NSF Workshop Report: Designing Tools and Curricula for Undergraduate Courses in Distributed Systems

July 8, 2012
Boston, Massachusetts

Principal Investigator: Jeannie Albrecht, Williams College
NSF Program/Division: CISE/CNS

Abstract

Distributed applications have become a core component of the Internet's infrastructure. However, many undergraduate curricula do not offer courses that focus on the design and implementation of distributed systems. As a result, undergraduates are not as prepared as they should be for graduate study or careers in industry. Historically, the problem has often been caused by a lack of resources, since many schools do not have the computing infrastructure needed to experiment with distributed systems. However, with the growing availability and accessibility of academic and industrial distributed computing platforms, this is no longer true. Even colleges with limited on-campus computing facilities now have the ability to experiment with large-scale systems using state-of-the-art networking technologies. This workshop focused on developing and disseminating new tools and curricula for undergraduate courses in distributed systems and computer networks that leverage the resources available in publicly-accessible testbeds. The 31 attendees came from a variety of backgrounds, including top-tier research universities, liberal arts colleges, and industry. This report summarizes the results of the main discussions and presentations from the workshop.

1 Workshop Vision and Motivation

As the number of Internet users continues to rise, Internet-based companies, such as Google and Amazon, are leveraging the aggregate computing power and robustness of distributed systems to satisfy the demands of their users. However, although distributed systems offer many advantages over non-distributed systems and applications, such as increased aggregate computing power and better resilience to failure, they also introduce many new challenges to developers. Configuring and maintaining a distributed set of computers for hosting an application is a tedious task. Detecting and recovering from bugs in applications that are running on hundreds of machines potentially spread around the world is much more challenging than debugging code locally. These challenges are overwhelming to developers without prior programming experience in distributed environments.

Since distributed computing is quickly becoming the de facto way to accomplish large-scale tasks on the Internet, one would think that the skills required to design, implement, and evaluate distributed systems would be available to students at the college and university level. Unfortunately this is frequently not the case, since many colleges and universities do not offer undergraduate courses in distributed systems. Particularly at small colleges, the problem is often due to a lack of computing resources. Without access to a dedicated computing cluster that permits students to run experimental code on distributed resources, it is difficult to expose students to the challenges of implementing and evaluating distributed systems. At large universities, although computing clusters are present, they are typically reserved for research purposes and are not readily available for use in the classroom. As a result, undergraduates are not receiving the training necessary to become good programmers in distributed environments. Further, they are not being exposed to current trends and topics in systems research, and thus many are not considering graduate school as a viable option after graduation.

Given the increasing popularity, accessibility, and maturity of publicly-available experimental testbeds, such as GENI [8], XSEDE [21], Open Science Grid [14], and Open Cirrus [13], as well as low cost industry-backed options such as Amazon EC2 [2], Microsoft Azure [11], and Google AppEngine [9], educators are no longer restricted to using only on-campus resources for assignments. Using the resources available in these platforms, even students at small colleges with minimal local computing infrastructure can gain hands-on experience with large-scale distributed systems. The technological richness that was previously only available to a limited number of undergraduates at top-tier research schools is now widely available. One of the key goals of this workshop was to redefine systems education at the undergraduate level by taking advantage of these distributed computing testbeds.

The first step to redefining undergraduate systems education is to inform educators at a broad cross-section of colleges and universities about the resources available to them and their students. Thus another goal of this workshop was to develop a set of materials for educators, starting with how to access and use the aforementioned testbeds and platforms. In addition, sample assignments with varying difficulty and other course-related material that leverages these resources were discussed. This workshop brought together users and developers of publicly-available, shared testbeds who share a common goal of lowering the entry barrier to large-scale, cutting-edge systems research and introducing students to distributed systems at early stages of their careers.

2 Summary of Events

The workshop was held in conjunction with the fourteenth GENI Engineering Conference (GEC) in Boston, MA on July 8, 2012. Registration and accommodations for participants was handled jointly with the GEC. The workshop consisted primarily of presentations from researchers and educators, followed by a question and answer/discussion session. Table 1 illustrates the workshop agenda, including the speakers for each session. There is also a workshop website that displays this information, as well as slides from each speaker. It is available at `http://www.cs.williams.edu/~jeannie/nsf-workshop`.

The following list includes the people who attended the workshop and their affiliations. Speakers are shown in boldface.

- Jay Aikat (UNC)
- **Jeannie Albrecht** (Williams College)
- **Anish Arora** (The Ohio State)
- **Mark Berman** (BBN/GPO)
- **Ethan Blanton** (Purdue)
- **Justin Cappos** (NYU Poly)
- **Jeff Chase** (Duke)
- Maureen Doyle (Northern Kentucky)
- Chip Elliott (BBN/GPO)
- **Sonia Fahmy** (Purdue)
- **Zongming Fei** (Kentucky)
- **Armando Fox** (UC Berkeley)
- Kevin Jeffay (UNC)
- Scott Kaplan (Amherst College)
- Kenny Katzgrau (Mozilla Ignite)
- Maifi Khan (UConn)
- Chip Killian (Purdue)
- Geoff Kuenning (Harvey Mudd College)
- **Jelena Mirkovic** (USC/ISI)
- Tom Murtagh (Williams College)
- **Tia Newhall** (Swarthmore College)
- Graciela Perera (Youngstown State Univ.)
- Rajesh Prasad (Saint Anselm College)
- **Sushil Prasad** (Georgia State University)
- **Joel Sommers** (Colgate University)
- Sara Sprenkle (Washington and Lee)
- Vic Thomas (BBN/GPO)
- Kevin Walsh (Holy Cross)
- Bing Wang (UConn)
- **Gary Wong** (Utah)
- Kaiqi Xiong (RIT)

Time Slot	Event/Description
8:00 – 8:30	Continental Breakfast and Registration
8:30 – 9:00	Opening Remarks – Jeannie Albrecht (Williams College) – Keith Marzullo (NSF) – Jeff Forbes (NSF)
9:00 – 10:30	Session 1: Platforms (Getting access, managing student accounts, etc.) – Justin Cappos (NYU Poly) - Seattle framework [18] – Gary Wong (Utah) - ProtoGENI framework [17] – Armando Fox (UC Berkeley) - Amazon EC2 and other cloud-based tools [2, 6, 7] – Jeff Chase (Duke) - ORCA framework [15]
10:30 – 10:45	Coffee & Tea Break
10:45– 12:15	Session 2: Educator Experiences (Undergraduate courses at small colleges) – Jeannie Albrecht (Williams College) - Distributed Systems [1] – Joel Sommers (Colgate University) - Computer Networks [19, 20] – Tia Newhall (Swarthmore College) - Parallel and Distributed Computing [12] – Zongming Fei (Kentucky/Calvin College) - Networking and Distributed OS [10]
12:15 – 1:30	Lunch
1:30 – 3:00	Session 3: Educator Experiences (Ugrad/graduate courses at large universities) – Armando Fox (UC Berkeley) - Massively open online courses (MOOCs) [5, 16] – Sonia Fahmy / Ethan Blanton (Purdue) - GENI-based classroom exercises – Anish Arora (The Ohio State) - Projects designed for local and remote testbeds – Mark Berman (BBN/GENI Project Office) - Sample GENI assignments
3:00 – 3:30	Platforms and TCPP – Jelena Mirkovic (USC/ISI) - DETERlab [4] – Sushil Prasad (Georgia State) - Technical Committee on Parallel Processing
3:30 – 4:00	Discussion on integrating classical and modern topics (The role of textbooks; Combining new technologies and old concepts)
4:00 – 4:15	Coffee & Tea Break
4:15 – 5:15	Panel/Round-table discussion and wrap-up

Table 1: Workshop agenda.

2.1 Session Summaries

This section briefly summarizes the main ideas of the presentations made in each session. The speakers' slides are available at the aforementioned website.

Session 1: Platforms - The main focus of the first session was to highlight some of the platforms available for hosting assignments. The platforms discussed included Seattle [18], ProtoGENI/Emulab [17], Amazon EC2 [2], and ORCA [15]. Given that the workshop was co-located with a GENI Engineering Conference, GENI platforms and tools were well-represented during this session. In general, the speakers highlighted the main features of each platform, and briefly summarized the design of the underlying systems, how student accounts are administered, where to go for more information, etc. The audience seemed very interested in learning more about these platforms, and many good questions were asked.

Session 2: Educator Experiences - The main focus of the second session was to summarize the experiences of educators at small colleges working primarily with undergraduate students. The speakers presented an overview of their courses on either Computer Networks, Distributed Systems, or Distributed Operating Systems, including sample projects, student performance, assigned textbooks and/or research papers, programming languages used, etc. Given the size of most departments represented, one interesting fact that arose out of the discussions in this session was that it is often not feasible to teach an entire course on Distributed Systems at a small school. Instead, the educators often merge the main aspects of Distributed Systems into another course on Computer Networks or Parallel Computing, for example. Also, given that many electives at small schools are only offered every other year, it is also difficult to use other systems classes (like Operating Systems or Computer Networks) as prerequisites to Distributed Systems. As a result, student background can vary widely. There were many good discussions and questions from other educators that resulted from this session.

Session 3: Education Experiences - The main focus of the third session was to summarize the experiences of educators at larger universities. The key difference between the speakers in this session versus the previous session was that the projects were designed for (very) advanced undergraduates or graduate students. The other key distinction was that there were typically more local computing resources available at larger universities than small colleges. The presentations in this session did not describe the structure of courses as much as the previous session. Instead, the speakers focused on individual projects, including details about how to set them up, student feedback, etc. Given the more advanced setup required for these projects, as well as the fact that this session was later in the day, the discussions did not seem as as lively as previous sessions.

3 Outcomes

Several common themes arose throughout the course of the day, both in the speakers' presentations and in followup discussions. They included:

- Combine well-defined labs with open-ended projects - The majority of the educator presentations, including those from small schools in addition to the larger universities, described a course in which the programming projects started as well-defined labs early in the semester, and culminated in a larger, independent final project on a topic chosen by the students. This structure seemed to strike an appropriate balance between teaching students the fundamental skills needed to develop, deploy, and evaluate distributed systems and encouraging creativity and exploration.

- Reading and writing papers - In addition to labs and open-ended projects, several of the educator presentations also emphasized the importance of teaching students to read and write about systems, in addition to teaching them how to implement them. There is often a disconnect between the code that students produce and their high-level vision of what the code actually does. When only looking at code, it is difficult to "see the forest for the trees." In addition, requiring students to step back and describe their systems in English helps them improve their written and oral communication skills. Similarly, by requiring that students read research papers from top systems conferences (in addition to traditional reading assignments from textbooks), students gain an appreciation for the importance of describing systems in a way such that other people can understand the design without reading or understanding the code.

- Emphasize breadth over depth - There was an overwhelming consensus that given the range of technologies available to students, it is important to emphasis breadth over depth. Thus, rather than spending an entire semester learning the specifics of a single platform or technology, most workshop attendees agreed that students benefit more from seeing a variety of platforms and technologies. It is important to teach students that there is no "one sized fits all" solution to systems design and analysis. When students are exposed to several different techniques, they not only learn about the advantages and disadvantages of each individual approach, but they also learn how to make design decisions. One key skill required to be a successful systems designer is the ability to evaluate tradeoffs, since the best solution for one problem is rarely the best solution for another problem. Emphasizing breadth over depth prepares students for making informed future decisions.

- Promote learning by doing - Another common theme in several of the presentations was the importance of “learning by doing.” Explaining concepts, technologies, and tradeoffs to students in a lecture setting is important, but without any hands-on experience in both the design and implementation of big systems, students are unable to appreciate how all of the concepts fit together. Similarly, whenever possible, students should gain this hands-on experience in “real” environments, since varying network conditions and unreliable links can greatly impact the performance of a distributed networked system. Students only truly master the necessary skills by actually building real distributed systems.

- Experience with wide-area debugging - Experienced systems designers know that a system that works in a small, non-volatile network, such as a LAN, often exhibits strange and unacceptable behaviors when run across the wide-area. Coping with the unpredictable and volatile properties of wide-area links can be very challenging. Until recently, it has been difficult to create an environment in which students could experience the challenges of wide-area debugging for themselves. However, the increasing availability of wide-area platforms such as GENI [8] has proven to be critically valuable in this regard by creating new opportunities for both educators and students to deploy their systems across the wide-area. Unfortunately, wide-area debugging can be frustrating for students. Educators should give students advanced warning about the difficulties that arise in these settings, and teach students how to cope with the challenges.

- Exposure to low level details - Higher level programming languages, such as Python and Java among others, have the advantage of hiding some low level details from developers. This allows for faster development, and often fewer bugs. However, the consensus from the educators at the workshop was that students need to be exposed to the low level details (i.e., sockets in C) at least once. By having to manually create sockets, set the appropriate parameters, and deal with errors that arise in these steps, students gain an appreciation for the abstraction provided by higher level languages. It is important for students to understand what exactly happens when they create a network connection in a high level language, even if the language itself hides the details.

- Help students learn to analyze system performance - One final theme that was repeatedly discussed at the workshop was the importance of teaching students how to experiment with and analyze system performance. Almost all good systems research papers include at least one performance graph that illustrates how the system performs under different conditions. In many computer science courses, especially at the undergraduate level, writing code that correctly solves the problem at hand with some reasonable level of elegance is often enough to receive a good grade on the assignment. Particularly in distributed systems, however, just getting the code to work is frequently not enough. Building a “good” system involves making tradeoffs based on the application’s intended use, and these tradeoffs must be quantified in many cases. Thus, as educators, it is important to teach students how to analyze system performance and measure the appropriate tradeoffs.

4 Next Steps

In addition to the aforementioned outcomes, conversations and discussions at the workshop revealed the following important future directions for the community:

- Perhaps most importantly, the workshop revealed the need for the attendees to continue to work together to build a stronger community. A non-trivial amount of relevant and useful course material has already been developed, and moving forward, there needs to be more of an effort to increase communication among educators and leverage the work of others.

- Although the main focus of this workshop was on redefining distributed systems education, several conversations indicated that there was also interest in gathering educators with more experience with courses in computer networks. The GENI Project Office has already agreed to fund this followup workshop, which will likely be held in October 2013.
- In addition to building a stronger community of educators with a common interest in distributed systems, the attendees of the workshop agreed that another important future goal is to continue to promote and advertise successful projects and course materials at major conferences and workshops, including SIGCOMM, OSDI, SOSP, NSDI, and SIGCSE. Several attendees suggested organizing panels or birds of a feather sessions at these venues in an attempt to reach out to additional educators.
- Finally, there were several discussions at the workshop that focused on developing a website that could serve as a central portal for aggregating course materials and modules. This website will need to provide wiki-like access, where individual educators may post or download various materials. However, it should also be monitored (although who should do this monitoring is not entirely clear) to provide some base level of completeness and relevance. One key question that arose in this context was where this website should be hosted. Several suggestions were made, but none seemed overly promising.

5 Summary

Overall, the workshop was a success in many ways. First and foremost, connections were made among people with similar interests, and conversations were started about what the next steps should be. Building a community of people who are committed to helping one another and working together is a very important step in any curricular discussion. In addition, several positive outcomes emerged, as discussed in Section 3.

One important non-outcome of the workshop was that there was little interest among the attendees in developing a fully standardized curriculum for undergraduate distributed systems. Given the variety of schools represented, it seems unrealistic to attempt to reach a consensus on exactly how the course should be taught in all schools. One complicating factor in this context is the fact that, particularly in small colleges, distributed systems is often combined with related topics, such as operating systems, networks, or parallel processing. Similarly, student background and course prerequisites vary widely across schools. Therefore, rather than developing a standard, “drop-in” curriculum, a more realistic and attainable goal is to develop a set of modules for key knowledge areas. The ACM/IEEE-CS Joint Task Force has been working to identify these key knowledge areas as part of the Curricula 2013 (CS2013) initiative [3]. Thus, a module might consist of a set of lectures addressing a specific knowledge area, as well as sample textbook readings, projects, and written homework. Educators can then combine subsets of these modules to build courses that fit into their respective curricula.

References

[1] Jeannie Albrecht. Bringing Big Systems to Small Schools: Distributed Systems for Undergraduates. In *Proceedings of the Fortieth ACM Technical Symposium on Computer Science Education (SIGCSE)*, March 2009.

[2] Amazon EC2. http://aws.amazon.com/ec2/.

[3] CS2013. http://ai.stanford.edu/users/sahami/CS2013/.

[4] DeterLab. http://www.isi.deterlab.net/index.php3.

[5] Armando Fox. CS169.1x: Software as a Service (EdX course website), Fall 2012. https://www.edx.org/courses/BerkeleyX/CS169.1x/2012_Fall/.

[6] Armando Fox and David Patterson. Crossing the Software Education Chasm. *Communications of the ACM (CACM)*, 55(5), May 2012.

[7] Simson Garfinkel. Commodity Grid and Computing with Amazon's S3 and EC2. *;login: (The USENIX Magazine)*, Febuary 2007.

[8] GENI. http://www.geni.net.

[9] Google App Engine. http://code.google.com/appengine/.

[10] W. David Laverell, Zongming Fei, and James N. Griffioen. Isn't It Time You Had An Emulab? In *Proceedings of the Thirty-Ninth ACM Technical Symposium on Computer Science Education (SIGCSE)*, March 2008.

[11] Microsoft Azure. http://www.microsoft.com/windowsazure/.

[12] Tia Newhall. Parallel and Distributed Computing Course Website (Spring 2012). http://www.cs.swarthmore.edu/ newhall/cs87/s12/.

[13] Open Cirrus. https://opencirrus.org/.

[14] Open Science Grid. http://www.opensciencegrid.org.

[15] ORCA Project Website. https://geni-orca.renci.org/trac/.

[16] David Patterson and Armando Fox. *Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing, Beta Edition*. Strawberry Canyon LLC, 2012.

[17] ProtoGENI. http://www.protogeni.net.

[18] Seattle: Open Peer-to-Peer Computing. https://seattle.cs.washington.edu/.

[19] Joel Sommers. Educating the Next Generation of Spammers. In *Proceedings of the Forty-First ACM Technical Symposium on Computer Science Education (SIGCSE)*, March 2010.

[20] Joel Sommers and Andrew Moore. Scaling the Practical Education Experience. In *Proceedings of the ACM SIGCOMM Education Workshop*, August 2011.

[21] XSEDE. https://www.xsede.org.