Teaching Statement

Malte Schwarzkopf

My teaching approach reflects my hands-on research philosophy of building practical computer systems. I see it as a key part of my future job to teach new generations of students to build and understand systems. In particular, I am enthusiastic to teach classes on distributed systems, operating systems, networking, and security, as well as seminar classes on my research interests, such as distributed systems for data center computing.

Throughout my Ph.D. at the University of Cambridge and my post-doctoral work at MIT, I have continuously taught and mentored students. These activities are fulfilling in their own right, and I appreciate the valuable insights that interaction with students provides for my own research. For example, preparing for class often makes me appreciate how a system I am teaching about relates to my current project; a class project sometimes sparks a new research project; and advising a student on experiment design can help clarify my own performance hypotheses.

I believe that understanding problems of distributed system design requires first-hand experience with them. My goal is therefore to instill an appreciation for the challenges and techniques of good system design in my students through hands-on experience. Consider the implementation of a complex RPC protocol, which can easily have subtle bugs that cause deadlock—*e.g.*, if the callee issues further RPCs that block. Students best understand the strategies to debug such situations and the anti-patterns to avoid if they have seen and written deadlocking code themselves. My teaching combines two-way interaction in class with such hands-on experiences to give students the knowledge and confidence to engineer their own systems in the future, either in research or in industry.

Classroom teaching. I have taught in both classroom and small-group settings. Most recently, I co-taught MIT's graduate distributed systems class (6.824) with Robert Morris in spring 2018; prior to that, I lectured part of the undergraduate distributed systems course at the University of Cambridge in 2014. In teaching 6.824, I emphasized case studies of real distributed systems and how they are used in industry, and provided hands-on exercises that helped students develop a deeper understanding of the principles behind these systems. In the 6.824 labs, students implement the Raft consensus protocol and a strongly-consistent key-value store built atop Raft. We provided students with skeleton code and test cases that simulate failures and an unreliable network, and asked them to fill in key components (*e.g.*, leader election and replication). The goal is not merely to produce code, but rather to develop an understanding of how to transform an abstract protocol specification into an actual distributed system. This experience prepared students for an independent project at the end of the class, in which they designed and implemented their own distributed system.

I also introduced similar hands-on elements in my prior teaching at the University of Cambridge as a small-group "supervisor", a position comparable to a U.S.-style TA role (albeit with groups of only 2–3 students). When teaching the freshman Operating Systems class, I always asked students to complete some practical exercises that made the material tangible. These exercises involved writing a Linux kernel module, making small changes to the Linux kernel, and writing their own shell. While the short class duration and strong exam focus of the Cambridge system make lab-style assignments difficult, my students appreciated the practical exercises, often put significant effort into them, and related to the material more deeply than if we had just discussed concepts on a whiteboard.

In addition to practical experience, I believe that two-way interaction is crucial to a great class. At Cambridge, I encouraged my students to submit their own questions ahead of our supervision meetings, and to drive the discussion when we met. At MIT, 6.824 students submit an answer to a set question on the material and a question of their own before each lecture. I spent a few hours on the evenings before my lectures reading all questions and responding with answers to several dozen student questions. Students greatly appreciate these answers, but their questions and my answers also help me anticipate and address confusions in class, and often provide interesting angles on the material. I will use similar techniques to foster student engagement in future classes I teach.

Student mentoring. I have mentored undergraduate and graduate students in several capacities. At MIT, I have mentored three undergraduate researchers and help advise two Ph.D. students—Jon Gjengset and Jonathan Behrens—with whom I work on the Noria data-flow computing system. Jon's work on high-performance data-flow in Noria won the SOSP 2017 Graduate Student Research Competition, and was awarded the runner-up prize in the graduate category of the ACM-wide 2018 Student Research Competition, under my mentorship.

I see my primary role as an advisor in helping students identify and systematically attack a research problem that they enjoy working on. Once a student has developed a sense of ownership and they feel confident seeking answers to their own research questions, the advising relationship becomes a fruitful dialogue in which I often learn as much as the student does. For example, when I advised Lara Timbó Araújo for her M.Eng. dissertation, we started out with a vague idea of using data-flow to enforce security policies between different user sessions in a web application. After we identified a challenge (space explosion with many users) and an initial idea of how to address it (sharing data-flow sub-graphs if possible), Lara would initially ask me for very specific advice on what to do next. Over time, however, our interaction flipped, with Lara taking control and evolving the idea increasingly independently, presenting her own insights to me. When she finished her M.Eng., Lara had a project to be proud of, but had also laid the ground for a new research system. I am now extending Lara's work with a new Ph.D. student and two undergraduate researchers, while Lara plans to apply to graduate school after a year in industry.

At Cambridge, I regularly advised undergraduate students' final-year bachelor thesis projects. These projects need to be standalone engineering efforts, but can have a research aspect to them. Of the 13 projects I advised, ten received a first-class grade (awarded to 25% of the class) and several won departmental awards. My approach to these projects was to identify an idea that excited the student without overly rigidly defining the project at the outset, and to then allow an interesting research insight to emerge. A good example of this is Adam Gleave's project, which investigated whether cluster schedulers built around constraint solvers can make low-latency decisions on large clusters. Realizing Adam's strength in algorithmic thinking, I suggested to him to consider approximate and incremental solutions to the min-cost, max-flow constraint satisfaction problem. Adam implemented several solvers, finding that the constraints typical for cluster scheduling problems produce poor approximate solutions, and that incrementalization alone cannot reliably achieve low latency. However, Adam realized that another algorithm with seemingly disqualifying worst-case complexity is well-suited to the types of constraints cluster schedulers generate. His insight formed a key part of the OSDI 2016 Firmament paper, with Adam as one of the co-authors. It also excited Adam for research, and he is now a graduate student in machine learning at UC Berkeley.

Finally, I gained insight into broader undergraduate student mentoring when I served as a sabbatical replacement Director of Studies (DoS) for St John's College, Cambridge in 2013–14 (a position roughly equivalent to a U.S. undergraduate faculty advisor). I guided students in their class choices, supervised their academic progress, and handled admissions—but most importantly, the DoS role gave me an opportunity to offer students new intellectual perspectives and influence their learning environment. For example, I introduced a bi-weekly talk series on computer science research, and asked students to present their class and side projects, as well as their summer internship work. This gave an equitable, structured platform to all students and—combined with my encouraging of a positive environment—especially benefited female and minority students, who otherwise might have been less confident to present their work. To my great pleasure, my students' exam performance exceeded the long-term average for St John's College, and two of my five final-year students ultimately went on to graduate school (at Oxford and UC Berkeley).

In my future teaching and advising, I will build on these experiences to help students succeed as engineers and researchers. I hope to help them discover how much fun computer systems research can be, while myself no doubt continuously learning from their ideas.