

Teaching Philosophy

Daniel Nurmi

When I was actively taking classes as an undergraduate, one of my biggest concerns was that project courses typically lacked a fundamentally challenging problem; the challenge was typically in learning the details of tools used to solve some problem with an obvious solution, but not in finding a unique solution to the problem itself. My personal approach attempts to remedy this deficiency, since I believe that providing a fundamentally interesting and challenging problem motivates students to put forth the effort it takes to learn the details voluntarily. For example, I was given the rare opportunity (for a graduate student) to personally create and teach a sophomore level class at UCSB entitled 'C, C++, and UNIX programming' in the Fall of 2004. I designed the class to be fundamentally project based, but wove the theme of 'video game programming' throughout all of the projects. We used standard texts (Kernighan and Ritchie, Stevenson, Stroustrup) and my lectures were based around an interactive lecture style, but the projects I designed were at once extremely challenging and each resulted in a finished, graphical video game. The student's response was incredibly positive, and I was both shocked and impressed with the quality and variety of the student projects. The main lesson I learned from the experience is the foundation of my teaching philosophy; a sufficiently interesting and challenging problem leads computer science students to excel. I intend to carry this lesson forward as I am given the opportunities to develop more educational experience, particularly in project based systems courses.

My specific teaching interests fall into two categories: challenging fundamental systems courses with a focus on parallel/distributed architectures and programming, and the phenomenological study of computer systems. With the rapid growth of increasingly parallel computational architectures in the mainstream (many-core, multi-core, Amazon EC2, Google, etc), I am very excited to be given the opportunity to teach undergraduate courses that not only reveal the existence of these architectures, but explore and experiment with their construction, deployment, and most importantly their programming. When combined with strong course work in fundamental operating systems, programming and scientific computing, such a course will generate a student who is uniquely prepared for our increasingly distributed computational world.

My second teaching interest stems from the observation that modern large scale computer systems are so incredibly complex that traditional parametric modeling techniques are difficult to employ when the goal is to produce on-line

system software. In my own research, we have found that non-parametric, empirical methodologies can perform as well or better than more standard modeling techniques, and lend themselves to use in practice due to their relative computational simplicity. Thus, I believe that a new approach is required in order to improve our understanding of modern computational systems. To translate this perspective into an interesting course, I would propose a project-based class around the idea of a computer system 'laboratory' where I would either instrument real systems (EC2, PlanetLab, TeraGrid, etc) or create simulation environments that need to be monitored with careful attention being placed on gathering clean, unbiased data. With this data, we would perform a series of investigative tests to analyze and model the data using a variety of statistical techniques, and then attempt to design both simple prediction systems and mathematical models based on the analysis. The students would gain a unique perspective from the class, and also learn many applied statistics techniques, knowledge that is becoming increasingly valuable to the computer scientist.

My experience, ranging from designing and building some of the fastest super-computers in the world when I worked at Argonne National Labs to my current research efforts in the analysis and mitigation of large scale super-computer system dynamism has given me a breadth of experience that I enjoy sharing with students who are interested in programming and studying modern parallel and distributed computer systems.