Hybrid Small Class Teaching: Dividing and Conquering Large Computer Systems Classes

Yao Guo
Peking University
School of EECS
yaoguo@pku.edu.cn

Junlin Lu
Peking University
School of EECS
ljl @pku.edu.cn

Yifeng Chen
Peking University
School of EECS
cyf @pku.edu.cn

Ming Zhang
Peking University
School of EECS
mzhang_cs @pku.edu.cn

Wenxin Li
Peking University
School of EECS
lwx @pku.edu.cn

ABSTRACT

Recent years have witnessed the rapid increase of undergraduate students in computer science majors. As a result, many computer science classes have reached hundreds of students, which may affect the quality of in-class teaching for some courses such as computer systems. This paper presents our approach in dealing with this issue with an approach we called *hybrid small class teaching*, which combines large class lectures with small class recitation, to increase the involvement of students and improve the performance of teaching. We have experimented this approach on an "Introduction to Computer Systems" course for the past five years with more than 20 professors involved each year. This paper presents our experiences and lessons learned through the experiments with hybrid small class teaching, as well as insights on how to deal with large classes in similar courses.

CCS Concepts

• Social and professional topics → Computer science education; Computational science and engineering education;

• Computer systems organization → ;

Keywords

Small class teaching, computer systems course, recitation

1. INTRODUCTION

In recent years, students choosing computer science as their majors have been increased rapidly. Combined with strong interests in computer science classes from other majors, many core computer science courses, such as programming and algorithms, have more than 300 students in each class. The quick increase of students may raise challenges in teaching and student performance. Currently, the typical way to deal with large classes is to split the students into multiple classes, either parallel classes taught by different professors in the same semester, or teaching the same courses for two to three times in each academic year. Although this kind of approaches work for introductory programming courses, it will be difficult to find more than a couple of professors to teach more advanced courses such as computer systems, operating systems, or computer architecture.

For example, when we first decide to add the course "Introduction to Computer Systems[1]" in our curriculum, we found that it is a comprehensive course involving multiple subjects that appear to be difficult to many first or second year computer science students. At the same time, with as many as 7-8 programming projects (labs) each semester, the course is very heavy and may be impossible to pass for students with weak backgrounds in computer science.

When the course was first developed in CMU, they realized that the course is very heavy, so that a recitation session was introduced, which is administered by TAs to review the key concepts and help students with difficulties in labs. Although TAs can provide effective assistance to students and help them understand knowledge, they are not as experienced as professors and could not provide in-depth knowledge on various topics in computer systems, or in wider computer science research areas.

With the support of a government grant[2], we decided to introduce a new approach with "*hybrid small class teaching*", in which we split the course into two main sessions: large class lectures with more than 100 students, where regular lectures were taught by two professors covering the key concepts and techniques, and small class recitation with fewer than 15 students, where each professor leads the discussion of course materials, helping students with questions and providing more advanced topics during the process.

The design of the hybrid small class teaching format aims to achieve the following goals:

- Helping students to get through one difficult comprehensive course with heavy workload and a variety of topics. Given the workload and difficulty of the course, if the course was taught in traditional lecture style with over 100 students in each class, we predict that as many as 10% of the student may fail the course if we do not lower the course requirements from the original course design.

ACM TUR-C'17, May 12-14, 2017, Shanghai, China

DOI: http://dx.doi.org/10.1145/3063955.3063960

1 The course was first developed by Carnegie Mellon University as CS213: https://www.cs.cmu.edu/ 213/.

2 "The Test Plan to Cultivate Top-Notch Students in Basic Discipline"

- Giving students the early opportunity to interact with a professor early in their program. As many classes become larger and larger, students do not have the chance to talk to a professor in many occasions until late in their program when they join a lab or take a small-size elective course.
- Giving students the chance to give presentation, interact with students and answer questions. The recitation course is designed to give students more chance to talk and interact with each other. For example, a course review on each lecture is given by a student in the recitation, giving students the opportunity to prepare presentation and answer questions from fellow students.

We have successfully experimented with this hybrid small class format on the *Introduction to Computer Systems* course in the past five years. Each year, we have more than 20 professors involving in the course, either teaching the lectures in large classes, or leading the recitation in small classes. We will present the background, course design and evaluation, as well as discussions on what we have learned through the experiments.

2. BACKGROUND

The course we choose to conduct the experiment is *Introduction to Computer Systems (ICS)*, a core course for computer science majors. The course and accompanying textbook was developed by Carnegie Mellon University more than 15 years ago. Since then, the course has been adopted by over 100 schools around the world.

The course serves as an introduction to various topics related to computer systems, including data layout, assembly code, computer architecture (pipelining), linking, caching, virtual memory, computer networks and Internet, processes and concurrent programming.

The major goal of the course is to give the students an overview of computer systems from a programmer's perspective, such that a student understands how a program is running on a modern computer system. In order to achieve this goal, the course has been designed with heavy workload, which was suggested to be taught in two semesters, instead of teaching the whole materials (in the textbook) in one single semester.

One special feature of the course is a set of 7 to 8 programming projects (labs) designed to help students deeply understand the key concepts and techniques in the course. Some of the labs are relatively easy, but some of them may require more than 20 hours of work in order to get a perfect score. Some labs require the submission to reach a certain level of performance metrics in order to gain 100% credit, which increases its difficulty and competitiveness.

Grading programming projects for hundreds of students would become an invincible task if not for the Autolab environment that was designed to handling the submission and grading all lab submissions in an automated way. Although several labs were not graded by Autolab originally, we have invested a couple of students to develop an extension of Autolab such that we are able to automate the grading process of all labs.

3. COURSE DESIGN

3.1 Overview

Currently, the ICS course is taught in the first semester of the second year, as a required course to all CS major undergraduate students.

Figure 1 shows the overall design of the course in the proposed hybrid small class format. The whole course is composed of two major components: large class lectures and small class recitations.

In large class lectures, we split all students into two parallel classes such that each class includes around 120 students. Each session was taught by two professors, each covering half of the topics. The lecture sessions were taught in traditional lecture style, with limited Q&A opportunities.

We then split the students into to multiple small classes with only 13 to 14 students in each class. As a result, we typically have 16 to 20 small classes, depending on the number of students taking the course each year. Each small class was administered by a professor, with the assistance of a TA, who is either a graduate student or an undergraduate student who had taken the course in previous years.

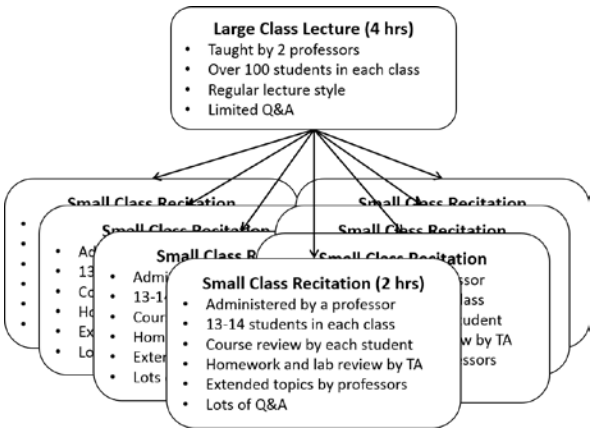

Figure 1: Overview of the course design.

3.2 Course Components

Typically, the small class session includes the following components:

- *Course reviews*. In each session, we review the lectures taught in this week. The review of each lecture was given by a student.
- *Homework and lab review*. The TA will give an overview of the homework problems and solutions each week. He/she will also talk about lab preparation issues and review each lab after it was completed.

- *Questions and discussions*. We use small classroom in order to generate an environment for discussion. Students are told that their participation in the discussions will be counted in their scores. Some professors keep a sheet to count the number of times each student asks/answers questions and convert the statistics into a part of their final scores.
- *Extended topics*. Here is the part where each professor can have some fun, telling stories or giving an overview of their research if there are free time. For example, a professor can give an overview of computer security measures when the course covers buffer overflow, or an overview of computer networking research when the course covers Internet and proxies.

3.3 Student Allocation

In order to split the students into comparable groups for each small class, we use a simple algorithm to guarantee that each small class includes a similar distribution of good students and weak students.

This is important in order to generate discussions because good students are those who can both answer questions and also ask insightful questions. At the same time, the students with poor performances can ask questions and learn from the good students.

We found that the distribution met our design goals as most professors report that the discussion atmosphere has reached a level never seen before in other courses with significantly more students.

After the initial year, we also tweaked the allocation algorithm to make sure each small class includes an equal number of female students. This also helps improve the class atmosphere with a healthy distribution of male and female students.

3.4 Professor Selection

Finding more than 20 professors to teach in a class is difficult. Besides providing the required incentives (which will be discussed later), we try to find more appropriate professors who are both interested in teaching and will also benefit from the class in the long run.

For each large lecture class, we only needs a couple of professors who can cover the whole content, which should be easy to find. However, in order to find professors for the small classes, we follow the following criteria:

- Professors in related subjects, especially who teach follow-up classes that are related to the course. For example, we include professors who teach operating systems, compilers, computer networking, computer architecture, etc.
- Professors who are involved in teaching administration. As this course is related to many other courses. We include several professors who serves as undergraduate program directors or related roles, such that they follow the courses and take notes for future course reforms.
- Professors who are doing research related to computer systems. The professors can give the students an overview of the most recent research topics. This also helps the students as they will have the chance to meet professors they might work with in the future.

3.5 Incentives

Many professors in other universities have asked us the same question: *How do you get all the professors to teach the same course?* This comes down to the incentives.

Besides monetary incentives that is coming from a grant, we believe the professors have the following incentives to join a course like this:

- *Get to know the students in person*. It helps the professors greatly when they need to recruit students to join their labs, especially if they want to find PhD candidate early in their years.
- *Spreading the word on their research*. Even the students in the class might not join the professor's lab, it also helps to talk about their research.
- *Interact with other professors*. We arrange regular meetings after each recitation for the professors to have informal meetings to discuss the progress, issues with students, or just chatting for fun. Many professors like this kind of meetings.

During our five-year experiments of this course, only a few professors quitted after the first year because they were very busy. Most other professors have been happy and devoted to teach the course even the monetary incentives are discontinued in the future.

4. EVALUATION

This section presents the results of experimentation, which is evaluated through student performance, as well as student and professor responses in our interviews.

4.1 Course Statistics

We have taught this course in this *hybrid small class* format for five years. Table 1 shows the basic statistics on this course.

As we can see from the table, the number of students has kept increasing each year, which indicates that the challenges also increase in dealing with so many students in CS majors. However, in our course setup, the only big changes we made is that we split the large class lectures into two parallel sessions in the year 2013 because the number of students has surpassed 200, which we feel maybe too many in a single class.

On the small class setup, it is easy to scale as we only need to recruit one or two new professors each year to keep the number of students in each small class in the range of 12 to 14 students.

Overall, the only logistics issues we met at the beginning is to synchronize each small classes and discuss the difficulties students

might face in each week through emails and face-to-face meetings. The course runs pretty smoothly after the first two years as most potential issues have been resolved.

Table 1: Course statistics in the past five years.

year	# of students	# of parallel lectures	# of small classes
2012	171	1	14
2013	212	2	16
2014	235	2	18
2015	239	2	18
2016	274	2	19

4.2 Student Performance

It is difficult to measure student performance in a course setup like this as we did not have this course in the traditional lecture style before we implement the hybrid small class format. However, the overall performance of the students can still be used as an indication of the success of this teaching format.

As we mentioned earlier, the workload of this course is very heavy, especially after we include the *Computer Architecture* chapter since the third year. However, since we have the recitation format to help the students understand the course and review their performances during the process, most students have completed the course in a satisfactory manner.

Figure 2 shows the (cumulative) distribution of scores for each academic year. Each year, we have a few students (less than 5) who could not finish or pass the course due to various reasons, including mental issues and family matters. For those students who have attended all recitation classes and who have completed all required elements including home works, labs and exams, only one of them have failed the course in the first three years.

The average scores each year is also stable, ranging between 75 to 80 points out of a total of 100 points.

In order to check how the student performances in different small classes might differ, we show the average scores of each small class in each year in Figure 3 We can see that, for each year, the difference between each class is within the range of 3 to 8 points, which may be caused by the differences in individual student performances, instead of professor teaching performances.

More details on the differences between small classes should be investigated further with more statistics from a multi-year study.

Overall, we are satisfied with the current status of the hybrid small class teaching approach. We have already extended the approach on another course for CS major students, and to other majors in our school as well.

4.3 Student Responses

Overall, most students have welcomed the format due to various reasons:

"This is the first time I have the chance to interact with a professor so often after I entered college."

"I am very thankful that the school have invested so many professors in one course, I have the obligation to work hard and learn it well."

"I liked the course review part, as it helped me refresh the lectures. As it was given by a fellow student, it also covered some topics the professors didn't explain very clearly in the lectures."

"The small class discussion really helps solve my questions as they can be answered by the professor, the TA and my fellow students."

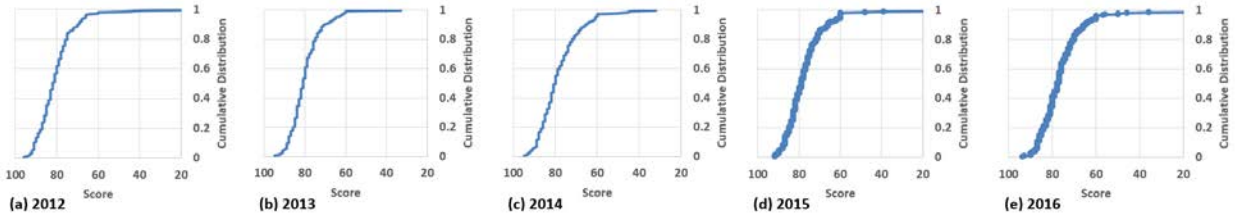

Figure 2: Cumulative distribution for student scores in the past five years.

Figure 3: Average scores for each small class, sorted in descendent order.

"I got to know the students in my small class well and I have made some friends!"

"It was fun."

One student praised on the professor in his small class (in an anonymized survey):

"Professor W was the best. He was very responsible. He explained every point clearly and he really cares about whether we have understood the problem or not. I have benefited from the small class a lot."

Some students still complained about the workload of the course:

"There were so many topics in the course. I didn't have the time to understand all of them deeply. For many topics, I only knew that was presented in the slides."

One student also mentioned the atmosphere in his small class:

"All those talking in the class are THOSE GOOD students. I didn't have a chance to join their discussions because I have no clue..."

Another student did not like the course review part:

"The course review was boring as many students just repeated what was taught in the class lectures."

4.4 Professor Comments

We have talked to several professors to gather their comments, here are some representative ones.

Professor Z commented on what the students could learn from the course:

"Not only could they learn course materials, they can improve their other qualities such as communication skills through the interaction with professors and students, and most importantly presentation skills when they have to present the course review in front of the whole class."

He also commented on whether the students have the incentive to ask questions and join discussions:

"For the students who do not understand some topics, they will of course ask questions. But even for the good students, they will have the incentives to answer questions and show off themselves as they know their participation will be noted by the professors and increase their scores in the end."

Professor X noted that he also benefited from teaching the course:

"I got to know the students in my small class very well, to the point that whether they are suitable for doing research. A couple of them have later worked in my lab. One of them have continued as a MS student!"

4.5 Impact on Follow-up Courses

The ICS course also serves as a prerequisite for many follow-up courses such as compiler, operating systems and computer networks. As students have gained some first-hand knowledge on the basics of these follow-up courses, it helped them to understand the follow-up courses much easily.

Professor B, who taught a follow-up course *Computer Networks*, noted that:

"Because the students have basic understandings of computer network topics such as sockets and TCP/IP, as well as hands-on knowledge on the implementation of a web proxy, they can grasp the topics more quickly and also more interested in the course."

Similarly, Professor G, a *compiler* course instructor, also commented that:

"I found the students after taken Computer Systems are better at understanding the compiler concepts and principles, as they have been familiar with assembly code and learned how programs run on a computer from the previous course."

5. DISCUSSIONS

5.1 Limitations in Evaluation

As the case in many studies regarding course reforming, it is difficult to evaluate the effect of the change in a more accurate matter. We have tried to evaluate the approach with students performances and responses from students and professors. Although the current evaluation can be used to provide a reasonable indication of success, we can still improve the evaluation part further in the future:

- With more data in the future, we are able to provide a more convincing evaluation on whether this kind of approaches can provide consistent benefits to students and professors.
- We plan to design a survey that may include both qualitative and quantitative questions such that more detailed evaluations can be performed.
- As we do not have a previous edition of the same course taught in a different format, we cannot compare the results directly. However, we plan to apply the similar approaches on an existing course, so we can compare the student performance data after and before we implement the hybrid small class teaching approach.

5.2 Applicability to Other Courses

As we mentioned earlier, for introductory courses such as basic programming in Java or C/C++, it is easier to find more professors who can teach the course directly in small classes, instead of using such a hybrid format. For more advanced elective courses, the number of students tends to be small any way. Thus the most

appropriate courses for this hybrid format are those courses that are taken by many students, the content of which is difficult for the students, and teaching the course can benefit a number of professors.

Besides an introductory course on computer systems, we believe other courses can benefit from this approach as well, such as algorithms, operating systems, computer architecture and computer networks, etc.

Currently, we have converted another course *Algorithm Design and Analysis* into the hybrid small class format, which was taught in the following semester, such that each student has the chance to meet two professors in a small class setting in their first two years.

5.3 Sustainability

Investing so many professors in one course raises the issues on whether this approach is sustainable. Our current experiments shows that it is possible to take such kind of approaches because we have over 100 faculty members in the CS-related departments.

If a department has far fewer professors and fewer students, this kind of approaches might not be practical as a small number of students do not need this kind of hybrid approaches.

Nonetheless, if the number of CS students keeps increasing in the future, many CS departments may face similar issues. If only one or two core courses can be offered in a hybrid small class format, it may help the students a lot during their four-year study because they become familiar with at least a couple of professors. This kind of hybrid format can also help improve the success rate for students in difficult courses such as *computer systems*.

Of course, professors may also need incentives to devote themselves to teach an extra course, but the workload of professors can be limited if they follow the course for more than a couple of years.

6. RELATED WORK

Small class teaching has been advocated as an effective approach to improve the performance of teaching. It has been studied in high schools [2], as well as undergraduate students in science and engineering [6].

In computer science teaching, student engagement [5] is one of the key factors that may affect success in a particular class. Previous approaches have shown that techniques such as a linked-courses learning community can help improve student engagement [4, 1]. Our approach in hybrid small class teaching has also focused on improve student engagement as the students have no choice but participate in course discussions in a small classroom.

Many new class formats have been proposed and experimented in recent years, most notably Massive Open Online Courses (MOOCs) [3] and Flipped Classrooms [7]. We believe these techniques can also be incorporated to further improve the small class teaching format.

7. CONCLUSIONS

We have presented our five-year experiments in applying a hybrid small class teaching format in order to deal with a large number of students in difficult CS courses such as computer systems. With a hybrid format including large class lectures and small class recitations, it is able to improve student engagement and improve student success rate. As the students have the opportunity to interact with a professor in small classes, it may provide other long-term effects on student-professor relationship as well as increasing student interests in research.

ACKNOWLEDGMENTS

We thank all the professors who have taught the course, especially the professors and students who have participated in the interviews. The project has been supported in part by the grant “The Test Plan to Cultivate Top-Notch Students in Basic Discipline” from the Ministry of Education of China. This paper is also partially supported by the National Natural Science Foundation of China (NSFC Grant No. 61472006).

REFERENCES

[1] M. Butler, M. Morgan, J. Sheard, Simon, K. Falkner, and A. Weerasinghe. Initiatives to increase engagement in first-year ict. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE'15, pages 308-313, 2015.

[2] B. Nye, L. V. Hedges, and S. Konstantopoulos. The long-term effects of small classes: A five-year follow-up of the Tennessee class size experiment. *Educational Evaluation and Policy Analysis*, 21(2):127-142, 1999.

[3] L. Pappano. The year of the MOOC. *The New York Times*, 2(12):2012, 2012.

[4] A. Settle, J. Lalor, and T. Steinbach. A computer science linked-courses learning community. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE'15, pages 123-128, 2015.

[5] J. Sinclair, M. Butler, M. Morgan, and S. Kalvala. Measures of student engagement in computer science. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE'15, pages 242-247, 2015.

[6] L. Springer, M. E. Stanne, and S. S. Donovan. Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. *Review of educational research*, 69(1):21-51, 1999.

[7] B. Tucker. The flipped classroom. *Education Next*, 12(1):82-83, 2012.