Screen-Sharing Concurrency

Rose Bohrer[1] [0000−0001−5201−9895]

Worcester Polytechnic Institute, Worcester MA 01609, USA rbohrer@wpi.edu

Abstract. We propose a lesson plan for concurrency which employs the screen-sharing features of video-conferencing software as a metaphor for interaction between concurrent software processes. The lesson plan is appropriate for use in a lesson for undergraduate students in the second year and above, such as in a systems programming or distributed systems course. It is particularly well-suited for online and hybrid classroom settings.

Keywords: Screens · Concurrency · Sharing.

1 Introduction

Concurrency is widely recognized as a key topic in a well-rounded undergraduate computer science curriculum. In the modern era, concurrency is widespread, from preemptive operating systems to web services to multi-core programs running on a single computer. It is unfortunate then that concurrent programming is often regarded as difficult to perform correctly and difficult to teach. This paper contributes nothing to the correctness of concurrent programs, but turns its attention instead to the teaching of concurrency.

The difficulty of teaching concurrency can be explained, in part, by combinatorial explosion: there are exponentially-many potential interleavings of the concurrent programs, which quickly make exhaustive case-by-case analysis impractical for even simple programs. In educational settings, it is impractical to explore every interleaving with students. Instead, teaching and programming both focus on higher-level programming abstractions (e.g., concurrency primitives such as mutexes and semaphores). These abstractions come with their own challenges. Not only do concurrency primitives remain subject to combinatorial explosion, but introducing abstractions introduces a layer of indirection in a student's understanding, such that one can easily forget whether they have explored all possible behaviors or fail to understand which concurrency primitives are truly serving to prevent a concurrency bug.

Traditional education overcomes this challenge with hands-on concurrency demonstrations, to provide students a direct experience of correct and incorrect system behaviors. Because concurrency entails multiple agents acting in concert, it is particularly well-suited to group activities in the classroom.

The rise of online and hybrid education, greatly accelerated by the COVID pandemic, has overturned much educational tradition, presenting both challenges and opportunities. Online instruction can be more accessible for everyone from

disabled people to returning students to students with lower socio-economic status. It can allow students to work on their own schedule, even when their schedule is busy. It can allow scales that are unprecedented in traditional classrooms. On the flip side, the rapid shift to online education during COVID has negatively affected learning outcomes. Online lectures can make students struggle to pay attention, especially when attending from a chaotic home environment. Pervasive pandemic anxiety reduces the capacity to form new memories and the loss of in-person social support networks removes an essential accountability mechanism and moral support. These negative impacts make it more essential than ever that instructors use modern online teaching technology to design activities that engage students while fostering social interaction among peers.

This paper proposes a lesson plan for concurrency based on hands-on interaction between students, built on a key observation about the Zoom videoconferencing application: screen-sharing and shared whiteboards constitute a medium for concurrent interaction.

2 Lesson Plan

The lesson consists of a series of group activities performed on Zoom, which first present the idea of concurrency bugs through demonstrations of increasing complexity, then present solutions.

2.1 Presenting the Problem

The problems of concurrency bugs are presented in the following stages.

Single Writer, Multiple Reader The instructor screen-shares the Zoom whiteboard and observes that it is a form of shared memory: every pixel stores digital data which are shared among all participants of the lecture. It is noted that the students are all engaging in an act of concurrency: all students are observing the instructor at the same time as one another. Students are cautioned that concurrency can be home to many bugs, and encouraged to keep an eye out for any bugs as the instructor performs the first activity: writing their name on the whiteboard.

The instructor observes that the exercise succeeded because a very specific mode of concurrency was undertaken: a single person wrote to the shared memory while everyone else read from it. The challenge comes when multiple people write at once.

Multiple Independent Writers To demonstrate the problem of multiple simultaneous writers, all students are invited to write their names on the whiteboard at the same time and share a laugh together as everyone writes on top of one another and produces an incomprehensible ball of scribbles.

Multiple Dependent Writers The instructor observes that concurrency is significantly harder when there are dependencies between the reads and writes of each process, i.e., when they must work together. The students are prompted to work together on a task: count the number of students in the class. To prevent students from cheating by merely looking at the participant list, they are told to show their work. If the students do not start writing anything, the instructor prompts them to use a system such as writing out the number of people who went so far or using tally marks. The bugs of each approach are then observed.

No Writers The concept of deadlock is presented by introducing a new rule: only write something after the person before you has written. Because the instructor (host) is at the top of the participants list, the first student never starts writing anything. After a long pause, the instructor indicates they were waiting on the person before them, then explains the concept of deadlocks occurring when multiple agents are waiting on one another in a cycle.

Dining Philosophers The next exercise emphasizes that more subtle deadlocks are possible. The students act out the Dining Philosophers problem together. A plate is drawn for each student and assigned to them, then a chopstick is drawn to each side. The students are told their mission of collecting two chopsticks at the same time. Depending on the order of operations, the students could succeed in their collective goal of getting one half of the students to complete their chopstick pair, or fail. Whichever happens, the students are instructed to repeat the exercise in a way that gets the opposite result. For example, if the exercise failed, they are now told that odd-numbered students should take chopsticks and even-numbered students should not.

2.2 Presenting the Solutions

The remainder of the lesson presents various strategies for resolving concurrency bugs.

Single Mutex To implement a single global mutex in Zoom, the following rule is instituted: one is only permitted to draw to the whiteboard while they are the one screen-sharing. Because the default is that only one person may share a screen at a time, without interruption, the screen share functions precisely as a mutex.

The Dining Philosophers exercise is repeated with the single global mutex. It results in a correct outcome but proceeds incredibly slowly. A tradeoff is highlighted: excessive concurrency leads to bugs, while insufficient concurrency results in performance loss. The goal then is to be as concurrent as possible without sacrificing correctness.

Multiple Mutexes A generic strategy is presented for working with multiple fine-grained locking: deadlock is avoided so long as locks are acquired and released in accordance with some lock ordering. For the next demonstration, the Dining Philosophers exercise is repeated but numbers are written by each chopstick to indicate their place in the lock order. It is observed that this demonstration proceeds far faster than the single-lock solution.

Data parallelism The attendance-counting exercise is repeated, but with coordination. Each student is told to draw a circle indicated their presence in the classroom, but they are told to arrange the circles in rows like the seats of a physical classroom. The instructor tells the leftmost member of each row to count their row and write the final count on the side. A single student is elected to wait until all row counts are received and compute a final sum.

The algorithm's correctness is explained: it exploits data parallelism to sum independent rows simultaneously. The fact that the data are independent is represented visually by the fact they take up different physical locations[1].

3 Barriers to Implementation

There are several barriers to implementation:

- Some of the exercises are designed to demonstrate the slow speed of sequential algorithms. On large class sizes, these algorithms may become so slow as to hinder learning rather than aid it. In these cases, students could be split into multiple breakout rooms to reduce group size.
- Some of the exercises require non-trivial setup, such as drawing spaces for every student. The author is not aware of any way to copy image files onto the whiteboard, which complicates preparation. The low-complexity solution is to assign a teaching assistant the menial work of drawing the setup for each activity, perhaps even in advance. An equally evil yet somehow more ethical solution is to exploit the capacity for typing text on the whiteboard. The instructor could, in principle, develop a custom font face whose glyphs contain the desired imagery for the exercises and type a carefully-engineered text to produce the desired image. At this writing, font size appears limited, in which case the desired image may need to be partitioned across a large number of glyphs. Though this approach is labor-intensive, it need only be performed once.
- Because the people in charge have more sense than me, I am not scheduled to teach any course in which this lesson would be relevant. Thus its implementation require at least one person read this paper. A surefire approach has been employed to this end: presenting the work at the most prestigious of computer science conferences.

[1] The parallel version of the attendance-counting exercise is taken from an in-person exercise in an offering of 15-150 taught by Dan Licata.

4 Future Work

Future work can be pursued along several directions, the most evident of which are exploiting additional Zoom features and implementing a wider range of concurrency concepts.

4.1 Zoom Features

The proposed exercises operate under the assumption that an ongoing screen share cannot be interrupted by a new screen share. Though this assumption is a reasonable model of the default Zoom settings, Zoom can also be configured to either (1) allow the host to interrupt an existing screen share or (2) allow any participant to interrupt an existing screen share. As the name suggests, interruption would be a powerful principle for exploration of concepts such as preemptive scheduling and even perhaps work-stealing scheduling.

The proposed exercises moreover assume that only a single screen share can be active at a time. Zoom can be configured to allow multiple screen shares at a time, which enables complex patterns of concurrent interaction. This permits a single reader to consume streaming data from multiple writers simultaneously and even enables multiple consumers of the same data to select which subset of the data (section of the screen) they wish to view. Immediate applications are clear, but non-immediate applications are potential.

A privacy feature of Zoom ensures that if the boundary of the shared window is overlapped by some other window, that portion is grayed out on the viewer's screen. This mechanism can be used to implement concurrent processes wherein some data are local to each agent while other data are shared.

The Zoom Chat window allows both broadcast and point-to-point communication, a power concurrency mechanism.

4.2 Concurrency Concepts

The present study primarily focuses on mutexes as a synchronization primitive through which concepts such as race conditions, deadlocks, lock ordering, and performance tradeoffs can be explored. This leaves much room for future work.

Firstly, additional concepts could be explored, including message-passing concurrency, scheduling, lock-free algorithms, and priority inversion.

Secondly, implementations of additional synchronization primitives might be pursued, such as semaphores, condition variables, and readers-writer locks.

5 Conclusion

College classes are often boring, and online education is an excellent way to make them even more boring. We should sometimes attempt to make them less boring. This article is a humble attempt to do so.

Acknowledgement and Call to Action

Assistant professor life is busy, but there's a reason I still made the time to write a SIGBOVIK paper this year. It's because when I changed my name, the SIGBOVIK editors were far more thorough and helpful in updating my old publications than in any of the half-dozen publishers I contacted regarding my name change.

Name change processes have improved somewhat in recent years, due in large part to the advocacy of the Trans Name Change Policy Working Group and the willingness of the Committee on Publication Ethics to issue relevant guidelines. However, much remains to be done. At the time of writing, no major publisher corrects names in citations (only SIGBOVIK does), and many take excessively long to perform corrections, with the ACM in particular routinely taking over half a year for small changes. CMU was particularly extreme. When I updated the name on my dissertation, they required my court order (which itself takes months and costs hundreds of dollars) and attempting to require the written approval of a transphobe before finally backing down when pushed. I wanted to contribute to making this SIGBOVIK a fun one because the unpaid volunteers running it have been more supportive than multi-billion-dollar publishers, professional organizations, and institutions that profit from my work.

At the same time, I wish to close with a call to action for cisgender academics. Name changes may have been annoying and imperfect, but the truth is that they're nothing compared to the daily dangers of being a transgender person in this country. In 2023, pundits publicly call for our complete eradication from public life to loud applause and then get invited to speak on college campuses. As I write, Oklahoma is on track to ban insurance coverage of all transition care, and trans people across the country wonder: when this legislation goes national, where will we have left to run? For many people, the answer will be nowhere.

I have been harassed in professional setting. I have been called slurs on the street in a supposedly progressive city for the crime of commuting to campus. But I call myself lucky because I even made it this far. Few trans academics make it to the faculty stage. While we may have to trudge through being the only one who looks like us in the room, the effect is even greater on trans students, who are left with zero advocates from within their community among the leadership of their community. It is no wonder then that trans students face all these problems tenfold, with the attendant impacts on everything from mental health to academic and life success.

So the next time the trans person next to you asks you stand up for them, do it.