EDITORIAL

Musings

RIK FARROW

Rik is the editor of *;login:*.
rik@usenix.org

I just got back from attending the USENIX Annual Technical Conference in Denver. For those of you without gray hairs, USENIX ATC was, for many years, a twice yearly event that drew thousands of attendees for the summer and winter conferences. These were the only USENIX conferences, and the papers covered topics from systems to system administration.

Starting with LISA, new conferences were spun off USENIX ATC to cover specific topic areas: security, file systems, storage, networked systems, and even OSDI, a systems conference. Today, USENIX ATC represents just a shadow of its past, and attendance has dropped from over 3000 to under 300. USENIX ATC is still an important conference for systems researchers.

That's not what I want to write about. USENIX conferences were not considered good material for obtaining tenure, the process whereby an Assistant Professor gains the approval of his or her peers and obtains a lifetime appointment as a full Professor. Margo Seltzer, a past USENIX Board president, worked hard at changing this perception of USENIX conferences, and today USENIX conferences have been given equal weight for evaluation with conferences sponsored by the more traditional CS organizations like ACM and IEEE.

Margo's success did not come without side effects. USENIX conferences were once known for the pragmatic nature of the research accepted and published in their proceedings. As the program committees shifted to a more academic focus, a lot of this pragmatism faded away, replaced by a new pragmatism, one focused instead on publishing papers.

As an outside observer—that is, one not interested in tenure—I was more aware of the change in the tenor of accepted papers. I witnessed a progression to more theoretical work and research that, while interesting, would never be implemented, as the improvements in performance were small, or were only useful in special cases, not for general use.

I also heard grumbling within the ranks. Before USENIX ATC even began this year, I heard people complaining about the quality of some paper reviews. One student suggested that paper submitters be allowed to rank reviewers, just as their own papers were being ranked.

Hakim Weatherspoon mentioned that even though the review process involves blinding, the obscuring of the identities of paper authors, when a group of program committee members tromps out of the PC meeting to avoid a conflict of interest, the remaining PC has very strong hints about the authors of that paper.

Turns out, I hadn't heard anything yet.

The Emperor's Clothes

Bryan Cantrill, the inventor of DTrace and current CTO of Joyent, presented a keynote at USENIX ATC '16 on Thursday morning. He titled his talk, "A Wardrobe for the Emperor: Stitching Practical Bias into Systems Software Research," something that told me little about what was about to unfold. I suggest that you listen to his talk [1], but be aware that it contains some strong language. Much of it is provacative and felt to me like a roast that included a significant number of the senior audience.

Bryan's main point is that program committees for systems conferences have not just veered away from the pragmatic, they've become way too focused on tenure-securing behaviors.

Bryan wasn't taking advantage of a speaking slot to complain about having his papers rejected. He was dramatic for a reason: to stimulate discussion about changing the way the computer science community accepts papers and the purposes for holding conferences. He scheduled a Birds-of-a-Feather (BoF) slot for later the same evening, and showed up to lead the discussion. There were about 30 people in attendance at the start of the BoF, and an hour later people were still talking.

At the BoF, Jon Howell (Google) pointed out that program committees stamp a "quality indicator" on the accepted papers. Another person mentioned the fear of accepting a paper that would be trashed when it appeared at the conference, leading to PCs being less willing to take chances on novel or not-fully-baked research. Bryan mentioned the low acceptance rates, which were once around 25%, but now are less than 20% and sometimes under 15%. Those rates are very low compared to other academic areas, such as microbiology.

Bryan suggested an arXiv (https://arxiv.org/) model, where all papers get "published" and people vote up research that they find the most useful or interesting. Mothy Roscoe thought that this would not work due to the volume of paper submissions, as it was difficult enough to review the smaller number of papers that were submitted to a conference like OSDI (that he co-chairs with Kim Keeton in 2016). Mothy went on to describe how they had structured the OSDI '16 PC, with a two-tiered reviewing system designed to be fair to paper submitters and to reduce the paper-reviewing load on reviewers.

I suggested accepting more papers than there are speaking slots, but only permitting those who could prove their ability to present through the use of a short video. Hakim Weatherspoon, USENIX ATC '16 co-chair, USENIX Board member, and the person who invited Bryan to deliver a keynote address, thought this was impractical. I, and others, pointed out that *every* person presenting at USENIX Security had to create a short video that appeared in the morning Lightning Talks, and using something like this would be a great way to choose those people best at making presentations.

I must confess that the ability to present is a personal sore point for me. I attend paper presentations so I can learn more, and I have learned a lot from the better presenters. The official line on presentations, something I had guessed, and then had confirmed during this BoF, is presentations are not for the purpose of educating your audience. While the presenter certainly does share some information about the paper, the official purpose was to allow audience members to challenge the accepted paper in public during the presentation, putting both the presenter's work, and the decision of the PC, on the line. In reality, this rarely happens, and usually consists of someone complaining that their own paper, with some related work, wasn't cited.

Bryan had also shown a graph of how industry participation in OSDI PCs had dropped from a high of 100% in the late '90s to around 20% currently, resulting in an academically weighted committee. Bryan's definition of a PC member with an "industry connection" was, he stated, very generous, and it would have to be to have reached 100% at any point.

A speech and a BoF about systems conferences and program committees cannot effect immediate change on its own. What Bryan has managed to do, I hope, is to draw attention to the current state of the process by which the systems program committees review and accept papers, and how conferences are organized. Whether we will ever have papers without presentations, accept more papers than there are presentation slots in the program, or use an arXiv-like system instead of the current system are things that steering committees and future program committees will have to decide.

The Lineup

After having written all of that, you might wonder if I've chosen any papers from USENIX ATC '16 for conversion into articles for this issue of *;login:*. Yes, I had picked out a couple of papers, but I also mined Eurosys 2016, as those papers had appeared earlier in the year, and I found some real gems there.

I asked the authors of two related papers if they could write articles about their research. You'll notice that both have quite a practical bent to them. The first paper, by Atlidakis et al., examines how the POSIX standard actually gets used in applications for Debian/Ubuntu, Android, and Mac OS X. What they reveal through their analysis of applications demonstrates the weaknesses of a standard that was created to aid in porting applications between systems over 25 years ago.

Tsai et al. take a different approach, studying how applications use the Linux API. Their work was motivated, in part, by a quest to learn just how much of the API was required to run the most commonly installed Linux system applications, and just how well Linux emulations succeeded at providing a complete Linux API. Some of the same issues, such as use of ioctl(), appear here that appeared in Atlidakis' work.

If you've ever wondered about just how hard it might be to write a useful operating system, you might have come to a thought I'd had decades ago: that the success of an operating system can be measured in its support for the applications that people actually want to use. The operating system itself is secondary as long as the OS offers good performance and reliability. Together these articles, and the papers they are based upon, take a current look at just how hard it is to upset the systems that are currently enthroned. The Tsai paper also points out that people program using frameworks these days, yet operating systems expose very different interfaces.

I invited Jian Xu and Steven Swanson, the authors of a FAST '16 paper on a file system designed for non-volatile memory, to write about their NOVA file system research. Getting another file system into Linux is almost impossible. Yet, non-volatile memories, such as spin memory or Intel/Micron XPoint 3D, require new approaches so that these new persistent storage systems can obtain the maximum potential performance. These memories are not like disks or Flash devices.

Continuing along the theme of systems, I interview Mothy Roscoe again. I had interviewed Mothy six years ago, but wanted to ask about the Barrelfish project, a multikernel OS research project that is still going almost a decade after it first began.

I asked Diego Ongaro, one of the authors of the Raft Consensus Algorithm, presented at USENIX ATC '14, to write more about Raft (https://raft.github.io/), an alternative to Paxos. Diego told me that he wanted to write about Runway, a tool for model checking, simulation, and visualizing the workings of distributed systems. Runway is a work-in-progress, and the Web page (https://runway.systems/) includes a visualization of the Raft Consensus Algorithm in action.

Kalia et al., winner of a Best Paper award at USENIX ATC '16, write about getting the best performance when using RDMA (remote direct memory access). RDMA has long been used in HPC, and is starting to gain some traction in distributed applications. The authors do a good job of explaining how to use RDMA operations for the best performance, something that turns out to be complex but when done well can result in a hundred times faster performance.

I asked Betsy Beyer, an editor of *Site Reliability Engineering* [2], if she would contact the co-authors of one of the *SRE* chapters that really called out to me, the one about the meaning of toil, and reprise it for *;login:*. They did that and more, coming up with a great case study about eliminating toil.

Jonathon Anderson wrote a second part to his article in the summer 2016 issue about routing on Linux systems. The Linux kernel supports multiple default routes, and for multi-homed servers, that's exactly what you want to use.

The Bitcoin blockchain appeals not only to those interested in a non-fiat currency, but also to anyone who would like to take advantage of a system that publishes secure summaries of transactions. Ali et al. present their open source system, Blockstack (blockstack.org), which provides a programming framework for people interested in building systems for name registrations (à la DNS), as well as other uses that rely on a blockchain.

I asked Bruce Potter, one of the founders of SchmooCon (http://shmoocon.org/) and a long-time security expert, to write about how to create a threat model for your own organization. Bruce tells us why a threat model is useful, as well as how you go about creating one and keeping it current.

Dave Beazley reveals one reason why he has been so focused on the Python 3.5 `async` feature. Dave has been writing a module called Curio, and in his column this month he argues for the separation of protocols, such at HTTP/2, and the underlying transport (such as sockets). Curio itself is a library for concurrent I/O (https://curio.readthedocs.io/en/latest/).

Dave Josephsen writes about the emotional stress caused by being on-call. As someone who has spent way too much of his life doing this, Dave was motivated by talks at Monitorama (http://monitorama.com) and wrote a Go program for extracting data using the PagerDuty API. Dave used Go and describes how his program can be easily changed to use other input sources or sinks.

David N. Blank-Edelman exposes us to Consul, an open source distributed key-value system from HashiCorp. The value of Consul goes well beyond Perl users for anyone building distributed services that need a way of finding and updating configuration information on the fly.

Kelsey Hightower makes the case for cleanly shutting down services. Kelsey uses a simple Go program and the `manners` module, which makes it easy to keep a Web service alive until it has processed current requests.

Dan Geer argues for a science of security. He uses the work of T. S. Kuhn as the basis for his argument that paradigms are collections of knowledge representing the current understanding of a branch of science. I liked Dan's discussion, although I wonder if we have even reached the level of engineering when it comes to security. If we built bridges like we build software, no one would dare drive across them.

Robert G. Ferrell maintains his position as commenter-in-chief, and discusses how to learn from distributed computing to make the electric grid truly distributed.

Mark Lamourine has written two reviews about books on Go, and one about Docker, including useful information about the new Docker tools for container orchestration.

Carolyn Rowland, the newly minted USENIX Board President, writes about her experience volunteering for USENIX in USENIX Notes.

Pragmatic Systems Research

USENIX has been known for the high level of pragmatism found in the research published in its conferences. If you read any of the series of history articles found in *;login:* issues from 2015, you would see early *;login*: issues with patches for device drivers included. The UNIX operating system once ran on systems with 32 kilobytes of RAM, and managed to be so useful as to spawn an entire industry.

Today, we have operating systems like Linux, with its 312 system calls, over 700 ioctls and prctls, as well as pseudo-file systems like `/proc` and `/dev` that expose kernel information as files. Yet programmers write using studio tools, which work on top of frameworks, suggesting that only the authors of frameworks need to understand an operating system's interfaces.

I strongly believe that Bryan Cantrill raised some important points in his USENIX ATC '16 keynote. I had heard rumblings about the failures of the current systems many times before. Program Committees are, after all, human institutions, and that means that bias is always an issue, not just a possibility.

Should systems research be published for the purpose of extending tenure to deserving Assistant Professors, or should that be a by-product of creating research that makes the systems we design more useful and secure? Ideally, we will be doing both.

References

[1] Bryan Cantrill's USENIX ATC '16 Keynote: "A Wardrobe for the Emperor: Stitching Practical Bias into Systems Software Research": https://www.usenix.org/conference/atc16/technical-sessions/presentation/cantrill.

[2] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, eds., *Site Reliability Engineering* (O'Reilly Media, 2016).