Abstract—This poster describes the author’s experience teaching an upper-level parallel programming course using peer instruction, a flipped classroom technique that uses clickers and peer discussion to help students learn the material. This was a preliminary offering, but shows the potential of adopting new pedagogy for parallel programming education.

Keywords—Flipped classroom; peer instruction; parallel programming education

I. INTRODUCTION

In a traditional lecture-based course, students get their first exposure to the material during class and then do their homework alone afterwards. The idea behind a “flipped classroom” is to reverse these, having students go over the lecture at home and then do homework in class. This is achieved in different ways, but typically students are required to read the textbook or watch a lecture video before class. Then, during class, they build on the understanding acquired beforehand by working homework-like problems. The main advantage of this approach is that the professor is in the room when students apply the course material to problems, which is typically when they realize gaps in their understanding. Thus, the professor can more easily see where students struggle and more quickly move to correct misunderstandings.

One specific approach to flipped classrooms is peer instruction, developed by Eric Mazur to improve physics instruction [1]. This approach gives a specific structure to the presentation of problems during lecture. It typically uses multiple choice problems and votes collected electronically, often using “clickers”, handheld radio devices with which students communicate their selection. First, the professor presents the problem and the students vote on their answer individually. Then the students discuss their answer in small groups and vote again. They again vote as individuals, but are potentially influenced by the discussion. Next, the professor asks students to present and explain their answer. This leads into a group discussion if there is disagreement. Eventually, the professor presents the correct answer and gives a brief lecture if necessary to clarify the concept behind the question.

Each step in the questioning process contributes to student learning. The initial vote commits each student to a specific answer, helping them form an opinion for the small group discussion and also preventing self deception about how well they did. The small group discussion may let students clear up each other’s misunderstandings and also helps students learn by explaining their ideas to each other. It can also increase their confidence so they are willing to contribute to the larger class discussion and/or let them know that other students are struggling to lessen feelings of isolation. Finally, the whole-group discussion and mini-lecture ensure that all students end up with the correct understanding.

When applied in computer science courses, peer instruction has been found to reduce the number of students withdrawing or getting less than a C by an average of 61% [2]. It has also been shown to improve students’ average final exam score by half a letter grade [3].

Here we report the author’s experience using peer instruction in a upper-level elective in parallel programming. To our knowledge, this is the first time the technique has been applied to this course; most reported use of peer instruction in computer science has been in introductory or core courses.

II. THE COURSE

For our course, we did not provide video lectures for students to watch due to the time that would be required to create them. Instead, students read from our textbook, “Principles of parallel programming” by Lin and Snyder [4]. In order to facilitate this strategy, the course followed the textbook fairly closely. In addition, the class read two research papers and used a video tutorial on CUDA [5].

The course was a broad introduction, covering the principles between both shared and distributed memory parallelism. To allow time for reading, the programming component was relatively light. They wrote a parallel quicksort in Chapel, parallelized a Java heat diffusion simulation in two ways (tasks created each time step and long-running threads synchronized with a barrier), and parallelized a large resource management simulator in Java (again using tasks).

Students also completed and presented a final project of their choice, most of which involved substantial programming.

A typical meeting in this class was organized around approximately 10 questions. The majority of these were multiple choice questions that students could answer via their clicker. The questions were selected to emphasize an
important point or likely area of confusion in the reading. For example, the following is a question from the second lecture that we asked in order to emphasize the distinction between concurrency and parallelism:

“Concurrency” and “parallelism” are distinct terms which are often confused. Which of the following is NOT correct?

A. “Concurrency” means dealing with operations that are logically simultaneous
B. Concurrency issues can exist in serial systems
C. Parallelism requires splitting an operation into parts that can be performed simultaneously
D. Load balancing is a concern for parallelism more than concurrency
E. Race conditions are a concern for parallelism more than concurrency

The non-multiple choice questions asked students to apply something from the reading, such as by writing a fragment of pseudocode. These were still presented in the multistage approach of peer instruction, with students working on the question individually at first, then in small groups, and then all together.

III. ASSESSMENT AND DISCUSSION

Because this was the first time this course was offered, it is not possible directly compare what students learned through peer instruction with what they learned with more traditional instruction. Thus, we rely on attitudinal measures. At the end of the course, the students were asked to rank ways it could be taught. Twelve of the eighteen students answered this question. Figure 1 shows the results. Peer instruction is clearly preferred by most students over traditional options; 5 students ranked it highest vs. ≤ 3 for other options and 8 students put it in their top two favorite techniques vs. ≤ 5 for the others. On the same survey, the students were asked to rate the effectiveness of each component of the technique (first vote, discussion with neighbor, second vote, whole-class discussion, and reading ahead of class) on a 6 point scale. For each component, the mode was 6 and for all components except “first vote” at least three quarters (10/13) of the class gave a rating of more effective than not (4–6). The instructor also received higher scores than usual for his enthusiasm and teaching effectiveness on the official course evaluations, though those categories could be influenced by the course topic as well as the pedagogy used.

Although the majority of students seemed to prefer peer instruction to lecture, this was not true for all students. Figure 1 shows two students ranking peer instruction as their least favorite. There were also comments on both the specific survey on peer instruction and the official course evaluation indicating a preference for lecture. A colleague who has been using peer instruction for longer says that this is typical; students are used to lectures and not all of them appreciate being forced to participate more in class.

From student comments and subsequent reading, we have also learned some ways to improve our implementation of peer instruction. First of all, we should not have waited as long as we did for students to vote. A few stragglers sometimes greatly slowed the class without much benefit, particularly for the first vote on a question when it is not very important how the students vote. This is apparently a common error made by new adopters of peer instruction [6].

In addition, several students requested more content on the slides than just the questions themselves. Even with the answers marked, slides with just the questions were not useful as a reference. In addition, some questions would sometimes greatly slowed the class without much benefit, as long as we did for students to vote. A few stragglers also learned some ways to improve our implementation of peer instruction. First of all, we should not have waited as long as we did for students to vote. A few stragglers sometimes greatly slowed the class without much benefit, particularly for the first vote on a question when it is not very important how the students vote. This is apparently a common error made by new adopters of peer instruction [6].

REFERENCES


<table>
<thead>
<tr>
<th>Full peer instruction</th>
<th>most</th>
<th>least</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flipped class with clicker questions, but no peer discussion</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Powerpoint lectures</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Chalkboard lectures</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Mix of these</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 1. Student ranking of preferred instructional technique for the course. (n=12 from class of 18)