Teaching Statement

Yang Zhou

I greatly enjoy the rewards of teaching and mentoring students. For me, the rewards consist of two significant parts: (1) the pride and fulfillment when my teaching helps students carry out their studies smoothly and when my mentored students grow into independent researchers, and (2) the interesting future research directions inspired or confirmed during teaching and mentoring. Driven by these rewards, I have taught as a teaching assistant and as a small-group "supervisor", and mentored four undergraduates and five junior PhD students in their research. Based on my research background, I am qualified to teach undergraduate courses of computer networks, operating systems, distributed systems, and algorithms and data structures, and graduate courses of data center networking and dataplane operating systems (detailed later).

Mentoring Experience and Methodology

I have mentored four undergraduates on system research, and informally mentored five junior PhDs on their research ideas, internship applications, and study experience at Harvard. Among the four undergraduates, one (Zezhou Wang) published an NSDI'23 paper with me and went to University of Washington (UW) as a system PhD; two of them (Xingyu Xiang and Matt Kiley) co-authored an NSDI'24 submission with me, and are about to apply for system PhDs as well as the rest one. Such mentoring brings me enormous pride, e.g., seeing Zezhou gets into the UW PhD program. It inspires my future research—working with Zezhou on eBPF sparks two follow-up projects: one has become the NSDI'24 submission, and another is showing promising results. Below I summarize my mentoring methodology:

- *Building students' confidence.* It is well-known that confidence is crucial for students, but how to build their confidence is challenging. One way I find helpful is respecting students' thoughts by giving them enough freedom to try their thoughts while keeping an eye on the big agendas and goals. Another way is connecting them to experts upon entering a new field, avoiding the steep learning curves overwhelming or destroying their confidence. The experts, who could be the mentors themselves, would point out the proper materials or steps for quick ramp-ups.
- *Encouraging students to form their own opinions and tastes.* I encourage and anticipate students to form their own opinions about systems, develop their own tastes on promising research problems, and stick with them. I do not worry too much about if students' opinions/tastes are wrong, as once they go deep into specific directions they believe, they will learn extensive experiences and insights to refine their previous opinions/tastes.
- *Collaborating widely.* Wide collaboration across industry and academia is especially beneficial for practical system research, and mentors should play the important role in connecting students with proper researchers in the wild. For example, my fault-tolerant far memory project Carbink was collaborated with Google via my co-advisor's connections, and then inspired by Google's desire for high availability. However, collaborating with industry usually requires teasing out real research challenges, while not being misled by massive engineering details; advisors should leverage their experience to help students (especially junior PhDs) navigate efficiently in this space. For another example, my eBPF-for-Paxos project Electrode would not be possible without the collaboration with Sowmya Dharanipragada who is a distributed system PhD at Cornell. Going forward, I would like to expand collaborations to theory, machine learning, architecture, programming languages, etc.

Teaching Experience and Philosophy

System course teaching: I was the teaching assistant (TA) for a computer system course, the Harvard CS145 Networking at Scale, along with an undergraduate TA. This course features eight P4-switch related projects, three of which are designed and developed by me including detailed guides and skeleton code. I held three one-hour sections covering network programming, background knowledge for projects, and handy tools for developing and debugging. Other duties include holding weekly office hours, answering students' questions on forums, and grading projects. In addition to TA, I also had a guest lecture experience at UC Berkeley on far memory techniques in data centers, mainly facing junior graduate students from architecture areas. I started from common and accessible facts like resource utilization and

DRAM prices, then explained why data center operators have an interest in far memory, and finally discussed my work in this space.

Algorithm course teaching: I was the small-group supervisor for the Algorithm Design and Analysis course at Peking University as an undergraduate. This role requires supervising around 14 students in small classes, giving recitations, teaching advanced algorithms and data structures, preparing new problem sets and quizzes, and grading, all on a weekly basis. I extensively introduced non-textbook topics related to my undergraduate research of probabilistic data structures. Although time-consuming, being such a supervisor is truly gratifying, especially when students understand my research and try various optimizations as their final course projects. One student (Yicheng Jin) in my small class is now pursuing a computer science PhD at Duke University.

Introductory teaching: I taught non-CS audiences about the Internet from a computer science perspective during the English Language Program at Harvard. It was a slightly difficult yet fun experience especially when I told the audience that Internet data is transmitted in small packets: they were shocked and immediately asked why, and then I gave them detailed yet understandable explanations until they grasped the design philosophy behind it. This experience gave me a good sense of how to teach introductory courses in the future. Below I summarize my teaching philosophy:

- *Building safe and inclusive environments.* Students in the same class usually have different prior knowledge; thus it is important to create safe and inclusive environments to make students feel they are welcome to ask both the simplest questions and challenging ones. I got such first-hand experiences when I took my co-advisor James Mickens' CS263 System Security course: it has the most open class environment I have ever seen because of James' unique humor, and students ask so many interesting questions during the class. As a result, I personally learned so much security knowledge, though my research is on networked systems.
- *Focusing on hands-on experiences.* I believe the best way to learn computer systems is through reading, running, debugging, and hacking well-written codebases in a hands-on manner. My personal experience in learning dataplane operating systems exactly follows this pattern: after reading relevant papers, I could not understand how specific designs get implemented and contributed to the final performance; then I decided to read the codebase of a dataplane OS called Caladan [1], and run and debug it; finally, I built my own research prototype atop it. After the process, my understanding of dataplane OSes became much clearer, and I gradually began appreciating the merits of various designs in this space. For future system courses I teach, I would like to incorporate well-written teaching systems, such as the WeensyOS [2], into my agenda to help students gain hands-on experiences.
- *Promoting critical thinking on the pros and cons of techniques.* I learned this from the Harvard CS260r Projects and Close Readings in Software Systems—Serverless Computing by Eddie Kohler, where he discussed serverless computing research from a traditional system research perspective. He showed impressive critical thinking on the pros and cons of serverless computing, and helped us grasp the real novel components of this paradigm without deifying any new terms. I plan to apply a similar philosophy to my teaching, encouraging students to critically think about new techniques around us, such as the emerging LLM techniques.

Course plans: In addition to the aforementioned undergraduate courses based on textbook knowledge, I would like to hold two advanced graduate courses and a seminar course based on my research:

- *Data center networking:* I will discuss how modern data centers design and build high-performance network fabrics including topology, routing, congestion control, fault tolerance, load balancing, etc.
- *Dataplane operating systems:* I will discuss how the OS evolves to keep up with the fast hardware in data centers, including user-space networking, efficient threading, light-weight isolation, etc.
- *System seminar course:* I will invite a broad set of system researchers from both academia and industry to give talks on various system research topics, and foster potential collaborations with students.

References

[1] The Caladan authors. Caladan opensource. https://github.com/shenango/caladan.

[2] Eddie Kohler. Harvard CS61 Systems Programming and Machine Organization (2023): WeensyOS. https://cs61.seas.harvard.edu/site/2023/WeensyOS/.