On the Use of Model Checking for Remote Instruction

Thomas Anderson
University of Washington
tom@cs.washington.edu

Ellis Michael
University of Washington
emichael@cs.washington.edu

1 INTRODUCTION

In prior work, the authors developed a set of online labs for teaching undergraduate distributed systems, called DSLabs [6]. The labs are ambitious – over a ten week quarter at our university, each pair of students is asked to design and implement a highly-available, fault tolerant, linearizable, sharded key-value store with dynamic load balancing and multi-key transactions. Many students describe completing the project as among the hardest academic challenges they faced in college.

Most relevant to online teaching of computer networking topics, DSLabs uses exhaustive model checking to give students real-time feedback on the correctness of their code, without constraining the design space of student solutions. Like networking protocols, distributed systems are highly concurrent, with behavior that depends on the precise ordering of events (message drops, message deliveries, timeouts, machine outages, and network partitions). A faulty program will often exhibit errors on only some event sequences, complicating testing. By systematically exploring all possible event sequences, model checking can uncover otherwise hard to find bugs. To allow for efficient state space reduction, we limit student protocols to deterministic handlers for event delivery and timeouts, invoked (repeatedly, in different orderings) under the control of the DSLabs framework; students can request randomness from the framework if needed.

Our aim is to provide students the tools to self-diagnose their designs so that conceptual errors can be caught early, motivating students who need it to ask for help. The DSLabs testing framework uses several novel techniques to narrow and shape the model checking search space, enabling it to find bugs more rapidly than other approaches. Instead of searching exhaustively up to a defined event depth, our framework can be configured to search through a series of waypoints (such as a series of outages or partitions that are known to be troublesome) before continuing the search. For Paxos, for example, we have a test that exercises majority intersection. For distance vector routing, one could insert a specific link weight update after convergence, using exhaustive search to ensure that no routing loops occur regardless of the sequence of events after the update. Each assignment includes a few dozen tests of this sort.

To be practical, we constrain the tests for each assignment to complete within a few minutes on a department server or student laptop.[1] (In industrial settings, it is common to run model checking for days across a large cluster.) Even so, the DSLabs framework can find bugs that manifest after several dozen timing-dependent concurrent events. In most cases, these bugs would be difficult for even a highly sophisticated teaching assistant to find by manual code inspection – an approach totally impractical to do at scale given that the course is taken by about 250 students per year at our university. We periodically validate our tests by running more exhaustive searches of student code offline; if we find new, uncaught bugs, we modify the tests to be more general in subsequent offerings.

When the model checker does uncover a violation of a safety property, it produces a concrete system trace describing which exact sequence of events, out of potentially millions of possible sequences, will reproduce the bug. This trace is either minimal by construction (i.e., because it was found via breadth-first search) or is run through a trace-shortening function. While these traces do not describe *why* the bug manifests, students often find it intuitive to reason about correctness by counterexample. Understanding which specific sequences of events lead to unexpected behavior is often the first step to students obtaining a deeper understanding of the problem specification. For concurrent code especially, where any change may perturb the specific execution sequence taken by a test suite, model checking can help remove the uncertainty of whether a bug is truly fixed.

The pandemic provides a natural experiment for the efficacy of automated model checking and self-diagnosis for remote learning of advanced computer science topics. The University of Washington runs on the quarter system, with the winter quarter starting immediately after New Year's Day; each quarter has ten weeks of instruction, a week of exams/final projects, and a week break between terms. Thus, spring quarter starts twelve weeks after the beginning of winter quarter. DSLabs was introduced at the undergraduate level in the spring 2017, offered again in spring 2018 and 2019, and due to increased student demand, offered twice (once in winter and once in spring) for 2020.

The timing of the Washington virus outbreak and subsequent closure of the University meant that the first eight weeks of winter quarter 2020 instruction proceeded normally, with a ninth week of declining physical class attendance. For the final week of the quarter, the campus closed and lectures (in this class and many others) were cancelled. For spring quarter 2020, the same class was offered online in the context where the entire campus was operated remotely, including discussion sections and office hours. For the previous live offerings, lecture videotapes were collected automatically and made available to students, allowing physical lecture attendance to be optional.

[1] One of our students created a meme to represent the fans on their PC while the tests were running [9].

2 RELATED WORK

Prior studies of live versus online learning have shown mixed results. At the high school level, a recent study looked at student math progress using an online learning system in place prior to school shutdown. The tool was originally designed to be used as part of mixed live and online instruction, but was converted to completely online use after the shutdown. Nationwide, students had uneven but generally lower rates of progress, with notably worse outcomes for lower income school systems [7].

Studies of online learning at the college level have been generally limited to either freshman classes [1, 3–5] or for-profit education [2]. Methodologies and experimental design differ among these studies, including varying the amount of live instruction in mixed settings, as well as comparing live versus entirely online education. In general, the results paint a consistent picture: students in the top third in terms of entering GPA see only a small, often statistically insignificant impairment in learning outcomes, from learning online versus live education. However, students with worse academic preparation and skills often experience significantly worse outcomes – a third to a half a grade point lower on common exams relative to what they would be expected to achieve had the class been taught live.

The college-level studies described above generally used the same instructional methodology for online versus live lectures, leaving open the question about whether innovative instructional techniques can close the gap imposed by online learning, particularly for struggling students. Other work has shown the promise of better tools for self-diagnosis. For example, a study of randomly assigned students in a freshman psychology course at the University of Texas at Austin showed that low income students who received customized online quizzes at the beginning of each lecture did on average a half a grade point better than those who did not [8], with significant spill-over effects for other classes those students were taking at the same time. Thus, preparation, study habits, social context, and supporting software all should be expected to have a significant effect on student outcomes with online learning.

3 RESULTS

We compare the cumulative distribution of student performance on the lab assignments across the most recent four offerings of UW's distributed systems class in Figure 1. We caution there are a number of confounding factors that limit the comparison across terms; we describe those factors next.

Spring 2020 was entirely online, taught by one of the authors. Winter 2020 was live for the portions of the class most relevant to the labs, taught by a colleague. Spring 2018 and 2019 were live, taught by each of the two authors. About half of the teaching assistants were the same between Winter and Spring 2020. Even when personnel is a constant, however, teaching effectiveness tends to improve with experience.

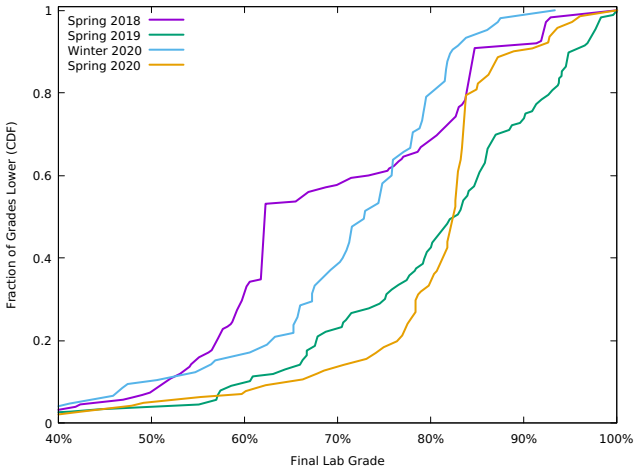

Figure 1: Cumulative distribution of DSLabs project grades across four offerings of distributed systems at the University of Washington-Seattle. Spring 2020 was taught entirely online.

We focus on the lab portion of the grade as the class has no final, and problem sets varied from year to year. The lab comprised the majority of the student grade in all quarters. Some improvements were made to the labs after the 2018 and 2019 offerings; in particular, some tests were added as we learned more about places that the students had difficulty. We normalize results to the maximum possible score to partially correct for this effect.

The course is not curved, and so students can generally determine in advance what grade they can expect; however, the threshold for a perfect score can vary depending on the offering and instructor. Students could complete additional work for extra credit. In particular, we made parts of the lab optional for Spring 2018 as we were learning how to teach the material in the labs. The expectations in Winter and Spring 2020 were to complete almost all of the lab (to the same degree). For Spring 2019, students were asked to complete the entire lab.

The class is an elective, taken primarily by computer science majors (other students may petition), and most commonly by students in their final year of study. Since the future closure for spring quarter was unforeseeable by (most) students, the choice of Winter versus Spring 2020 registration was independent of delivery method. The two instructors in Winter and Spring received similarly high teaching ratings.

Students in Spring 2020 were made aware that the course would be online, and students could choose to de-register. However, for most students, the course would not be offered again prior to their graduation. Pre-registration was completed shortly before virus spread was observed in Washington state. About 15-20% of students dropped the class between that point and the end of the first week of classes. That is somewhat higher than normal. Some students opted to register but with a lighter than average course load given that all classes would be online.

4 CONCLUSIONS

The results qualitatively support the hypothesis that educational tools supporting self-diagnosis can be an effective bridge to normalize achievement outcomes for online versus live instruction, especially for lower achieving students. Fewer students than expected did extra credit in the online version of the class, but contrary to prior studies, students in the lower half of the class did as well or better than in previous terms. The extreme lower tail of the curve for the various offerings was similar, well within the expected variation for the number of students in the class. Course expectations and the experience and skill of the teaching assistants had a larger effect on student outcomes than whether the course was offered live versus online, at least in this setting of an advanced elective computer science course with strong software tools to support online student learning.

REFERENCES

[1] W. T. Alpert, K. A. Couch, and O. R. Harmon. A Randomized Assessment of Online Learning. *American Economic Review*, 106(5):378–82, 2016.

[2] E. P. Bettinger, L. Fox, S. Loeb, and E. S. Taylor. Virtual Classrooms: How Online College Courses Affect Student Success. *American Economic Review*, 107(9):2855–2875, 2017.

[3] W. G. Bowen, M. M. Chingos, K. A. Lack, and T. I. Nygren. Interactive Learning Online at Public Universities: Evidence from a Six-Campus Randomized Trial. *Journal of Policy Analysis and Management*, 33(1):94–111, 2014.

[4] D. Figlio, M. Rush, and L. Yin. Is It Live or Is It Internet? Experimental Estimates of the Effects of Online Instruction on Student Learning. *Journal of Labor Economics*, 31(4):763–84, 2013.

[5] T. Joyce, S. Crockett, D. Jaeger, O. Altindag, and S. O'Connell. Does Classroom Time Matter? *Economics of Education Review*, (46):64–77, 2015.

[6] E. Michael, D. Woos, T. E. Anderson, M. D. Ernst, and Z. Tatlock. Teaching Rigorous Distributed Systems With Efficient Model Checking. In *Proceedings of the Fourteenth EuroSys Conference 2019, Dresden, Germany, March 25-28, 2019*, pages 32:1–32:15, 2019.

[7] OpportunityInsights. Track the Recovery: https://tracktherecovery.org, 2020.

[8] J. Pennebaker, S. Gosling, and J. Ferrell. daily online testing in large classes: Boosting college performance while reducing achievement gaps. 2013.

[9] A. Student. Literally every fan in my PC: https://imgur.com/BbqULTc, 2020.