network design, high-performance system software development, parallel compiler optimization, and parallel application development. In this way, the classes have broad content with depth and practical levels of difficulty.

Method 2: Utilize researchers' educational experiences who usually have experience for nurturing students.

Researchers usually have their opinions and experiences on the Q2 issue (support of HPC talent). They usually have a clear demand for student capability. HPC researchers and developers know exactly what kind of talent they need from HPC students. The participation of these researchers urges faculty members to form problem-solving teaching sessions. HPC researchers learn more about parallel computing platforms than lecturers do. Due to their rich research experiences, they correctly know large-scale computing needs and what kind of parallel computing platforms is more suitable for students. They can help lecturers to setup practical environments. They introduce some real cases for explaining some terms, which makes knowledge more vivid. They provide project materials and form project assignments by working with the lecturers and teachers. Usually real-world problems are not suitable for assignments or projects because they are too difficult for students to solve. Instructors need to tailor practical content and match it to relevant teaching content. In designing projects, lecture teachers and researchers do this procedure together.

Example: At NUDT, the "Parallel Programming" has been a vehicle for projects already discussed in the literature [11]. These projects reflect a wide range of scientific computing applications such as computational fluid dynamics (CFD), text mining of biomedical literature, and so on. These projects greatly help students understand efficiency and scalability issues in HPC. Examples include the following.

- Load balance optimization for CFD molecular dynamics coupling computing
- Parallel mining of biomedical literature-oriented biological variability
- Parallel mining of biomedical literature-oriented genes
- 3-D simulations of human ventricular tissues
- Portable and extendable toolkit for scientific computing (PETSc) based parallel tolerance optimization
- Parallel optimization of spectrum transformation shallow water wave mode
- Cell response big data biological effect evaluation
- Dose planning method (DPM) based radiotherapy parallelization

- Convolutional neural networks (CNN) and convolution process parallelization

Method 3: Help researchers effectively engage and generate the talented HPC students they will need.

In this cooperation mode, researchers can provide not only lecture content and laboratory project materials, but also experiment instructions. Faculty members did much to encourage researchers joining each stage of student practice. The practice or experimental session provides researchers a chance to cultivate their own students by their own way. It is not true that researchers help lecturers; instead, courses help researchers train their own students in more effective ways. Once researchers and lecturers reach an agreement on this point, researchers have a great passion to participate in the entire practices. This innovative teaching approach addresses the Q3 issue.

Example: At NUDT many students struggle to write fully functional and adequately documented parallel programs. To improve student performance, faculty members implemented a moderated two-stage format for some parallel programming projects in graduate-level parallel programming classes [12]. Each project partitions into two stages where students complete the assignment individually without any collaboration in the first stage. Then students work in pairs on the same project in the second stage so they can review each other's work from the first stage and improve their programs collaboratively. For two of the five projects, a meeting occurs between the two stages where instructors moderate group discussions on general issues raised by students. Instructors found that students' performance improved from stage one to stage two. In addition, the two projects with a moderated meeting show better performance gains. Surveys showed that students' perceptions of and experiences with the moderated two-stage projects were effective. Students favor working on two-stage projects because they had a chance to discuss challenging concepts. The moderated discussion session tends to guide students toward the correct path for stage two should they make mistakes in the first stage.

IX. CONCLUSION AND FUTURE STEPS

This paper focuses on three major issues in HPC education, namely, the problems of a knowledge delivery system, the problems of practice and experimental environments, and the problems of teaching modes. The authors presented high-level HPC educational concepts, educational framework designs, and strategies for cultivating capable HPC students.

Before presenting solutions to the three issues, the authors showed unique perspectives and practical experiences regarding student capability and the principles of layer modeling for parallel computational thinking. They suggested ways to restructure and optimize the HPC curricular system with the goal of cultivating student capacity and capability. They also showed a step-by-step practical system in which real platform, real application, and research-teaching integration modes immerse students into an HPC world for better understanding. The authors also showed how researchers with rich computing and engineering experience can become deeply involved in course design, in-class teaching, and in laboratory instruction.

A. HPC Educational Effectiveness at NUDT

The effectiveness of HPC courses at the NUDT mainly lies on two aspects: (1) it created its own HPC curriculum system and a step-by-step practice environment that benefit computing majors and non-computing majors, covering two colleges; (2) it was able to demonstrate that students' innovation capability showed significant improvement. NUDT has achieved satisfactory results in the HPC talented students. For example, one student who participated in a project titled, *parallel mining of biomedical literature-oriented biological variability*, has published several high-quality papers in the competitive international journals. Relying on the practice projects, some students participated in several competitions many times with excellent achievements such as the Parallel Application Challenge Competition and the International College Supercomputing Competition. These achievements demonstrate that students have significant achievements in moving forward on their way to further discovery. The university has also cultivated many outstanding high-performance computing graduates for HPC development and research.

B. Implementing HPC Education at Other Universities

NUDT has applied and promoted its HPC curriculum system at other institutions such as Sun Yat-sen University, Hunan University, the National Supercomputing Centers in Guangzhou and in Changsha, with effective results. Multiple universities have jointly completed the "High-Performance Computing Course Construction" project, built dozens of core courses including high-performance computing systems, algorithms and applications that cover ocean simulations, intelligent computing, and other typical application areas.

C. Future Steps

Among its next steps, NUDT will further reinforce its practice exercises and projects and gradually form a more robust set of exercises. Other activities include cooperation with some other top universities in China to promote HPC educational ideas and strategies for cultivating competent HPC graduates. This work has already begun.

NUDT is doing its best to form a template of parallel computing course and practice guidance that could be applied in other Chinese universities. It expects to change the current situation in China where few universities have HPC courses or sequences. Some universities even want to open new HPC courses, even though they lack a feasible curriculum, experimental platforms, practice methodology, and professional teachers. The situation seems very promising.

ACKNOWLEDGMENT

This work is supported 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), the 2019 Hunan Province Postgraduate Outstanding Professional Case Foundation of China (High-Performance Computing Series Case Library), and the Hunan Province Degree and Postgraduate Education and Teaching Reform Foundation of China under grant No. JG2017B004.

REFERENCES

[1] B. Barney, "Introduction to parallel computing," https://computing.llnl.gov/tutorials/parallel comp/, 2019.

[2] D. Johnson, D. Kotz, and F. Makedon, "Teaching parallel computing to freshmen. 1994," Conference on Parallel Computing for Undergraduates, 1994.

[3] Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER). https://tcpp.cs.gsu.edu/curriculum/?q=node/21183

[4] "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, 2012.

[5] Association for Computing Machinery (ACM) – IEEE Computer Society. "Joint task force on computing curricula: computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science," New York, NY, USA, Tech. Rep., 2013, 999133.

[6] A. B. Bondi, "Characteristics of scalability and their impact on performance," in Proceedings of the 2Nd International Workshop on Software and Performance, ser. WOSP'00. New York, NY, USA: ACM, 2000, pp. 195–203. [Online]. Available: http://doi.acm.org/10.1145/350391.350432

[7] Association for Computing Machinery (ACM) – IEEE Computer Society. "Joint task force on computing curricula: computer engineering curricula 2016: Curriculum guidelines for undergraduate degree programs in computer engineering," New York, NY, USA,

[8] Gagne's Hierarchy of Learning. http://www2.rgu.ac.uk/celt/pgcerttlt/how/how4a.htm .

[9] Juan Chen*, Li Shen and Jianping Yin. Parallel Computational Thinking: Our Practice on Tianhe-2. Invited Report. *SIGCSE 2016 Pre-Symposium: Computational Thinking - A Chinese Perspective*. Memphis, USA, March 2016.

[10] Juan Chen*, Li Shen, Jianping Yin. Parallel Programming Course Development based on Parallel Computational Thinking. In *the Proceedings of ACM Turing Celebration Conference (TURC 2018),* Shanghai, China, May 19–20, 2018. (Best Paper)

[11] Juan Chen*, Li Shen, Jianping Yin, Chunyuan Zhang. Design of Practical Experiences to Improve Student Understanding of Efficiency and Scalability Issues in High Performance Computing (poster). In *the Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE 2018)*. February 21st - 24th, 2018. Baltimore, Maryland, USA.

[12] Juan Chen*, Yingjun Cao, Linlin Du, Youwen Ouyang, and Li Shen. Improve Student Performance Using Moderated Two-Stage Projects. In the *Proceedings of ACM Global Computing Education Conference (CompEd 2019)*. Chengdu, China, May 17-19, 2019.

Appendix A

Partial snapshot for the step-by-step teaching practice exercises and projects.

Type	**Exercises/Projects**	**Courses**
Small-scale verification experiments on classic problems	Parallelizing trapezoidal rule Parallel matrix-vector multiplication Parallel odd-even transposition sort Ping-pong communication test Collective communication test	Introduction to Parallel Programming
	Latency and bandwidth test on NUMA structure	Parallel Computer Architecture
Medium-scale tests based on typical cases and benchmarks	Parallel matrix-matrix multiplication on 32 computer nodes with various scales Parallel Monte Carlo Method to estimate PI Parallel n-body problem	Introduction to Parallel Programming
	NPB benchmark scalability evaluation and analysis HPCC benchmark scalability evaluation and analysis	Advanced Parallel Programming
	Loop optimization algorithm verification Automatic vectorization Data locality optimization	Parallel Compiler and Optimization
Large-scale or comprehensive typical application problem solving	On-chip interconnection network performance simulation and analysis Resource management scheduling methods	Parallel Computing System-I High-Performance Computing
	The Weather Research and Forecasting Model based numerical simulation	Parallel Computing System-II
	Computational Fluid Dynamics performance analysis and optimization FFT algorithm test and optimization Parallel sparse matrix multiplication Multi-grid scalable parallel algorithm performance testing and evaluation	Large-Scale Parallel Application and Optimization High-Performance Computing