What Influences Students' Understanding of Scalability Issues in Parallel Computing?

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

Brett A. Becker
School of Computer Science
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

Youwen Ouyang
Department of Computer Science and Information Systems
California State University, San Marcos
San Marcos, CA, US
ouyang@csusm.edu

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

ABSTRACT

Graduates with high performance computing (HPC) skills are more in demand than ever before, most recently fueled by the rise of artificial intelligence and big data technologies. However, students often find it challenging to grasp key HPC issues such as parallel scalability. The increased demand for processing large-scale scientific computing data makes more essential the importance of mastering parallelism, with scalability often being a crucial factor. This is even more challenging when non-computing majors require HPC skills. This paper presents the design of a parallel computing course offered to atmospheric science majors. It discusses how the design addressed challenges presented by non-computer science majors who lack a background in fundamental computer architecture, systems, and algorithms. The content of the course focuses on the concepts and methods of parallelization, testing, and the analysis of scalability. Considering all students have to confront many (non-HPC) scalability issues in the real world, and there may be similarities between real-world scalability and parallel computing scalability, the course design explores this similarity in an effort to improve students' understanding of scalability issues in parallel computing. The authors present a set of assignments and projects that leverage the Tianhe-2A supercomputer, ranked #6 in the TOP500 list of supercomputers, for testing. We present pre- and post-questionnaires to explore the effectiveness of the class design and find an 11.7% improvement in correct answers and a decrease of 36.8% in obvious, but wrong, answers. The authors also find that students are in favor of this approach.

KEYWORDS

Atmospheric science majors, Computing non-majors, Scalability, Undergraduate education

https://doi.org/10.22369/issn.2153-4136/12/2/12

1 INTRODUCTION

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 atmosphere, earth, and environmental realms [1], in addition to commercial applications such as data mining and transaction processing.

Integrating parallel computing earlier into the undergraduate curriculum has been under exploration since the 1990s. In 1994, Donald Johnson et al. [2] proposed teaching parallel computing to freshmen by integrating parallel computation into a data structures course. The Curriculum Development and Education Resources (CDER) center for parallel computing education proposed a detailed curriculum and promoted progress in parallel computing undergraduate courses [3]. Nonetheless, introducing parallel computing into the early years of a bachelors curriculum remains challenging.

Reflecting the growing importance of parallel computing in undergraduate curricula, CS2013 (the ACM/IEEE Joint Computer Science Curricula) stated, "Previous curricular volumes had parallelism topics distributed across disparate KAs [knowledge areas] as electives. Given the vastly increased importance of parallel and distributed computing, it seemed crucial to identify essential concepts in this area and to promote those topics to the core" [4, p 29]. CS2013 introduced a new KA in parallel and distributed computing which explicitly names scalability [5] as a core-tier2 topic.

Understanding scalability issues is key for learning parallel computing. This paper introduces the design of a compulsory parallel computing course at the College of Meteorologic Oceanography at National University of Defense Technology in China. Atmospheric scientists need parallel computing to solve design issues for the parallelization of optimal interpolation algorithms and atmospheric data analysis [6]. This course is intended to provide a broad overview of the topics in parallel computing, as a lead-in for more advanced classes that follow it. These non-computing majors need to leverage HPC to make optimal use of their applications and to solve problems on different scales. Proper solutions resulting in satisfactory speedup are necessary but difficult to design. To start, these applications need to consider the physical architecture to and fully exploit hardware parallelism. The increase in large-scale

scientific computing data aggravates the difficulty of exploiting parallelism.

1.1 Challenges of Understanding Scalability

Scalability is the mechanism by which a parallel system's speedup changes with the number of available processors. Amdahl's law dictates the achievable speedup and efficiency — specifically what happens to efficiency when both the number of processors and the problem size increase. The scalability of a parallel algorithm on a parallel architecture is a measure of the algorithm's ability to effectively utilize an increasing number of processors. Scalability analysis is helpful in selecting the best algorithm/architecture combination for a given problem under different constraints on the growth of the problem size and the number of processors [7, 8].

Scalability is divided into hardware scalability and software scalability, which refers to the ability of system to deliver greater computational power when the amount of resources is increased. Hardware scalability refers to the capacity of the whole system, which theoretically can be proportionally increased by adding more hardware. Software scalability refers to parallelization efficiency - the ratio between the actual speedup and the ideal speedup over a given number of processors [9].

The scalability of a system can take many forms, including speed, efficiency, size, applications, generation, and heterogeneity [10, p 63]. In terms of speed, a scalable system is capable of increasing its speed in proportion to the increase in the number of processors. In terms of efficiency, a scalable parallel system means its efficiency can be kept fixed as the number of processors is increased, provided that the problem size is also increased. Scalability testing can be performed at the hardware or software levels. Parameters used for scalability testing differ from one application to another. Different forms of scalability were also mentioned in [10, p 66], "In his vision on the scalability of parallel systems, Gordon Bell indicated that in order for a parallel system to survive, it has to satisfy five requirements. There are size scalability, generation scalability, space scalability, compatibility, and competitiveness." Three of these survivability requirements are the forms of scalability. Here size scalability measures the maximum number of processors a system can accommodate. Generation scalability refers to the ability of a system to scale up by using next-generation components.

HPC application scalability is inherently complicated as the performance of modern HPC systems approach exascale. Exascale computing refers to computing systems capable of at least a billion billion calculations per second. It is becomming even more complex for HPC applications to fully exploit hardware parallelism, due to many factors. In addition, many applications have poor scalability regardless of the underlying hardware. See [11] for more details.

Scalability modeling and evaluation for real problems are often abstract. Programs are often designed and tested for smaller data sets on fewer processors, but the real problems are much larger and need more hardware, in recent times scaling up to millions of cores. Performance and correctness of programs based on scaled-down systems is difficult to establish [12, p 208], but it remains a cost efficient and practical means of testing. Based on such tests, performance extrapolation is not intuitive.

1.2 Research Goals

This work has three research goals:

- RG1 Explore the effects of understanding or misunderstanding parallel computing scalability on students' performance.
- RG2 Explore the relationship between real-world scalability and parallel computing scalability from the perspective of understanding and learning.
- RG3 Explore students' questions valuable to the understanding of parallel computing scalability.

2 BACKGROUND

2.1 Parallel Computing Course and Scalability

Many modern instructors agree that parallel computing topics should be covered in first- or second-year undergraduate classes [2, 13–15]. Additionally, in Section 1, the authors discussed the fact that parallelism is also a growing trend to which CS2013 has responded. The primary reasoning is that a solid understanding of parallel computing concepts will tremendously improve students' ability to write software that is able to effectively utilize the underlying parallel hardware architecture. For example, Yousun Ko et al. [14] found if parallel computing concepts were introduced as a senior-level undergraduate or graduate elective, students had difficulty transitioning from sequential to parallel thinking. Lori Pollock et al. [16] also thought parallel programming required a very different thought process from traditional sequential programming, as the programmer must think differently, such as performing tasks in parallel, organizing information communication, and balancing workload between parallel processes. Making such a switch from sequential thinking to parallel thinking was a big step for many students. CS2013 recommends parallel computing could be their freshmen or sophomores courses.

Some challenges in parallel computing courses are closely related to scalability. For example, Yousun Ko et al. [14] presented a challenge problem for understanding parallelism. They chose an analogy from cooking to introduce task, data, and pipeline parallelism. They also used the concrete example to illustrate task parallelism and data parallelism. Another challenge they presented is about parallel program performance evaluation. They observed the inevitable question was, "Why is my parallel program slower than the sequential version?" They answered this question by introducing the definition of speedup, scalability, and efficiency, followed by Amdahl's law and performance bottleneck analysis. Besides the above two challenges, Yousun Ko et al. [14] presented three other course modules and challenge problems. They used the decomposition approach of knowledge to structure the course as five course modules, among which each module teaches one fundamental concept of parallel programming. All parallel programming concepts were introduced with the help of worked-out programming examples.

Besides the challenges of switching from sequential thinking to parallel thinking mentioned above, Pollock et al. [16] presented the practical challenges for inexperienced programmers: i) lack of stable and useful debugging tools; ii) the need to analyze why their program is not performing well in parallel and how to improve its performance. They used cooperative learning to meet the practical

challenges, as well as to provide a more real-world context to the course. In the experience of teaching HPC [17], H. Neeman et al. used analogies to explain some concepts to capture the fundamental underlying principles without going deep into technical details.

2.2 Real-World Scalability and Cognitive Ability

There are many scalability issues in real life. For instance, compound interest concerns how investments scale with time, and population growth concerns how the population of reproducing organisms scales with time. Fittingly, most HPC and parallel computing concepts also come down to time — after all, that is why these domains exist — to do more in less time. However, it is well known that many people have trouble truly comprehending the growth of a fund due to compound interest, or the growth of populations with time. It seems scalability is linked to cognition.

In 2005, Frederick introduced the cognitive reflection test (CRT), which researchers have cited nearly 3,500 times. The CRT is a simple three-item measure of one type of cognitive ability. Specifically, Frederick found that CRT scores were predictive of the kinds of choices that prominently feature in tests of decision-making theories. The CRT questions are [18, p 27]:

Q1 A bat and a ball cost $1.10 in total. The bat costs $1.00 more than the ball. How much does the ball cost?

Q2 If it takes 5 machines 5 minutes to make 5 widgets, how long would it take 100 machines to make 100 widgets?

Q3 In a lake, there is a patch of lily pads. Every day, the patch doubles in size. If it takes 48 days for the patch to cover the entire lake, how long would it take for the patch to cover half of the lake?

Interestingly, questions Q2 and Q3 deal with scalability, and Q2 also concerns explicit parallelism.

The CRT questions are crafted very carefully, and for a reason. Each question has an incorrect "intuitive" answer. Frederick provides substantial evidence that these incorrect yet intuitive answers are indeed intuitive. Two pieces of evidence are 1) the incorrect intuitive answers, such as 24 days (half of 48 days) for Q3[1], dominate in trials with large populations; and 2) respondents answer with the correct response much more often with analogous problems that invite more computation (i.e. don't have obvious intuitive answers). For example, respondents miss Q1 much more often than they miss this analogous problem that essentially forces calculation due to the lack of an intuitive answer:

- A banana and a bagel cost 37 cents. The banana costs 13 cents more than the bagel. How much does the bagel cost?

We will come back to the idea of questions on real-life scalability concepts and "intuitive answers" in Section 4.2.

3 STUDY DESIGN

The study was carried out in College of Meteorologic Oceanography in Spring 2019. Fourteen sophomores enrolled in the "Parallel Computing" course, and all students participated in the study. All of them had no prior experience with systematic computer architectures, operating systems, and algorithms. The prerequisite of the course is C programming.

3.1 Course Design

Figure 1 shows the structure of the course, including 12 lecture classes, 6 laboratory classes, assignments, projects, and pre-/post-questionnaires. The length of each class period is 90 minutes with a 10-minute middle break. Formative assignments were released once or twice each week and required to be finished before Sunday night 23:00 pm. Projects were released on Thursday night laboratory class between Weeks 4 and 7. Each project lasted one week until the next Wednesday night.

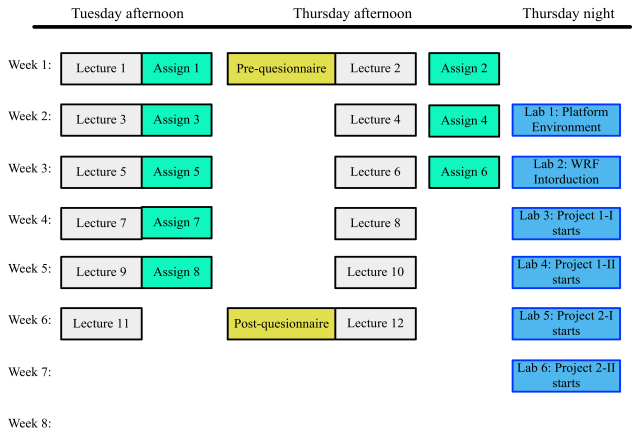

Figure 1: Weekly and daily structure of the course.

The course learning objectives for the lecture part are shown in Table 1. One of the authors is the lead instructor of this course. In order to teach scalability, the authors split the effort from the following eight aspects, which are abbreviated D1–D8 in Table 4.

Table 1: Parallel computing course content topics.

Topic	Content	Lecture
1	Overview of parallel computing: Flynn's taxonomy, parallel hardware and software, interconnection network, etc.	1–2
2	Basic principles of parallel computing: task partition, parallel task scheduling, principles of parallel algorithm design, performance metrics, concepts of speedup/efficiency/scalable applications, etc.	3–4
3	Distributed-memory programming with MPI, collective communication, performance evaluation of MPI programs, scalability, etc.	5–7
4	Shared-memory programming with OpenMP, data dependences, loop scheduling, cache coherence, etc.	8–9
5	Applications: Jacobi methods and computation-communication overlap, numerical weather forecast model WRF, numerical climate forecast model CAM, etc.	10–12

1) Decompose the scalability topic into some detailed notions. According to core topics of parallel computing provided by NSF/IEEE-TCPP Curriculum Initiative [3], the authors decompose the scalability topic into three-type 19 notions as Table 2 shows. The three

[1]The correct answer is 47 days. If the patch doubles in size every day, one day before the final day, 1/2 of the pond must be covered.

types are architecture, programming, and algorithms. Each notion has its learning outcome. The scalability topic is too abstract to teach. However, this decomposition can help make clear what scalability stands for in the parallel computing world, such as what detailed contents of scalability people should teach students and what learning outcome students should have.

2) Design the assignments. Design the fundamental assignments as well as literature reading tasks. The course includes eight formative assignments and a literature reading task throughout an 8-week period. Each of these was graded based on functionality and documentation. All eight formative assignments are about the answer to some basic questions, followed by fundamental parallel programming exercises, which are suitable to all different majors (See Assignments 1–8 in Table 3 for more details). The literature reading task is special for atmospheric science majors in order to deepen their understanding of weather research, the forecasting model, and its parallel solution method. The students need to read at least one paper from the following three papers, which are titled "Development of a Next-generation Regional Weather Research and Forecast Model", "Precipitation Simulations Using Weather Research and Forecasting (WRF) as a Nested Regional Climate Model," and "Sensitivity of WRF Forecasts for South Florida to Initial Conditions."

3) Design the projects. Two application projects are special for Atmospheric Science majors in our class. These two projects are both about numerical weather forecast simulation on a WRF model. WRF model simulation is fundamental to most atmospheric science majors in their professional study and research. WRF is short for the Weather Research and Forecasting model, which is a mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications [19]. The WRF model features two dynamical cores, a data assimilation system, and software architecture supporting parallel computation and system extensibility. The model serves a wide range of meteorological applications across scales from tens of meters to thousands of kilometers. Before the 2019 spring semester, students did not use WRF until the senior year. It was a challenge for sophomores to finish the two projects. In order to reduce the difficulty of studies, the authors divided Project 1 into two phases: Project 1-I and Project 1-II. Project 1-II is a moderately incremental release based on Project 1-I. The same is true for Project 2. See Table 3 for more details of the two projects. Before releasing the two projects, the authors have two laboratory classes to introduce the Tianhe-2A system environment and basic usage, followed by the WRF model background (Lab 1 and Lab 2 in Figure 1). The Tianhe-2A supercomputer is ranked #6 in the TOP500 list of supercomputers[2].

4) Correlate scalability topic & notions to assignments and projects. The authors connect all the scalability notions to eight assignments and four projects in terms of the contents of assignments and projects as well as each learning outcome shown in Table 2. This correlation is helpful to grade the understanding of scalability. The scalability grade for each assignment and project can be given based on students' learning outcomes. The scalability grade of each student is counted by the average scalability scores of all assignments and projects.

[2]www.top500.org. TOP500 List, November 2020

Table 2: Decomposition of scalability topics and correlation of scalability notions, learning outcomes, and assignments & projects. The rightmost column 'Learning Outcome' is taken from Tables 1-3 in Reference [3]

Topics/Notions	Assign	Project	Learning Outcome
Architecture			
SMP → Buses	A1	-	Limited bandwidth and latency, scalability issues
NUMA → Directory-based CC-NUMA	-	-	Be aware that bus-based sharing does not scale, and directories offer an alternative
Message passing (no shared memory)	-	-	Shared memory architecture breaks down when scaled due to physical limitations (latency, bandwidth) and results in message passing architectures
Latency	-	P1, P2	Know the concept, implications for scaling, impact on work/communication ratio to achieve speedup
Bandwidth	-	P1, P2	Know the concept, how it limits sharing, and considerations of data movement cost
Cache organization	-	-	Know the cache hierarchies, shared caches (as opposed to private caches) result in coherency and performance issues for software
Programming			
Shared memory	A8	-	Be able to write correct thread-based programs (protecting shared data) and understand how to obtain speed up
Synchronization	-	-	Be able to write shared memory programs with critical regions, producer-consumer communication, and get speedup; know the notions of mechanisms for concurrency
Performance metrics	A2, A6, A7, A8	P1, P2	Know the basic definitions of performance metrics (speedup, efficiency, work, cost), Amdahl's law; know the notions of scalability
Speedup	A2, A6, A7, A8	P1, P2	Understand how to compute speedup, and what it means
Efficiency	A2, A6, A7	P1, P2	Understand how to compute efficiency, and why it matters
Amdahl's law	A2, A6, A7	P1, P2	Know that speedup is limited by the sequential portion of a parallel program, if problem size is kept fixed
Gustafson's law	A2	-	Understand the idea of weak scaling, where problem size increases as the number of processes/threads increases
Isoefficiency	-	P1, P2	Understand the idea of how quickly to increase problem size with number of processes/threads to keep efficiency the same
Algorithm			
Speedup	A3, A6, A7	-	Recognize the use of parallelism either to solve a given problem instance faster or to solve larger instance in the same time (strong and weak scaling)
Scalability in algorithms and architectures	A6, A7	P1, P2	Comprehend via several examples that having access more processors does not guarantee faster execution — the notion of inherent sequentiality
Model-based notions	A8	P1, P2	Recognize that architectural features can influence amenability to parallel cost reduction and the amount of reduction achievable
Matrix computations	A5, A6, A7	-	Understand the mapping and load balancing problems on various platforms for significant concrete instances of computational challenges that are discussed at a higher level elsewhere
Matrix product	A6, A7	-	Observe a sample "real" parallel algorithm

Table 3: Descriptions of eight assignments and two projects.

Assignments & Projects	Tasks
Assignments 1–3	Search for information: TOP500, parallelism of pipeline, parallelism of vector operations, speedup formulas, Foster's methodology.
Assignments 4–7	MPI programs: trapezoidal rule MPI parallelization; matrix-vector multiplication MPI parallelization with row partitioning; matrix-vector multiplication MPI parallelization with column partitioning.
Assignment 8	OpenMP programs: Odd-even transposition sort OpenMP parallelization.
Project 1-I	Install WRF software and library, and run an ideal case with different amount of computing nodes.
Project 1-II	Install WRF software and library, and run two more ideal cases with different amount of computing nodes.
Project 2-I	Compile real model em_real and run a real model data. WRF preprocessing system WPS need to be installed before running. Download actual observational data from the official website and do the test with different parallel scales.
Project 2-II	Reset the model domain by modifying the model grid parameters (resolution, range, and grid location) in model input file 'namelist.input', recompile the real model em_real and run real model data. Repeat the steps of Project 2-I.

5) Design incentive mechanism to encourage questioning in class. Students were encouraged to question in class by adding the number of questions into final grades. On average, more than ten questions per class were proposed, which greatly increased participation in class activities and inspired students' learning enthusiasm. After class, TAs collected all the questions and counted the grades, which were published to students every other week.

6) Do experimental instruction. Four teaching assistants (TAs) participated in the class activities, especially being involved in experimental instruction in laboratory classes. Before projects began, one TA presented a talk to introduce the background, demands, and expected outcomes. TAs finished all the experimental steps and prepared a detailed experiment instruction manual before students started projects. In laboratory classes, students were divided into four groups and each TA gave individual tutoring to one group of students having difficulties. Students were required to write scientific reports describing their experiments, including objective, platform & environment, steps, results & analyses, questions, and experiences. Students struggled with writing these experimental reports and the analysis of parallel program performance and scalability. This appeared to be busy work to the non-CS majors. For some common questions, the teacher asked one student to present their initial results firstly and then organize a discussion to analyze reasons. Finally, students reached an agreement and designed one more experiment to validate their assumptions. A discussion usually lasted about 15 minutes. The authors twice had such discussions.

7) Design of the evaluation and grading mechanism. This course had no final examination. Student performance was scored by assignment scores, in-class questioning, attendance, project scores, and literature reading scores. Section 4.1 gives the detailed evaluation method.

8) Design pre-/post-questionnaires and feedback questionnaire. Before the parallel computing course began, a pre-questionnaire was administered to test the understanding of students' real-world scalability. These six questions all draw from real life examples of scalability — See Q1–Q3 in Section 2.2 for examples. Each question has three categories of answers: an incorrect yet intuitive answer, an "obviously" wrong answer, and a correct answer. The authors did not expect (or want) the students to simply calculate the right answer — any of these questions can be easily calculated given enough time. Instead, we were trying to test their "intuition" of what answer "seems" correct. In other words, we wanted to measure students' real-world intuitions that they have gained through experience. Students were told to try to capture their "gut feeling" — their intuition — when answering the questionnaire. Accordingly, students were given three minutes (30 seconds per question). At the end of the course, a post-questionnaire, the same as the pre-questionnaire, was administered to students. We then analyzed these scores using a paired statistical significance test called the *Scalability Understanding Paired Test* and correlated the questionnaire results with final course grades. Students were not formally assessed on their questionnaire responses/scores — in other words, the questionnaire scores did not factor into final grades.

At the end of the course, students were encouraged to complete a feedback questionnaire inquiring about their understanding of scalability issues, including a self-evaluation of the learning outcomes of the 19 scalability notions, misunderstandings and correctness experiences with scalability issues, their biggest impression of the course, their confidence toward parallel computing, and other experiences with course activities.

3.2 Correlation of Course Design and Research Goals

What the authors did for teaching scalability in Section 3.1 is based on our three research goals. There are some correlation between them, as Table 4 shows, by ticking √. In Table 4, D1–D8 stand for eight aspects of our course design for teaching scalability in Section 3.1.

Table 4: Correlation of course design and research goals.

RG	D1	D2	D3	D4	D5	D6	D7	D8
RG1	√	√	√	√			√	√
RG2	√						√	√
RG3		√	√		√	√		√

4 RESULTS

4.1 RG1: Scalability Understanding and Performance

We measure student performance using their assignment scores, in-class questioning, attendance, project scores, and literature reading scores. The scores for eight assignments, in-class questioning scores, and attendance comprise 30% of the final course mark. The scores for Project 1-I, Project 1-II, Project 2-I, and Project 2-II comprise 50% of the final mark, and literature reading scores comprise 20% of the final mark. The final course grades for all students are shown by the blue solid line in Figure 2.

According to Table 2, the authors measured student scalability grades for each assignment and each project. This scalability grade for each assignment and project is divided into three levels of achievement: sophisticated (90 points), competent (70 points) and

not yet competent (50 points). Then, we calculated the average scalability grade for all assignments and projects, which constitutes the scalability grade of each student for learning parallel computing (See the red dashed line in Figure 2). A chi-squared test of goodness-of-fit was performed to determine whether the scalability grade of a student for learning parallel computing is linearly correlated to the final course grade. The scalability grade of a student and the final course grade were positively correlated, $r(14) = 0.66, p = 0.010$. This is evidence that one can reasonably expect a better course performance from students with a better understanding of parallel computing scalability.

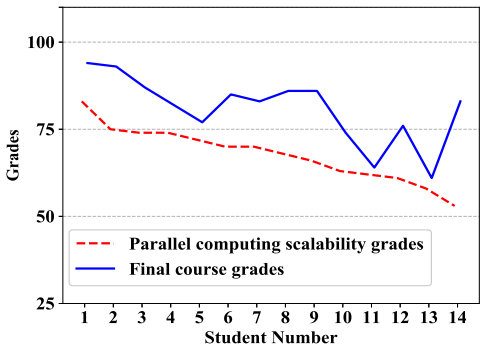

Figure 2: The impact of understanding or misunderstanding parallel computing scalability on students' performance.

At the end of the course, a multiple-choice feedback questionnaire was sent to all 14 students, and all of 14 students responded. The feedback questionnaire was designed to examine students' experience with learning parallel computing scalability. Questions in the feedback questionnaire were three multiple-choice, two single-choice, and three subjective questions. As can be seen, 78% students thought parallel computing scalability studies were very beneficial for understanding key concepts in parallel computing and improving their parallel programming ability. The remaining students thought such help was okay. Students self-evaluate their understanding of parallel computing scalability with five stars. The proportions of five, four, three, and two stars in students' self-evaluation are 21%, 28%, 35%, and 14%, respectively.

In the three multiple-choice quesitons involving architecture, programming, and algorithms, students picked out some items (rows) from Table 2 for which they thought they have achieved the learning outcomes. According to students' self-evaluation, the top five items the students achieved are *Programming*-Speedup (14/14), Efficiency (13/14), Amdahl's law (12/14), Gustafson's law (11/14) and *Algorithm*-Speedup (11/14). The worst five items are *Programming*-Isoefficiency (1/14), *Architecture*-NUMA→Directory-based CC-NUMA (3/14), Cache organization (4/14), *Programming*-Shared memory (4/14), and Synchronization (4/14). This is evidence that our decomposition of scalability topics is really beneficial to teach students to understand scalability issues. It also shows that more assignments and projects can help students better understand scalability notions. In both of the two projects as well as several fundamental programming assignments, students need to use speedup, efficiency, and Amdahl's law concepts to calculate and evaluate the results. Repetitive calculations correct some misunderstandings and deepen their understanding to scalability. On the contrary, for those knowledge notions lacking exercises, it is hard to expect students to have a good learning performance. For example, the NUMA concept, cache organization, shared memory, and synchronization are not directly relevant to assignments or projects. The learning performance of understanding these notions is worse than that of understanding speedup, efficiency, Amdahl's law, and Gustafson's law.

Students needed to give at least one experience of misunderstanding parallel computing scalability. We list all the feedback as follows, merging some same or similar feedback.

- Students A1, A2 thought that the running time was inversely proportional to the number of processes. But in the actual cases, they observed that running time sometimes was constrained by bandwidth.
- Students B1, B2 could not understand why the increasing computing power or number of processes sometimes could not bring about speedup improvement.
- Students C1, C2, C3, C4 thought using the more processor cores must result in the faster speedup, the higher efficiency, and the stronger scalability, but by experimental results they found more processor cores were not sure to bring about higher performance, and sometimes parallel time would be reduced only when the number of processor cores reaches a certain value.
- Student D thought scalability was only related to the application problem itself, but they later found scalability was also closely related to the parallel algorithm.
- Student E thought matrix-vector parallelism in row partition had the same effect with that in column partition, but by communication analysis and experimental results, they found different matrix partitions would bring about different amounts of collective communication and also big performance differences.
- Student F thought the scalability only represented the running time of a program and the shorter running time meant better scalability.
- Students G1, G2, G3 thought there was no relationship between speed, the amount of input data, and the number of processes/threads, which prevented them from understanding speed and scalability.

They were asked to explain if learning more knowledge of scalability and overcoming the misunderstandings of scalability were useful to their performance improvement of learning parallel computing or not. All the answers are YES.

4.2 RG2: Real-World vs. Parallel Computing Scalability

We released a pre-questionnaire and a post-questionnaire to test students' real-world scalability cognitive ability. The correctness ratio increased by 11.7% from the pre-questionnaire to the post-questionnaire. The number of obvious but wrong answers was

reduced by 36.8%. This proves that students' understanding of scalability improved. The Pearson correlation coefficient between pre-questionnaire grades and post-questionnaire grades was 0.73.

We compare two questionnaire results and parallel computing scalability grades in Figure 3. The Pearson correlation coefficient between pre-questionnaire grades and parallel computing scalability grades is 0.0079. This proves that real-world scalability grades and parallel computing scalability grades are correlated with very low strength. The Pearson correlation coefficient between post-questionnaire grades and parallel computing scalability grades is increased to -0.34. It shows that the authors could expect the correlation strength between real-world scalability and parallel computing scalability can be increased by learning parallel computing.

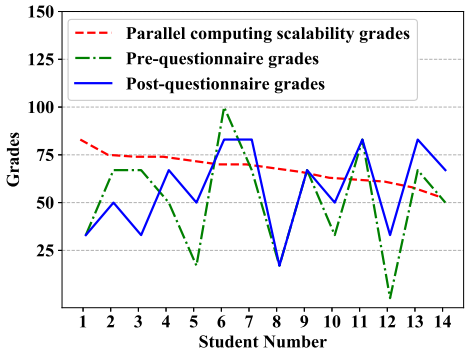

Figure 3: The relationship between real-world scalability and parallel computing scalability.

4.3 RG3: Students' Questions Valuable to the Understanding to Scalability

Students' questions mainly came from in-class questions, laboratory class questions, and project questions. Each class lecture averaged ten questions. These questions were commonly about topics such as architecture concepts and programming problems. Major questions related to scalability issues were as follows: How to calculate the efficiency? What is weak scaling? What is strong scaling? How to identify the scalability of one program? What is the difference between Amdahl's law and Gustafson's law? How to understand the four steps of Foster's parallelization method and how to apply the Foster method to solve a real application? How to calculate the number of collective communications for different partitioning approaches?

Lab class questions are mainly about programming problems, algorithm implementation, and actual operations on the Tianhe-2A supercomputer, such as how to assign jobs to the expected amount of computer nodes? How to map processes to computer nodes and processor cores? Below are some choice questions that show the value of understanding parallel computing scalability.

Q1 The impact of different matrix partitioning methods on parallel performance in a matrix-vector multiplication program.
Question: For a matrix-vector multiplication program, which is better, partitioning the matrix by column or by row? Why?
Teaching Solution: Analyze two matrix partitioning methods (by row and by column) and compare their numbers of collective communications. Run programs and compare execution time under two partitioning methods. Students will find partitioning the matrix by column would produce more collective communications compared to partitioning the matrix by row and then result in parallel performance reduction.

Q2 The impact of memory bandwidth on speedup and efficiency.
Question: There was a case in Project 1-II: a program with 8 processes achieved higher speedup than with 4 processes. But the program running with 16 processes did not achieve expected speedup like with 8 processes. Why? Then the speedup with 32 processes were again higher than that with 16 processes. Why?
Teaching Solution: For a shared-memory architecture on one compute node of the Tianhe-2A supercomputer we count the number of physical cores used for each row and consider the impact of limited memory bandwidth on speedup and efficiency. We doubt this speedup jump is related to the number of assigned cores on one compute node. It is possible to do more experiments to verify our assumption, where different numbers of compute nodes and processor cores are assigned.

Q3 The impact of architecture and node allocation strategy on parallel execution time.
Question: Students found an 8-process program running on one compute node was slower than running on two compute nodes. Why does the fixed amount of processes have a different execution time? How do different node allocations influence parallel execution time?
Teaching Solution: Illustrate the concept of memory bandwidth and list the facts that influence memory bandwidth. The limited memory bandwidth on a shared-memory architecture sometimes has an influence on parallel execution time. Analyze why different assignments of processes to compute nodes will bring about different actual memory bandwidth on one node. Suggest an experiment to verify the assumption: parallel execution time of a memory-intensive program will be greatly influenced by actual memory bandwidth.

5 CONCLUSIONS

This study focused on what influenced students' understanding of parallel computing scalability issues. The authors found understanding or misunderstanding parallel computing scalability has a correlation with students' performance. We explored the relationship between real-world scalability and parallel computing scalability, and we found real-world scalability and parallel computing scalability were correlated with low strength. There is no evidence to prove real-world concepts and experiences will greatly influence the learning of parallel computing concepts. However, the authors could expect the correlation strength between real-world scalability and parallel computing scalability can be increased by learning parallel computing. We need more research and analyses about the two types of scalability in the future. Finally, the authors showed some examples of student questions that are valuable to the understanding of parallel computing scalability. According to pre- and post-questionnaires, the effectiveness of our parallel computing course resulted in an 11.7% improvement in correct answers and a

decrease of 36.8% in obvious but wrong answers. Most students are in favor of the approach used.

ACKNOWLEDGMENTS

The authors would like to thank Dr. John Impagliazzo (Hofstra University) for providing insight to improve the paper and the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the 2017 Hunan Province Degree and Graduate Education Teaching Reform Research Foundation of China under Grant No. JG2017B004, the 2019 Hunan Province Higher Education Teaching Reform Research Foundation of China (titled with Teaching Practice of Training High-Performance Computing Talents Relying on High-level Scientific Research), and the 2019 Hunan Province Postgraduate Outstanding Professional Case Foundation of China (titled with High-Performance Computing Series Case Library).

REFERENCES

[1] Blaise Barney. Introduction to parallel computing. https://computing.llnl.gov/tutorials/parallel_comp/, 2019.

[2] Donald Johnson, David Kotz, and Fillia Makedon. Teaching parallel computing to freshmen. 1994. *Conference on Parallel Computing for Undergraduates*, 1994.

[3] 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.

[4] Joint task force on computing curricula, association for computing machinery (acm) and ieee computer society. computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. Technical report, New York, NY, USA, 2013. 999133.

[5] André B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd International Workshop on Software and Performance*, WOSP '00, pages 195–203, New York, NY, USA, 2000. ACM.

[6] G. von Laszewski. An interactive parallel programming environment applied in atmospheric science. In *Proceedings of the 6th Workshop on the Use of Parallel Processors in Meteorology, G.-R. Hoffman and N. Kreitz, Eds., European Centre for Medium Weather Forecast. Reading, UK: World Scientific*, pages 311–325, 1996.

[7] V.P. Kumar and A. Gupta. Analyzing scalability of parallel algorithms and architectures. *Journal of Parallel and Distributed Computing*, 22(3):379–391, Sep 1994.

[8] J. Y. Shi, M. Taifi, A. Pradeep, A. Khreishah, and V. Antony. Program scalability analysis for hpc cloud: Applying amdahl's law to nas benchmarks. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 1215–1225, Nov 2012.

[9] Xin Li. Scalability: strong and weak scaling. https://www.kth.se/blogs/pdc/2018/11/scalability-strong-and-weak-scaling/, 2018.

[10] Hesham El-Rewini and Mostafa Abd-El-Barr. Advanced computer architecture and parallel processing. John Wiley & Sons. ISBN 978-0-471-47839-3., 2005.

[11] JJ Dongarra. *On the Future of High Performance Computing: How to Think for Peta and Exascale Computing*. Hong Kong University of Science and Technology Hong Kong, 2012.

[12] Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar. *Introduction to Parallel Computing (2nd Edition)*. Pearson Education Limited, 2003.

[13] E. Saule. Experiences on teaching parallel and distributed computing for undergraduates. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 361–368, May 2018.

[14] Yousun Ko, Bernd Burgstaller, and Bernhard Scholz. Parallel from the beginning: The case for multicore programming in the computer science undergraduate curriculum. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 415–420, New York, NY, USA, 2013. ACM.

[15] D. J. John. Integration of parallel computation into introductory computer science. In *Proceedings of the Twenty-third SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '92, pages 281–285, New York, NY, USA, 1992. ACM.

[16] Lori Pollock and Mike Jochen. Making parallel programming accessible to inexperienced programmers through cooperative learning. In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, pages 224–228, New York, NY, USA, 2001. ACM.

[17] H. Neeman, J. Mullen, L. Lee, and G. K. Newman. Supercomputing in plain english: Teaching high performance computing to inexperienced programmers. In *Proceedings of the 3rd LCI International Conference on Linux Clusters: The HPC Revolution*, 2002.

[18] Shane Frederick. Cognitive reflection and decision making. *Journal of Economic perspectives*, 19(4):25–42, 2005.

[19] Weather research and forecasting model. national center for atmospheric. https://ncar.ucar.edu/what-we-offer/models/weather-research-and-forecasting-model-wrf, 2020.