Teaching and research complement each other. I like to polish the research by teaching in the classroom, while augmenting the teaching materials by active research.

I have a strong passion for helping students who are dedicated, but not necessarily performing well in class. Though I am excited to work with high achieving students, it is also challenging and engaging to interact with students who find the material challenging. These are the students with whom I give spontaneous lectures that distill class material to its core concepts. The more I work with these students, the more I find that identifying the core of an idea and connecting the ideas together in a big picture is a key skill that young students need to learn to be successful in class and beyond. I have been teaching following two classes every year since 2012:

○ **Introduction to Computer Systems (ICS).** This is a two-semester introductory undergraduate class on computer systems. It serves as the foundation for later system classes such as operating systems, compilers, computer architecture, computer networks, concurrent programming, among others. This class includes topics ranging from assembly code, optimizing compilers, computer arithmetic, memory organization and management, networking technology and protocols, and concurrent computation. This class is recognized as an outstanding class of Fudan University.

○ **Performance Engineering of Software Systems.** The free performance from Moore’s Law is going to end because of power wall, heat wall and memory wall. A programmer has to be a performance engineer since performance is the currency in the world of computing, which can be traded for new functionalities, such as scalability, security, reliability, to name a few. This hands-on project-based introduction to building scalable and portable high-performance software systems covers performance engineering techniques such as performance analyses, algorithmic techniques for high performance, instruction-level optimizations, cache and memory hierarchy optimization, and parallel programming.

**Graduate level course on “Parallel Programming System”:** Given the opportunity, I would like to design a graduate-level course on parallel programming systems building on the aforementioned undergraduate “Performance Engineering of Software Systems” course. This course will cover both high-level programming models, memory abstractions, algorithms and data structures with an emphasis on cache efficiency, performance metrics (time, space, and caching), imperative and functional parallel programming languages like Cilk and NESL, low-level provably efficient runtime systems (how to map multithreaded computation to a $P$-processor system), synchronization and race detection (conflict detecting and avoiding). The course would include a large term project comprised of several smaller hands-on projects designed to give insights into and increase understanding of the key concepts learned in the class. By organizing smaller projects into a final large one, students will gain confidence in building small to medium sized parallel programming systems. Moreover, since future project components are based on past components, students will revisit old concepts as they re-implement old parts of their projects.