OPERATING SYSTEMS REVIEW

A Publication of the
Association for Computing Machinery
Special Interest Group on Operating Systems

Volume 42, Number 3 **April 2008**

SPECIAL TOPIC **Computer Forensics**
GUEST EDITORS **Ewa Huebner and Frans Henskens**

OPERATING SYSTEMS REVIEW

A Publication of the ACM Special Interest Group on Operating Systems

OPERATING SYSTEMS REVIEW is a publication of the ACM Special Interest Group on Operating Systems (SIGOPS), whose scope of interest includes: Computer operating systems and architecture for multiprogramming, multiprocessing, and time sharing; resource management; evaluation and simulation; reliability, integrity, and security of data; communications among computing processors; and computer system modeling analysis.

Membership in SIGOPS (at $15 per annum) is open to ACM members and associate members, and student members (at $5 per annum). Non-members of ACM may subscribe to OPERATING SYSTEMS REVIEW at $30 per year. All SIGOPS members receive OPERATING SYSTEMS REVIEW. SIGOPS membership application forms are available from ACM Headquarters, 2 Penn Plaza, Suite 701, New York, NY 10121-0701, telephone +1-212-869-7440. Changes of address and other matters pertaining to the SIGOPS mailing list should be directed to ACM Headquarters, not to any of the SIGOPS officers.

Contributions to OSR: Several types of contributions to OSR are solicited from the operating systems community. First, many issues of OSR will be organized around a special topic or theme. Guest editors are sought to solicit, review, select and coordinate articles related to the special topic. If you are interested in suggesting a special topic or in serving as a guest editor, contact the OSR editor. Second, contributions related to upcoming special topics are requested. Such contributions should be sent to the guest editor for the issue of interest. Calls for participation in upcoming special topics issues will be posted on the OSR web page (http;//www.acm.org/sigops/osr.html) and announced via the SIGOPS mailing list. Finally, individual submissions not related to specific special topics are welcome. Proposals for individual submissions should be sent to the OSR editor for review and to discuss possible publication dates. Each proposal should clearly say how the submission would be of interest to the SIGOPS community.

OPERATING SYSTEMS REVIEW (ISSN 0163-5980) is published four times a year (January, April, July, and October) by the Association for Computing Machinery, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and at additional mailing offices. POSTMASTER: Send address changes to OPERATING SYSTEMS REVIEW, ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701.

Chairs' Report on Twenty-First ACM Symposium on Operating Systems Principles

Thomas C. Bressoud
General Chair
Department of Mathematics and Computer Science
Denison University
Granville, OH 43023
bressoud@denison.edu

M. Frans Kaashoek
Program Chair
Department of Electrical Engineering and Computer Science
MIT
Cambridge, MA 02139
kaashoek@csail.mit.edu

1. INTRODUCTION

The 21st ACM Symposium on Operating Systems Principles (SOSP 2007) was held at the Skamania Lodge in Stevenson, Washington, USA from October 14th to October 17th 2007. The conference site is located in the Columbia River Gorge National Scenic Area, a spectacular canyon along the border between Oregon and Washington States in the Pacific Northwest, where the Columbia River cuts through the Cascades mountain range. Delegates were treated to breathtaking views of the gorge and mountains from the lodge's rustic and warm common areas and the weather cooperated for the arrival and early parts of the conference, though the rains came in force in the latter part of the conference.

1.1 Delegate Composition

The number of delegates totaled 491, ranking only behind SOSP 2003 in Lake George in attendance, where there were 495 delegates. Attendance was up roughly 25% from SOSP 2005 in Brighton, England (398 delegates), which was representative in size of other recent SOSPs. Table 1 shows the attendance of the last five SOSP conferences.

Table 1: SOSP Attendance

Year	Location	Delegates
1999	Kiawah Island, SC, USA	396
2001	Lake Louise, Banf, CA	351
2003	Bolton Landing, NY, USA	495
2005	Brighton, England	398
2007	Stevenson, WA, USA	491

Consistent with past SOSPs, almost half of the delegates were students, numbering 230 (47%). Among the remaining non-student delegates (261), over half (157) were from industry, with the rest (104) from academia. The ratio of industry delegates to academia delegates in this non-student subgroup showed a shift toward industry over prior SOSPs, where the ratio had been close to 50-50. SOSP 2007 also showed an increase in the number of institutions represented. Even in high-attendance years, like at Lake George, the number of institutions had been relatively stable around 100, with the attendance differences coming in the form of larger delegations per institution. This year, some 126 institutions were represented. The largest delegations were from MIT and Microsoft, with UCSD, Cornell, Google, and UT Austin following with sizable delegations. By geographic region, just about 400 of the delegates came from North America (81%), with European countries accounting for more than 70 delegates (15%) and the remaining 20 delegates came from Asia and Australia. By contrast (and to be expected), the European participation was down from SOSP 2005 in England, where there were 120 delegates, but was up significantly from SOSP 2003 in NY, USA, when there were only 50 European delegates.

1.2 Delegate Feedback

An online survey was conducted following the conference to solicit feedback from the attending delegates. Feedback categories were scored on a scale of 1 (boring/inedible/too-little/never-go-again) to 10 (exciting/gourmet quality/too-much/super). Response rate was 56% (272 respondents). Table 2 presents the results of this survey.

Table 2: SOSP Feedback Survey

Category	Average Score
Paper Quality	7.2
WiPS	6.0
Posters	6.1
Food	7.2
Site: Lodge	8.0
Site: Location	8.4
Interaction	6.2
Overall	8.4

2. TECHNICAL PROGRAM

2.1 Paper Selection

The technical program included 25 papers selected from among 131 submissions. By comparison, SOSP 2005 had 155 submissions and SOSP 2003 had 120 submisions. Selecting these 25 papers was difficult because so many of the submissions were of high quality. To make the selection process as fair and as consistent as possible the program committee employed a different process than used by previous SOSPs (but used successfully by other conferences such as SIGCOMM). The program committee consisted of 13 "heavy"-load and 13 "light"-load members. The heavy-load members reviewed about 34 submissions each and attended the face-to-face PC meeting in Cambridge, MA USA. The light-load members reviewed about 24 papers each and did not attend the PC meeting. In contrast, recent SOSPs used a small number of PC members (12-15) who read a large fraction of all submissions, sometimes assisted by external reviewers. SOSPs before that required all PC members to read all submissions.

The goal of the new process was to resolve the tension between having high-quality, consistent reviews, a large number of submissions (it has been steadily growing over the years), and a productive face-to-face meeting. With more PC members the PC did not have to rely on external reviews, which can be inconsistent because the external reviewers see only a small sample of the submissions, yet the workload for the individual PC members was manageable, allowing thorough reviewing. By having a subset of the PC members meet in person, the PC was able to have in-depth discussion and reach consensus through discussion (rather than voting). The larger overall PC also allowed a broader group of people to participate in the decisions.

Paper selection was a three round process, with multiple reviews by the PC generated in each round and with reviewers targeted by subject expertise. The first two rounds reduced the pool of considered papers by 50%. The 62 remaining papers produced another two reviews apiece and all 705 reviews were assessed in preparation for the PC meeting. At the PC meeting, the 62 papers were ranked by review scores for discussion order and each assigned a champion to summarize content and strengths and to lead the discussion on individual papers. The PC discussion for each paper followed until consensus was reached. Throughout the process anonymity was maintained and conflicts of interest precluded by removing authors or those with direct association with an author from the discussion. In the final selection, 3 papers were co-authored by heavy-load PC members, and 6 were co-authored by light-load PC members.

Independent of the regular program committee, Rebecca Isaacs of Microsoft Research Cambridge organized a shadow program committee for SOSP 2007. This educational experience is reported on later in this issue in a separate article.

2.2 Program

The program presented important results in a wide range of areas, continuing the recent SOSP trend of technical breadth as exhibited by the nine session topic areas:

· Web Meets Operating Systems
· Concurrency
· Byzantine Fault Tolerance
· Software Robustness
· Distributed Systems
· System Maintenance
· Energy
· Storage
· Operating System Security

All nine of the sessions were "scribed" by student volunteers to capture the question and answers following each paper presentation. These scribe notes, along with the paper abstracts to set context, are included in a following article in this issue. For the full papers, the reader is encouraged to read the proceedings. Copies of the presented slides are also available on the conference web site under the *Techincal Program* link at `http://www.sosp2007.org`.

Through the support of ACM, all presentations were also video-recorded. These videos, which include captures of the slides as well as the speakers, is part of an effort by the ACM Digital Media Capture initiative for demonstrating capture and delivery of conference presentations and will be available along with the papers in the ACM digital library. As part of this initiative, the presentations are also currently available on the web at `http://dmcc.acm.org/sosp2007/`.

SOSP 2007 offered opportunities to learn about the state of the art in systems research through Work-in-Progress (WiP) presentations and through a Poster Session. A total of 12 WiP talks were accepted and can be found, along with PowerPoint and/or PDF of the slide presentations at `http://www.sosp2007.org/wip-program.html`. There were 24 posters presented at the conference, most with accompanying demonstrations. The list of posters can be found at `http://www.sosp2007.org/poster-program.html` These two forums gave delegates the opportunity to meet and share ideas with some of the brightest young researchers in the field.

3. AWARDS

Three papers were distinguished with the "Best Paper" award:

- *Secure Web Applications via Automatic Partitioning* by Stephen Chong, Jed Liu, Andrew C. Myers, Xin Qi, Krishnaprasad Vikram, Lantian Zheng, and Xin Zheng
- *Zyzzyva: Speculative Byzantine Fault Tolerance* by Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong
- *Sinfonia: A New Paradigm for Building Scalable Distributed Systems* by Marcos K. Aguilera, Arif Merchant, Mehul Shah, Alistair Veitch, and Christos Karamanolis

In addition, during the conference, the audience voted on the papers/presentations that they enjoyed the most. The "audience choice" papers selected by this process were:

- *TxLinux: Using and Managing Hardware Transactional Memory in the Operating System* by Christopher J. Rossbach, Owen S. Hoffman, Donald E. Porter, Hany E. Ramadan, Aditya Bhandari, and Emmett Witchel
- *Dynamo: Amazon's Highly Available Key-Value Store* by Guiseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swami Sivasubramanian, Peter Vosshall, and Werner Vogels
- *Generalized File System Dependencies* by Christopher Frost, Mike Mammarella, Eddie Kohler, Andrew de los Reyes, Shant Hovsepian, Andrew Matsuoka, and Lei Zhang
- *Secure Virtual Architecture: A Safe Execution Environment for Commodity Operating Systems* by John Criswell, Andrew Lenharth, Dinakar Dhurjati, and Vikram Adve

SOSP 2007 provided the opportunity to honor some of the field's most significant research and researchers. Continuing its inaugural at SOSP 2005, the SIGOPS Hall of Fame awards were presented at the conference. Five awards were presented, bringing the total awards conferred to ten. The following papers received awards:

- Leslie Lamport, *Time, Clocks, and the Ordering of Events in a Distributed System*, Communications of the ACM 21(7):558-565, July 1978.
- Andrew D. Birrell and Bruce Jay Nelson, *Implementing Remote Procedure Calls*, ACM Transactions on Computer Systems 2(1):39-59, Feb 1984.
- J. H. Saltzer, D. P. Reed, and D. D. Clark, *End-To-End Arguments in System Design*, ACM Transactions on Computer Systems 2(4):277-288, Nov 1984.
- Michael Burrows, Martin Abadi, and Roger Needham, *A Logic of Authentication*, ACM Transactions on Computer Systems 8(1):18-36, Feb 1990.
- Fred B. Schneider, *Implementing Fault-Tolerant Services Using the State Machine Approach: a tutorial*, ACM Computing Surveys 22(4):299-319, Dec 1990.

The Hall of Fame awards and the policies governing their selection are explained in another article in this issue.

At the banquet, the prestigious Mark Weiser award was presented. The Mark Weiser Award was created in 2001 by ACM SIGOPS, to be given to a young individual who has demonstrated creativity and innovation in operating systems research. The selection committee choses the recipient based on: "contributions that are highly createive, innovative, and possibly high-risk, in keeping with the visionary spirit of Mark Weiser." Peter Chen is the outstanding young researcher that was the recipient of the 2007 Mark Weiser award.

4. WOMEN'S WORKSHOP

The Systers electronic community was born at SOSP in 1987. In recognition of Systers' 20th anniversary, we held a workshop in conjunction with SOSP 2007. The workshop was targeted to women students at the graduate and senior undergraduate levels who have interests in computer systems research as well as early career women faculty and researchers. Workshop attendees also attended the SOSP conference. The workshop ran from the late afternoon of Saturday, October 13 through Sunday, October 14 at Skamania Lodge.

The workshop was designed as a community-building event, serving both to educate more women about the opportunities in systems research and to support women who have started working in the field. The content of the sessions enhanced the attendees' appreciation and understanding of the technical papers presented at SOSP. The workshop served as a friendly first meeting ground for junior members of the community to meet each other and more experienced researchers.

The list of speakers in the workshop program included: Carla Ellis, Sharon Perl, Barbara Liskov, Dilma da Silva, Cynthia Dwork, Susan Eggers, Rebecca Isaacs, Kim Keeton, Jinyang Li, Margaret Martonosi, Luiba Shrira, and Yuanyuan Zhou.

The workshop was a wonderful success. The number of student attendees numbered 63, and the non-student attendees added an additional 16 for a total of 79 participants in the workshop. Feedback from the women delegates was consistently glowing with much gratitude conveyed for the decision to include the Women's Workshop as part of SOSP 2007 and the great benefit from the content of the program itself. The organizers would like to particularly thank CRA-W, HP, and Google, who sponsored the workshop as well as the industry benefactors cited below that enabled so many student women to attend the workshop and SOSP.

5. SCHOLARSHIPS AND BENEFACTORS

One of the aspects of SOSP that makes the conference great is the participation of students. The students that are coauthors on papers, the students presenting WiP results, and the students presenting posters combine with the other attending students to make a very rich experience and to create a synergy with experienced researchers in discussing the latest results and directions in the systems field. With student participation consistently reaching levels of nearly half the delegate population, it is clear that the pipeline of future researchers in our field is strong and vital.

This level of student participation would, quite literally, not be possible without the help of both industry and government support. In past iterations of the conference, industry and the National Science Foundation (NSF) have combined to help fund the travel, shared lodging, and registration for fifty to sixty-some students (these are reprsentative numbers from 1999 through 2005), with total benefactor contributions amounting to US$50,000 to $75,000.

Beyond the "normal" level of student support, SOSP 2007 wished to support all of the student women participating in the Women's Workshop and to increase funding specifically to help underrepresented minorities in our field. This involved going to our benefactors and asking, not for a redistribution of their contribution to help in these areas, but additional funding, so that we could continue to support the level of students we have in the past as well.

SOSP 2007 succeeded beautifully! Scholarship contributions from our industry and governement benefactors totalled US$157,000 and supported 117 students. This is over half of the attending students. What is more, 51 of those scholarships were for women and underrepresented minorities. This helped to bring the overall ratio of women at the conference up to nearly 20%, and within the student population, the proportion of women was a full 30%. We need to make this a trend, and not just a data point. This can be accomplished by continuing some of the efforts initiated at SOSP 2007, including the Women's Workshop.

The following organizations gave generously to support these student scholarhips:

Table 3: Scholarship Benefactors

InfoSys
National Science Foundation
Hewlett-Packard
Sun Microsystems
Microsoft
Google

Additional donations also helped fund the conference and keep registration costs from escalating. These benefactors include SIGOPS, Intel, VMware, IBM Research, and Yahoo! Research.

6. APPRECIATION

That SOSP 2007 was a success, there is no doubt. This was due to an excellent technical program as well as to the conference site and logistics. The credit goes to so many volunteers who gave so much of their time and their talent to plan and execute and to attend to the thousands of details required to carry off a successful conference. It is like a complex design process that proceeds from the concept stages to the detailed implementation. These people share the vison of this conference, SOSP, as one that is special, and worth the additional effort to see it continue as one of the very best.

From the general chair: I must begin by thanking the program chair of the conference, Frans Kaashoek. Frans went far beyond just the care and creation of the technical program. He worked closely with the "general" side of the planning. He struck a excellent balance in offering advice when I sought it, pitching in when the need arose, and in using his influence to bring people in our community to help when needed. I very much appreciate all of his help. Thanks.

From the program chair: Big thanks to Tom for running a fantastic conference. Tom picked the location, ran the logistics of the conference, put the team together, managed the finances, interfaced with ACM, and in general worried about every detail. Tom had help from a wonderful team of people, but in the end the buck stopped with Tom. As can been seen from Table 2 in section 1.2, the site and conference overall were ranked very high by the attendance. On behalf of the SIGOPS community, Tom, Thanks!

And from both chairs: Likewise, we must also single out Jon Walpole, our local arrangements chair. At the onset of conference planning, none of us fully appreciated the amount of work required for local arrangements, with all of the "behind the scences" things that need to be done. Jon did a superb job and deserves all of our thanks. He has ours.

The team that together put on SOSP 2007 has our deep thanks. Jacob Lorch worked very hard (and successfully) raising scholarship monies, and giving Robbert van Renesse and Hakim Weatherspoon the resources for thier job of selecting the scholarship recipients. With additional advertising, and with additional dollars to spend, we also had record numbers of scholarship applicants making their job, in fact, more difficult. Michael Kozuch handled registration, and this went remarkably smoothly under his capable charge. We would like to thank Scooter Morris for initiating video capture of the SOSP sessions and to Rama Ramasubramanian for overseeing and executing the process. Our thanks also to Jason Flinn for handling publicity and getting our web site up. Jeremy Stribling served as our web master and was always helpful and responsive to our *many* requests for updates and changes. Our thanks to Jeremy for his many efforts.

The Program Committee for SOSP 2007 did an excellent job. This work involved much time and effort and a fervent desire to see the very best papers selected and presented. It is the high quality program that is the foundation for a good conference. We offer our sincere thanks to all of the members of the program committee.

The women's workshop was also a success due to the help of many. We would like to particularly thank Carla Ellis for her passion and efforts. She really served as both general and program chairs for the workshop. Also on the subject of the women's workshop, we would like to thank Sharon Perl and Jeff Mogul. They also had a passion for this effort and worked to see its successful outcome, including the involvement of their companies and the financial backing of Google and HP.

Respectfully submitted,

Thomas C. Bressoud
M. Frans Kaashoek

Report on the 2007 SOSP Shadow Program Committee

Rebecca Isaacs

October 2007

1 Introduction

Since its recent establishment, the European chapter of SIGOPS (EuroSys) has initiatied a number of activities to "increase the visibility and quality of systems research in Europe"[1]. This report describes one such activity, a shadow program committee for SOSP that was open to junior systems researchers based anywhere in the world. Feedback from the participants confirms that the experience was worthwhile and enjoyable, and it is my belief that a shadow PC also has long term benefits for the wider systems research community, as explained further below.

The model for the exercise was the Sigcomm shadow PC that Anja Feldmann organised in 2005 [2]. The shadow PC behaves to a large extent exactly as a real PC: members are assigned papers, write reviews giving feedback to authors, and then attend a meeting to choose a program. However after this point the shadow PC activities diverge from the usual PC behaviour. Firstly the shadow PC program is not made public, and secondly, the PC spends some time at the meeting discussing the peer-review process itself. This discussion typically covers how to write a systems paper, how to review one, the dynamics of choosing a conference program, and direct comparisons with reviews written by real PC members.

The primary purpose of a shadow PC is educational. For many junior systems researchers, the top conferences seem inaccessible and remote, appearing to be dominated by a small number of mainly US institutions. There is a sense of exclusion on three fronts:

- Those who do not understand the criteria that characterise a good systems paper often lack the wherewithal to even submit a paper, let alone one with a serious chance of acceptance.
- Without a strong publication record or a proactive mentor finding a place on that first PC can be difficult. Thus the opportunities to learn reviewing skills and to publically demonstrate competence in that regard are limited.
- A lack of involvement in the ongoing peer review process can lead to appreciating neither the value of feedback from peers, nor the variable nature of program selection.

The shadow PC exercise itself explicitly addresses the first two points. By participating in a PC, and having a subsequent group discussion to reflect on the experience, participants should acquire a much better appreciation of how to get papers published. The non-threatening and voluntary nature of the shadow PC provides a way to start to tackle the second problem.

The final point is addressed somewhat more indirectly, with the hope that the shadow PC exercise will stimulate a number of self-sustaining activities that will help to improve the situation. These might include discovering groups with mutual interests leading to visits, internships and collaborations, attaching value to attending SOSP rather than just reading the proceedings, and providing an opportunity, that might not otherwise arise, for young people to demonstrate their abilities and knowledge outside their home environment, and to find the confidence to start to engage with the community.

Any of these outcomes will broaden and strengthen the systems research community generally. The moti-

[1] `http://www.eurosys.org/`

vation for EuroSys in particular to sponsor a shadow PC is that these concerns resonate with observations about the state of systems research in Europe, many of which are articulated in the EuroSys white paper [1]. Because researchers everywhere, not just in Europe, also share these concerns, applications to take part in the SOSP shadow PC were invited from all systems researchers.

SOSP was chosen to be the conference to shadow because it is the top operating systems conference, ensuring exposure to the highest quality papers, and it has relevance right across the community unlike some others that are perceived as more specialized (eg NSDI). For the shadow PC participants there is a trade-off with achieving the same benefits by being on a real PC of a minor conference, where that is an option. Members of the SOSP Shadow PC volunteered with the expectation that the positive effects would outweigh the costs of the extra work.

In the remainder of this report I will briefly cover the organisation and logistics of the shadow PC, and then try to summarise the lessons learnt from the experience, both for the participants themselves, as well as about PCs in general in light of the differences between the shadow PC and the real PC.

2 Participation

The shadow PC was advertised on the Eurosys members mailing list, on the Sigops announcements list, and on the SOSP 2007 web site. Any systems researcher was invited to apply, with the advertisement stating that "ideally he or she is a post doc or recently appointed faculty member that has not yet been a member of PCs such as SOSP, OSDI, NSDI." I also contacted several senior people asking for recommendations, as well as writing to a few people directly. The goal was to have a shadow PC that was larger than the real one, and to try and ensure balance, breadth, sufficient maturity to do the job, and fair representation (e.g. not 10 people from the same institution).

The end result was 30 participants, about half of whom are researchers in industry or academia, with the remainder split evenly between faculty members and PhD students (most of whom finished, or intend to finish, during 2007). Around half of the PC members came from seven countries in Europe, and the remainder from the US and Canada.

SOSP submissions were not automatically made available to the shadow PC, but instead authors were asked to "opt-in" via a checkbox on the paper submission page. Of 131 submissions, 101 authors agreed to let the shadow PC review their papers. The real PC members were also asked whether they were prepared to share their reviews (anonymously) with the shadow PC, and all agreed to do so. As was the case with SOSP itself, all reviewing proceeded on a double-blind basis. Throughout the exercise it was made very clear to the shadow PC that they had the same obligations of confidentiality and respect for the double-blind reviewing process as the real PC. Authors were sent the shadow PC reviews a couple of weeks after the decisions of the real PC had been communicated.

Although the shadow PC was intended to be a practical learning experience, members were not thrown into the reviewing task completely without help. Before the reviewing period began, the shadow PC were given two documents with advice on reading papers [3] and on writing reviews [4].

3 Organisation

In order to maximise opportunities to learn, reviewing was organised slightly unconventionally. In the first round, each member reviewed 10 papers. Then in the second round, instead of eliminating the lowest ranked papers, assignments were made to ensure that each person reviewed at least one of the highly-ranked papers and at least one from the set with the lowest scores. Between three and four papers were allocated for the second round. The intent was to give the ample scope for calibration, while maintaining a relatively light reviewing load. The shadow PC used the same reviewing software as the real PC, making the experience as close to authentic as possible.

All but four members were able to attend the PC meeting in Cambridge and in addition, two members of the real PC were present in an advisory capacity.

The target number of papers for the program was 18, and the PC discussed 40 papers, leaving around an hour to reflect on the experience of being on a PC and on the process of choosing a program. To explore how the shadow PC compared with the real PC we compared the reviews from both committees for one paper that had been discussed earlier in the day. Although the meeting overran with a lively and informative discussion, fortunately we were not too late to go the pub before having a group meal.

From its batch of 101 papers the shadow PC selected 16 for its program. Of the 25 papers accepted by the real PC, the shadow PC had access to 18, of which nine were accepted by both committees, and a further four discussed but not accepted by the shadow PC.

4 Experiences

Examining the reviews of the nine papers that were accepted by the real PC but rejected by the shadow PC, it is apparent that the shadow PC was strongly influenced by poor presentation and prone to concerns as to whether the contribution is novel. In contrast to the reviews of the real PC, the shadow PC tended to adopt a non-commital viewpoint, most likely reflecting a lack of self-confidence.

Some of the differences between the programs resulted from the format of the shadow PC meeting. Unlike the real PC, the shadow PC did not have a single, specific goal (to choose a program), but the intention was also to contrive a situation where the mechanisms of a PC could be better understood, as well as to gain an appreciation of how papers are received by reviewers and what sort of papers tend to be judged favourably. Therefore, rather than considering each paper in strict order based on review scores, there was some jumping around, and of course the shadow PC covered far fewer papers. Inexperience played a big part—members did not know, for example, how much detail to include when introducing a paper for discussion. Most people erred on the side of too much.

I conducted an anonymous survey of the shadow PC participants after the meeting, which had 27 respondents. In answer to the question “was the experience worthwhile”, 25 chose “very much so” (the most positive rating), and two “somewhat” (the second most positive). The shadow PC members appeared to have found the experience itself beneficial, although it remains to what extent it addresses any of the broader concerns highlighted in the introduction section.

Some other interesting points that came out of the survey included:

- The most useful part of the experience was participating in a PC meeting, with seeing papers submitted to SOSP and reviewing papers submitted to SOSP a close joint second.
- 80% thought the PC dinner was important, in particular the opportunity to get to know people at a similar career stage from different places.
- Only half of the respondents felt they had adequate time to prepare for the PC meeting. This was most likely caused by a combination of not knowing what to expect at the meeting, combined with some last-minute preparation by the PC chair (which is not uncommon at real PC meetings too!)
- Aspects most enjoyed include meeting new people, reading and discussing the papers, the PC dinner, finding out how a PC meeting works and gaining insight into how to write papers and hearing comments from the real PC members who were present.
- Suggestions for future shadow PCs: “encourage the shadow PC to read more papers than they are assigned”! Provide more structured questions for reviewers, e.g. “Does this paper present anything new in its field?” The PC chair is advised to keep the PC meeting on schedule and give more advance notice of paper discussion leads. Members would like to have more real PC members present. A preliminary session the night before the meeting would help to prepare for the meeting itself.

On the day of the meeting the decisions of the real PC were not yet known to the shadow PC, and so we did not compare the two programs in our discussion. However by the time of completing the survey, the shadow PC had seen the final SOSP program and were asked to suggest reasons for the differences between the real and shadow PC programs. The answers are revealing. Below is a representative subset:

- "At the shadow PC meeting, I felt that only the members who reviewed were making any useful contributions to the discussion...With reviewing/reading more papers, I feel the PC meeting will be a more informed discussion ground".
- "The decision making during shadow PC seemed to be very much driven by the discussion leader and champion. Something more structured, even merely some check-boxes, should be of help."
- "The shadow PC seemed overly critical of papers on topics they understood and overly awed by papers on unfamiliar topics".
- "We were easily swayed to accept (or at least take into consideration) papers that were really out of scope or not of sufficient quality".
- "Most of the difference can be explained by experience and a real need to dispense with perfunctory advice to the authors and cut to the heart of the matter".
- "It was hard for the shadow PC to judge what was fixable in terms of writing and experimentation in the shepherding process".
- "There are some areas the shadow PC simply does not have adequate knowledge of".
- "Fewer reviews than the real PC, probably leading to somewhat greater randomness in our results".

5 Lessons learned

Although each individual will have taken something different from the experience, there were five points that came out in the discussion and the survey as being of particular value to many of the participants:

- The consequences of giving a low or high review score before the meeting. It often happens that papers with uniformly low scores are rejected without discussion. If you think a paper should be at least discussed, then you need to take a stand.
- The way that reviews are read and weighted by other PC members. A review is expected to rapidly convey information about the paper to rest of the PC, as well as provide feedback to the author.
- The importance of preparation when arguing for or against a paper. Spending time on your own reviews and reading other people's reviews in advance of the meeting will help to crystallize your arguments. If you want to make a useful contribution to the program selection process, do not underestimate the value of preparation!
- The benefit of having at least skimmed some of the papers you didn't review. Your perception of a particular paper may change in the context of other submissions, for example knowing whether any other papers consider the same topic.
- The impact of persuasive, articulate people at the meeting. More often than not, these people are persuasive because they have prepared well.

Lessons were also learned from the point of view of being an author, not least the need to write your paper in a way that someone at the PC meeting can argue for it, and an appreciation of the standard of writing and research contribution for SOSP.

For future shadow PC chairs some of the issues caused by reviewer inexperience could be addressed by appointing both an advocate and a detractor for each paper discussed rather than just a single discussion lead, thus forcing more debate to take place. Other improvements might include providing a generous amount of preparation time in advance of the meeting, and being specific about details such as timing (for example: "you will be asked to summarize the paper in two minutes or less").

To conclude, a high standard of research is reliant on an equally high standard for the mechanisms used to judge the quality of that research. I believe that a key factor in maintaining excellence in the peer review process is the opportunity to learn the skills required. The primary benefit of a shadow PC, both for the individual participants and for the community at large, is in providing an effective way of training those skills.

6 Acknowledgements

Thanks to Frans Kaashoek, the real PC Chair, for his support, encouragement and practical assistance, and thanks to Timothy Roscoe and S. Keshav for transcribing their extensive experience of reading and reviewing to document form. I am grateful to EuroSys, whose board proposed the idea of a shadow PC and who paid for the PC dinner, and to Microsoft Research for providing the room and refreshments throughout the day. Mike Schroeder and Paul Barham were the SOSP PC members who attended the shadow meeting and their presence was very much appreciated. A number of people gave valuable feedback on earlier drafts of this report, in particular Andy Warfield. Finally, thanks also to the shadow PC members who were extremely diligent, very enthusiastic, and somehow found the means to attend the meeting in Cambridge and make the experience a success.

References

[1] Peter Druschel, Rebecca Isaacs, Thomas Gross, and Marc Shapiro. Fostering systems research in Europe. A White Paper by EuroSys, the European Professional Society in Systems, April 2006.

[2] Anja Feldmann. Experiences from the Sigcomm 2005 European shadow PC experiment. *ACM SIGCOMM Computer Communication Review*, 35(2):97–102, July 2005.

[3] S. Keshav. How to read a paper. *ACM SIGCOMM Computer Communication Review*, 37(3):83–84, July 2007.

[4] Timothy Roscoe. Writing reviews for systems conferences, March 2007. `http://people.inf.ethz.ch/troscoe/pubs/review-writing.pdf`.

Policies for the SIGOPS Hall of Fame Award

Jeffrey C. Mogul
jeffmogul@acm.org

1. INTRODUCTION

The SIGOPS Hall of Fame Award was established in 2005 to recognize "the most influential Operating Systems papers" of the past. See `http://www.sigops.org/awards/hall-of-fame.html` for the Web page that describes the award.

The initial specification of the award turned out to have some problems in practice, which have become visible in the process of granting these awards over the past three years. This article describes the evolution of the award's specification.

At the SIGOPS business meeting, held as usual at SOSP 2007, we discussed the policies for the Hall of Fame Award. Generally the attendees agreed that the policies established for 2007 should be retained.

The rest of this article discusses the policy issues in more detail.

2. POLICY ISSUES

I will address the following policy issues:

- What papers are eligible?
- Who should decide which papers win?
- What are the conflict-of-interest rules?
- How many awards per year?
- How do the rules get changed?

2.1 What papers are eligible?

In the original specification of the award, eligibility was limited to SOSP papers at least 20 years old. This raised at least two questions:

- SIGOPS sponsors two major operating systems conferences, SOSP and OSDI. In recent years, the community has viewed these as of approximately equivalent quality, and it seemed reasonable to grant Hall of Fame awards at OSDI. However, since the first OSDI was held in 1994, no OSDI papers would be eligible under the "20 year" rule until 2014. Handing out awards at OSDI when no OSDI papers were eligible seemed a little odd. It also wasn't clear whether a paper really needed 20 years to demonstrate its lasting value.
- Many of the seminal operating systems papers, especially in the early years of the field, were published outside of SOSP or OSDI. Many people felt the limitation to papers published in just one or two conferences was unnecessarily arbitrary.

Of course, any award of this kind is necessarily arbitrary with respect to eligibility; an awards limited to the "most influential SOSP paper at least 20 years old" would still be quite an honor. But after some discussion among the members of the 2006 OSDI Program Committee (PC), among members of the 2007 SOSP PC, and at the 2007 SIGOPS business meeting, the community consensus seems to be that our award should have a broader scope: "operating systems papers that have appeared in the peer-reviewed literature at least ten years in the past."

We expect future award committees, however, will usually favor older papers, since the authors of these papers deserve recognition while they are still active in the field.

Another question arose because the specification says "[n]ominations will be solicited of the SIGOPS membership via email." Does this mean that only papers so nominated are eligible? In particular, can committee members nominate papers that were not nominated by SIGOPS members outside the committee? Generally this seems acceptable, and in fact has been the practice given that the general membership has not submitted that many nominations. This seems to cause no harm, and the community nominations often come with much more detailed rationales, so they are not at a significant disadvantage.

2.2 How many awards per year?

The original specification said "[to] bootstrap the award, up to five awards will be given at SOSP 2005 and SOSP 2007." In fact, at SOSP 2005 only four papers received awards. The consensus following the 2007 award process was that, while the backlog of meritorious older papers is decreasing, if we continue at a rate of only one award per year, it will be a long time before we catch up – perhaps too long.

This is another arbitrary decision. Strictly limiting the number of future awards to one per year might increase the prestige value of the award, but might also lead to perceived unfairnesses as authors of highly influential papers may have to wait for many years before being recognized.

The consensus at the 2007 SIGOPS business meeting was somewhat vague, but my sense was that people were willing to give future award committees some latitude to confer more than one award per year. This might continue either until the award committees decide that they have caught up, or until the community decides to impose a strict limit because the committees have become too generous.

2.3 Who should decide which papers win?

Originally, the task of deciding on award winners was given to the SOSP (or OSDI) PC for that year. Both the SOSP 2005 and OSDI 2006 PC chairs discovered that this did not work very well; most PC members were already burned out from the heavy review load, as well as the tasks of shepherding papers and picking best-paper awards for the current year. As a result, PC members tended not to participate with sufficient enthusiasm in the process of choosing Hall of Fame award winners.

For SOSP 2007, we instituted a new model, in which the award committee chair (chosen by the current PC chair) constitutes a committee from the chairs of the most recent four SOSPS and one co-chair from the most recent four OSDIs. This approach provides some load balancing (assuming that these "retired" chairs aren't as burned-out as current PC members), while also providing some continuity year to year. The somewhat arbitrary decision to use former SOSP and OSDI chairs was based on an assumption that these people have already been chosen for their good judgement and familiarity with the OS literature.

Since it might be impossible to convince an eligible former chair to participate in the committee during a given year, the new model allows the award-committee chair to substitute as necessary to come up with enough committee members. For example, in 2007, John Wilkes (chair of SOSP 1999) graciously agreed to participate.

2.4 What are the conflict-of-interest rules?

The original specification did not explain how committee members should decide if they were conflicted with nominated papers. Since members of our PCs have learned to be highly cautious about declaring conflicts for the papers that they review, our norms are quite strict. This approach generally does not work for determining conflicts for the Hall of Fame process, since authors of influential, older papers tend to have made connections with many other researchers over the intervening years – especially with the well-connected researchers who end up on PCs. Conflicts also arise from same-institution relationships, which can also multiply over the years. (There is also some possibility that overburdened PC members declared conflicts so as to avoid having to participate in the process.) When only a few award-committee members are able to vote on a nomination, and especially when the eligible voters for two nominations do not overlap, it becomes very hard to make good decisions.

In 2007, we decided on an explicit conflict policy before considering any nominations, and tried to balance the level of strictness so that the process could be seen as fair without running the risk of running too low on voters. The policy declares a conflict if the committee member was or is:

1. Involved in writing nominated paper
2. Currently from same institution as paper's author(s)
3. Same institution at time nominated paper was written/published
4. Relative of author of nominated paper

Note that these rules are generally analogous to the classic ISCA rules [1] but are not nearly as strict.

Even with these relaxed rules, in 2007 we discovered that at least one nominated paper had three rule-2 conflicts (currently from the same institution as an author of the paper). This left us with just five voters (out of eight committee members) who could vote on this paper. We realized that we had failed to determine in advance how many votes would be required in such a case. For example, a majority of the eligible voters (3 of 5) would not have been a majority of the whole committee, so this paper might have been accepted with fewer votes than a conflict-free paper whose nomination was rejected by a 5 to 3 majority. We therefore decided to condition acceptance on a positive vote total equal to an actual majority of the entire committee, allowing only the unconflicted voters to vote. In this case, that required all five voters to approve.

Rule-1 conflicts (a committee member involved in writing the nominated paper) are likely to arise, given that the award committee is composed of people who have some seniority in the OS community and are therefore likely to have written some influential papers during their careers. For this kind of nomination, we of course excluded the affected member(s) from all discussions and voting. We also set a high standard; we agreed not to approve such nominations unless we unanimously believed that they were clearly superior to all alternative papers.

Such a high standard does disadvantage committee members who have written possibly award-winning papers and who might be on the award committee for many years, which is a potential flaw in the current model for composing the committee.

2.5 How do the rules get changed?

SIGOPS does not have a well-defined process for changing the rules for this award. In fact, the rules have been different in each of the first three years, which is mostly my fault: as a co-chair of OSDI 2006, I hastily instigated a change that allowed nomination of previous OSDI papers, and as the award committee chair in 2007, I somewhat more carefully instigated the changes that are described in this article. In both cases, I obtained the approval of the current SIGOPS chair, but that process lacked community input and transparency, and in 2006 led to some criticism.

The current rules were discussed and generally approved at the 2007 SIGOPS business meeting, but that was not a truly democratic process (especially since at least half of the SOSP attendees spent the time drinking some excellent local wines instead of at the business meeting). The consensus seems to be that the award committee should be trusted to make necessary and modest changes on its own, with approval from the SIGOPS chair, but that larger changes demand further discussion with the membership.

3. SUMMARY

The SIGOPS Hall of Fame Awards have become a significant honor, and an opportunity for the community to recognize the publications that have had lasting influence on our field. We have to balance the need to preserve the prestige of the Awards against the need to recognize a broad range of significant work. Stable policies that gain community consensus will maintain the value of this award.

4. REFERENCES

[1] Mark Hill. Program Chair's Message. In *Proc. International Symp. on Computer Architecture (ISCA)*, 2005.

APPENDIX

A. HISTORY OF THE AWARD

The first awards were made at SOSP 2005. The awards were chosen by the SOSP PC, from the set of SOSP papers published at least 20 years earlier (that is, in or prior to SOSP 1985). That committee was authorized to choose five papers, but decided to give out only four awards.

SIGOPS co-sponsors the OSDI conference, in even-numbered years. For OSDI 2006, the official rules did not seem to apply, since the first OSDI was held in 1994. The OSDI co-chairs (myself and Brian Bershad) contacted the SIGOPS chair for guidance, and we ended up improvising modified rules: for 2006, the Hall of Fame award would be given to one paper published in OSDI at least 10 years earlier (that is, in 1994 or 1996). Again, the PC chose the award paper. These rules led to some complaints.

For SOSP 2007, the PC chair asked me to handle the Hall of Fame Award. After discussion among the SOSP PC members and with the new SIGOPS chair, we tried to rationalize the process with rules that seem to have achieved consensus: We would again give out up to five awards (the last chance to give out more than one, under the original design); we would consider any peer-reviewer paper in the operating systems literature at least 10 years old; and the award committee would consist of eight recent chairs or co-chairs of SOSP and OSDI. In 2007, five papers were chosen to receive awards; the youngest was from 1990 (17 years old) and not all of them were originally published in SOSP.

B. PAST AWARDS AND CITATIONS

Here is a list of the Hall of Fame awards to date, along with the statements prepared by the Award committee that describes why each paper was selected.

Links to all award papers are available via `http://www.sigops.org/awards/hall-of-fame.html`

B.1 2005 Awards

Edsger W. Dijkstra, *The Structure of the THE Multiprogramming System*, Proceedings of the First ACM Symposium on Operating Systems Principles, October 1967, Gatlinburg, TN, USA.

> The first paper to suggest that an operating system be built in a structured way. That structure was a series of layers, each a virtual machine that introduced abstractions built using the functionality of lower layer. The paper stimulated a great deal of subsequent work in building operating systems as structured systems.

Peter J. Denning, *The Working Set Model for Program Behavior*, Proceedings of the First ACM Symposium on Operating Systems Principles, October 1967, Gatlinburg, TN, USA.

> This paper introduced the working set model, which has became a key concept in understanding of locality of memory references and for implementing virtual memory. Most paging algorithms can trace their roots back to this work.

Dennis M. Ritchie and Ken Thompson, *The UNIX Time-Sharing System*, Proceedings of the Fourth ACM Symposium on Operating Systems Principles, October 1973, Yorktown Heights, NY, USA.

> At a time when operating systems were trending towards complexity, UNIX emerged as a hallmark of elegance and simplicity.

Butler Lampson, *Hints for Computer System Design*, Proceedings of the Ninth ACM Symposium on Operating Systems Principles, pp. 33-48, October 1983, Bretton Woods, NH, USA.

> A classic study of experience building large systems, distilled into a cookbook of wisdom for the operating systems researcher. As time has passed, the value of these hints has only grown and the range of systems to which they apply enlarged.

B.2 2006 Award

George C. Necula and Peter Lee, *Safe Kernel Extensions Without Run-Time Checking*, Proceedings of the Second USENIX Symposium on Operating Systems Design and Implementation, October 1996, Seattle, WA.

> This paper introduced the notion of proof carrying code (PCC) and showed how it could be used for ensuring safe execution by kernel extensions without incurring run-time overhead. PCC turns out to be a general approach for relocating trust in a system; trust is gained in a component by trusting a proof checker (and using it to check a proof the component behaves as expected) rather than trusting the component per se. PCC has become one of the cornerstones of language-based security.

B.3 2007 Awards

Leslie Lamport, *Time, Clocks, and the Ordering of Events in a Distributed System*, Communications of the ACM 21(7):558-565, July 1978.

> Perhaps the first true "distributed systems" paper, it introduced the concept of "causal ordering", which turned out to be useful in many settings. The paper proposed the mechanism it called "logical clocks", but everyone now calls these "Lamport clocks."

Andrew D. Birrell and Bruce Jay Nelson, *Implementing Remote Procedure Calls*, ACM Transactions on Computer Systems 2(1):39-59, February 1984.

> This is *the* paper on RPC, which has become the standard for remote communication in distributed systems and the Internet. The paper does an excellent job laying out the basic model for RPC and the implementation options.

J. H. Saltzer, D. P. Reed, and D. D. Clark, *End-To-End Arguments in System Design*, ACM Transactions on Computer Systems 2(4):277-288, November 1984.

> This paper gave system designers, and especially Internet designers, an elegant framework for making sound decisions. A paper that launched a revolution and, ultimately, a religion.

Michael Burrows, Martin Abadi, and Roger Needham, *A Logic of Authentication*, ACM Transactions on Computer Systems 8(1):18-36, February 1990.

> This paper introduced to the systems community a logic-based notation for authentication protocols to precisely describe certificates, delegations, etc. With this precise description a designer can easily reason whether a protocol is correct or not, and avoid the security flaws that have plagued protocols. "Speaks-for" and "says" are now standard tools for system designers.

Fred B. Schneider, *Implementing Fault-Tolerant Services Using the State Machine Approach: a tutorial*, ACM Computing Surveys 22(4):299-319, December 1990.

> The paper that explained how we should think about replication ... a model that turns out to underlie Paxos, Virtual Synchrony, Byzantine replication, and even Transactional 1-Copy Serializability.

Session Scribe Notes for Twenty-First ACM Symposium on Operating Systems Principles

Editor
Thomas C. Bressoud
Department of Mathematics and Computer Science
Denison University
Granville, OH 43023
bressoud@denison.edu

INTRODUCTION

The following article is divided into nine sections, one for each of the sessions presented at SOSP 2007. For each session, two student volunteers took notes at the conference, capturing the questions and answers following each of the papers presented in that session. Note that the session order follows the program as given at the conference which, due to logistic necessity, differs slightly from the order in the proceedings.

In each section, we give the scribe volunteers authorship credit and then follow with each of the papers. These notes are as captured by the scribe, and have not been reviewed by either the paper presenter nor the delegate questioner for accuracy.

In some cases, the scribes summarized the presentation, and that summary is included. In all cases, we include the papers' abstracts to help set the subject context before the presentation of the questions and answers.

Many thanks to the student scribe volunteers for their efforts and help in capturing this aspect of SOSP 2007.

SESSION 1: WEB MEETS OPERATING SYSTEM

Scribes for this session were Celina Gibbs (University of Victoria) and Alana Libonati (NYU).

Protection and Communication Abstractions for Web Browsers in MashupOS

by Helen J. Wang (Microsoft Research), Xiaofeng Fan (Microsoft Research), Jon Howell (Microsoft Research), and Collin Jackson (Stanford University)

Abstract: *Web browsers have evolved from a single-principal platform on which one site is browsed at a time into a multi-principal platform on which data and code from mutually distrusting sites interact programmatically in a single page at the browser. Today's "Web 2.0" applications (or mashups) offer rich services, rivaling those of desktop PCs. However, the protection and communication abstractions offered by today's browsers remain suitable only for a single-principal system—either no trust through complete isolation between principals (sites) or full trust by incorporating third party code as libraries. In this paper, we address this deficiency by identifying and designing the missing abstractions needed for a browser-based multi-principal platform. We have designed our abstractions to be backward compatible and easily adoptable. We have built a prototype system that realizes almost all of our abstractions and their associated properties. Our evaluation shows that our abstractions make it easy to build more secure and robust clientside Web mashups and can be easily implemented with negligible performance overhead.*

Presentation by **Helen J. Wang**

Greg Minshal, unaffiliated
Q: Has this research area reached an end or are we still exploring the design space?
A: Still in the exploration phase.
Q: What are the open questions?
A: Our solution is complete. But, looking for all browsers to realize the sandbox approach. We don't see the transition path as hard.
Q: What is the future work?
A: The key challenge is to make isolation boundaries flawless. Need to develop tools to detect if browser extensions violate the protection model.

Micah Brodsky, MIT CSAIL
Q: In terms of unauthorized content, why the design choice of "not mine - don't trust" instead of fine-grained authorization?
A: It is up to the integrator to decide. Not necessarily will they all allow the same set. Sometimes you don't care, you just don't want those resources to be accessed. This may differ for different services, the goals of the user for one service may not apply to another service (other integrators may not care). In certain cases you may not want this information revealed.

Joshua Triplett, Portland State University
Q: Have you considered integrating intra-browsers with WHAT-WG standard for cross domain http requests?
A: Yes, this is part of the current work.

AjaxScope: A Platform for Remotely Monitoring the Client-side Behavior of Web 2.0 Applications

by Emre Kiciman (Microsoft Research) and Benjamin Livshits (Microsoft Research)

Abstract: *The rise of the software-as-a-service paradigm has led to the development of a new breed of sophisticated, interactive applications often called Web 2.0. While web applications have become larger and more complex, web application developers today have little visibility into the end-to-end behavior of their systems. This paper presents AjaxScope, a dynamic instrumentation platform that enables cross-user monitoring and just-in-time control of web application behavior on end-user desktops. AjaxScope is a proxy that performs on-the-fly parsing and instrumentation of JavaScript code as it is sent to users' browsers. AjaxScope provides facilities for distributed and adaptive instrumentation in order to reduce the client-side overhead, while giving fine-grained visibility into the code-level behavior of web applications. We present a variety of policies demonstrating the power of AjaxScope, ranging from simple error reporting and performance profiling to more complex memory leak detection and optimization analyses. We also apply our prototype to analyze the behavior of over 90 Web 2.0 applications and sites that use large amounts of JavaScript.*

Presentation by **Emre Kiciman**

Fred Schneider, Cornell University
Q: Did you think about addressing user privacy on the client side?
A: Yes, I thought about it. Everything is limited by the javascript sandbox with no extra privileges. Future work will look at marking sensitive information that will be sent out to the webserver.

Tom Roeder, Cornell University
Q: This seems much like aspect-oriented programming, have you considered using this approach?
A: Yes, currently the implementation uses CSharp to manipulate the AST.

Secure Web Applications via Automatic Partitioning

by Stephen Chong (Cornell), Jed Liu (Cornell), Andrew C. Myers (Cornell), Xin Qi (Cornell), Krishnaprasad Vikram (Cornell), Lantian Zheng (Cornell), and Xin Zheng (Cornell)

Abstract: *Swift is a new, principled approach to building web applications that are secure by construction. In modern web applications, some application functionality is usually implemented as client-side code written in JavaScript. Moving code and data to the client can create security vulnerabilities, but currently there are no good methods for deciding when it is secure to do so.*

Swift automatically partitions application code while providing assurance that the resulting placement is secure and efficient. Application code is written as Java-like code annotated with information flow policies that specify the confidentiality and integrity of web application information. The compiler uses these policies to automatically partition the program into JavaScript code running in the browser, and Java code running on the server. To improve interactive performance, code and data are placed on the client side. However, security-critical code and data are always placed on the server. Code and data can also be replicated across the client and server, to obtain both security and performance. A max-flow algorithm is used to place code and data in a way that minimizes client-server communication.

Presentation by **Stephen Chong**

Marvin Theimer, Amazon
Q: Large programs are typically restructured to use batching. This seems to be missed with your fine grained message approach.
A: Future work of reordering of source for better performance.

Jay Lepreau, University of Utah
Q: Your conclusion is that Swift makes it easy to write these applications, but when sizes reach 70,000 LOC. I am skeptical that this will scale to real applications, unless the model is simple. It would be great if it works.
A: Large applications have no significant problems with performance.
Q: I am not worried about performance, I am worried about human time.
A: Annotations are required, but the number of annotations are proportional to the security concerns of the system. Applications with moderate security concerns will scale.

Petros Maniatis, Intel Research
Q: This has a close connection with label based taint tracking, runtime integrity variables, control paths: is this compatible?
A: We do static analysis, not runtime since runtime analysis may miss implicit flows
Q: Is this too conservative?
A: No it is fine.

Atul Adya, Microsoft
Q: Do you consider server crashes?
A: No, we don't handle fault tolerance.

Emere Kiciman, Microsoft Research
Q: In terms of persistent states, do you have labels to handle state?
A: In the shopping cart example, we are connecting to a mysql database for persistance. Persistence annotations are part of planned future work.

Derrick Coetzee, Microsoft Research
Q: It seems there are a lot of constraints here that must be declaratively specified, isn't this a problem? Isn't it possible to accidentally write a program that cannot be split to client and server partitions? Do you provide any help with this case?
A: One possible solution is to partition everything on the server but, efficiency is harmed.

John Dunagan, Microsoft
Q: How many times do I have to be careful in partitioning? Do I have to think about every variable?

A: Not all variables must be annotated. The compiler deals with inference and not all variables in the application will impact security.

SESSION 2: CONCURRENCY

Scribes for this session were Eric Eide (University of Utah) and Diwaker Gupta (UCSD).

TxLinux: Using and Managing Hardware Transactional Memory in the Operating System

by Christopher J. Rossbach (UT Austin), Owen S. Hoffman (UT Austin), Donald E. Porter (UT Austin), Hany E. Ramadan (UT Austin), Aditya Bhandari (UT Austin), and Emmett Witchel (UT Austin)

Abstract: *TxLinux is a variant of Linux that is the first operating system to use hardware transactional memory (HTM) as a synchronization primitive, and the first to manage HTMin the scheduler. This paper describes and measures TxLinux and discusses two innovations in detail: cooperation between locks and transactions, and the integration of transactions with the OS scheduler. Mixing locks and transactions requires a new primitive, cooperative transactional spinlocks (cxspinlocks) that allow locks and transactions to protect the same data while maintaining the advantages of both synchronization primitives. Cxspinlocks allow the system to attempt execution of critical regions with transactions and automatically roll back to use locking if the region performs I/O. Integrating the scheduler with HTM eliminates priority inversion. On a series of real-world benchmarks TxLinux has similar performance to Linux, exposing concurrency with as many as 32 concurrent threads on 32 CPUs in the same critical region.*

Presentation by **Chris Rossbach**

Ken Birman from Cornell noted that some things are just inherently hard to implement correctly (in particular, long-running and nested transactions). He asked if the authors had observed any problematic locking patterns in their analysis of the Linux code, and whether those occurrences suggest better design patterns for handling certain types of concurrency. The speaker responded in the affirmative, giving the example of long-running transactions in Bonnie++ that can cause overflow in the transactional hardware, leading to performance problems. The `cxspinlock` API can be used to partially work around this problem, by using locks instead of transactions.

Gilles Muller from EMN asked that if one were to start from the ground up, how would one redesign the lock API? That is, instead of coming up with a replacement for spinlocks and xspinlocks (for practical reasons), is an inherently better API possible? Further, moving forward, how much more do developers need to learn about transactional locks? The speaker remarked that undoubtedly a better API was possible, should one design from scratch. He expressed hope that programmers should not have to learn much more in terms of new concepts. However, programmers still need to design and code carefully in order to maximumize performance, and programmers also need better support for system tuning and debugging.

Andrew Black from Oregon State University asked about the evaluation: since all the reported numbers are from simulations, how are they normalized against simulator performance? That is, how much is performance impacted by the parameters of the simulation? In response, a fundamental assumption behind the work was pointed out: that the parameters of the simulation are chosen based on where the architecture and memory systems seem to be headed.

Finally, **Marc Shapiro** from INRIA asked about the cost of making the described changes to the contention manager and some other components of the system. The presenter replied that the unique features of the TM system that they utilized should not be expected to cost much more than basic TM hardware. In some cases, contention manager can indeed get a little complex, but hopefully even in those cases the cost is bounded.

MUVI: Automatically Inferring Multi-Variable Access Correlations and Detecting Related Semantic and Concurrency Bugs

by Shan Lu (University of Illinois), Soyeon Park (University of Illinois), Chongfeng Hu (University of Illinois), Xiao Ma (University of Illinois), Weihang Jiang (University of Illinois), Zhenmin Li (University of Illinois), Raluca A. Popa (MIT), and Yuanyuan Zhou (University of Illinois)

Abstract: *Software defects significantly reduce system dependability. Among various types of software bugs, semantic and concurrency bugs are two of the most difficult to detect. This paper proposes a novel method, called MUVI, that detects an important class of semantic and concurrency bugs. MUVI automatically infers commonly existing multi-variable access correlations through code analysis and then detects two types of related bugs: (1) inconsistent updates – correlated variables are not updated in a consistent way, and (2) multi-variable concurrency bugs – correlated accesses are not protected in the same atomic sections in concurrent programs.*

Presentation by **Shan Lu**

George Candea from EPFL began the discussion by observing that system programmers typically just throw up their hands when they run into false positives, since it is non-trivial to track them down. He wondered how hard it is for programmers to deal with false positives in MUVI. Lu replied that in MUVI, it should be very easy for programmers to recognize false positives, because of the context information in MUVI. Other tools such as RacerX have different sources of false positives, and it is hard to say which class of false positives are easier to track down than others, but in general it should not be too hard. Still, they are trying to reduce the rate of false positives from MUVI.

Bjoern Doebel of TU Dresden addessed the ability of MUVI to handle complex code: multiple branches, functions, and so on. Lu re-emphasized that MUVI does flow-

insensitive analysis, so it lose some precision. In that sense MUVI will be more conservative, and it may miss some conditional correlations.

Finally, **Zhe Zhang** from NC State University noted that variable correlations may not be one-to-one, but might instead be more complex. For instance, there might be a variable which holds the sum of several other variables. Lu pointed out that MUVI breaks multi-variable correlations into two-variable correlations, and even otherwise, their data-mining technique is capable of detecting multi-variable correlations, so this shouldn't pose a problem.

SESSION 3: BYZANTINE FAULT TOLERANCE

Scribes for this session were Francis David (UIUC) and Fabio Oliveira (Rutgers University).

Zyzzyva: Speculative Byzantine Fault Tolerance

by Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong (UT Austin)

Abstract: *We present Zyzzyva, a protocol that uses speculation to reduce the cost and simplify the design of Byzantine fault tolerant state machine replication. In Zyzzyva, replicas respond to a client's request without first running an expensive three-phase commit protocol to reach agreement on the order in which the request must be processed. Instead, they optimistically adopt the order proposed by the primary and respond immediately to the client. Replicas can thus become temporarily inconsistent with one another, but clients detect inconsistencies, help correct replicas converge on a single total ordering of requests, and only rely on responses that are consistent with this total order. This approach allows Zyzzyva to reduce replication overheads to near their theoretical minima.*

Presentation by **Ramakrishna Kotla**

Summary: Ramakrishna started by stating the main aim of the work: to make it easier to incorporate reliability into high-performance distributed systems, without sacrificing performance. In light of the complexity in the design space of BFT-based reliable systems due to the non-obvious tradeoff between throughput and latency, Ramakrishna motivated the need for a new BFT protocol that outperforms existing approaches in terms of both throughput and latency. He then introduced speculative BFT replication as the key insight behind Zyzzyva, the proposed protocol. By means of speculation, the replicas execute the requests with no agreement, relieving the system from the associated overhead.

The Zyzzyva protocol performs the output commit at the clients. Client-side output commit supports the notion of speculative request execution by having the clients determine if the system is consistent. Only after determining that a reply is stable will a client commit. The history of requests sent along with a reply allows the client to verify if the reply is stable. There are 3 cases to consider. (1) If all responses (3f + 1) match, the client can commit. (2) If the client gathers 2f + 1 matching responses, it sends commit certificates to all replicas and commits after receiving 2f + 1 matching ACKs. (3) If the client receives fewer than 2f + 1 matching responses, the client retransmits the request, making the system change its current view. Liveness is guaranteed because correct clients ensure system progress. Safety is guaranteed because faulty clients cannot forge request histories and two valid commit certificates cannot have varying prefixes.

Next, Ramakrishna commented on some optimizations and introduced the Zyzzyva5 protocol, a variant that uses 2f + 1 additional replicas to make the faulty case faster. Then, he showed the results of the evaluation, comparing Zyzzyva to existing protocols. Zyzzyva has a significant throughput improvement. In terms of latency, Zyzzyva outperforms other approaches, but Q/U is 15that is optimal for Q/U. Zyzzyva was shown to approach the optimal performance expected from a BFT protocol, with a performance comparable to that of an unreplicated service.

During the Question and Answer session, **Petros Maniatis**, from Intel, stated that he liked the work. Then, in light of the original Zyzzyva protocol and the Zyzzyva5 variant, he wanted to know what the final word was, i.e., which protocol (Zyzzyva and Zyzzyva5) is better for which? Ramakrishna responded that, in choosing the protocol, one needs to consider the tradeoff between space and time. He said that, in case of failures, Zyzzyva5 is better; otherwise, Zyzzyva is more desirable. He added that they are essentially the same protocol, the only difference being the number of replicas. Petros then asked how should one decide on the number of replicas. Ramakrishna answered that one should start with Zyzzyva and, later on, if there are failures, the system should switch to Zyzzyva5. Another person then asked:"if H/Q overlapped its 1st and 2nd phases, wouldn't its throughput be comparable to that of Zyzzyva" Ramakrishna said that this overlapping is only applicable to Zyzzyva.

Tolerating Byzantine Faults in Database Systems using Commit Barrier Scheduling

by Benjamin Vandiver (MIT), Hari Balakrishnan (MIT), Barbara Liskov (MIT), and Sam Madden (MIT)

Abstract: *This paper describes the design, implementation, and evaluation of a replication scheme to handle Byzantine faults in transaction processing database systems. The scheme compares answers from queries and updates on multiple replicas which are unmodified, off-the-shelf systems, to provide a single database that is Byzantine fault tolerant. The scheme works when the replicas are homogeneous, but it also allows heterogeneous replication in which replicas come from different vendors. Heterogeneous replicas reduce the impact of bugs and security compromises because they are implemented independently and are thus less likely to suffer correlated failures.*

The main challenge in designing a replication scheme for transaction processing systems is ensuring that the different replicas execute transactions in equivalent serial orders while allowing a high degree of concurrency. Our scheme meets this goal using a novel concurrency control protocol, commit barrier scheduling (CBS). We have implemented CBS in *the context of a replicated SQL database, HRDB (Heterogeneous Replicated DB), which has been tested with unmodified production versions of several commercial and open source databases as replicas. Our experiments show an HRDB configuration that can tolerate one faulty replica has only a modest performance overhead (about 17% for the TPC-C benchmark). HRDB successfully masks several Byzantine faults observed in practice and we have used it to find a new bug in MySQL.*

Presentation by **Ben Vandiver**

Summary: Ben motivated the work through the observation that more than half of reported bugs on database systems exhibit a non-crash behavior; therefore, a Byzantine fault model would be more appropriate than the currently assumed fail-stop behavior. Ben pointed out that a heterogeneous database replication scheme, where the replicas run different DBMS versions would be paramount to guaranteeing failure independence among the replicas. The proposed replication scheme, called HRDB (Heterogeneous Replicated DataBase), assumes that the database replicas provide serializable isolation and strict two-phase locking. HRDB is meant to give users the abstraction of a single-copy serializable view of the database.

Next, Ben pointed out the main weakness of traditional solutions to Byzantine Fault Tolerance: globally ordering client requests and having the replicas execute them in the agreed order limits concurrency. He stated that, besides the need for correctness, HRDB must perform replica coordination via a mechanism that is able to extract concurrency. That mechanism is called CBS (Commit Barrier Scheduling).

Ben then described the architecture of HRDB. The clients interact with a front-end, called the shepherd, which coordinates the unmodified back-end database replicas and makes ordering decisions about the client requests. The database replicas vote on the result of the issued SQL queries. The system needs at least f + 1 matching votes, where f is the maximum number of faulty replicas HRDB tolerates. In the sequel, Ben introduced CBS, the proposed scheme for replica coordination. CBS operates under a primary-secondary scheme in which all transactions are first run on the primary. The order at which the queries are executed on the primary is then enforced on the secondary replicas. Correct execution is guaranteed because the statements belonging to a particular transaction are executed in order and all replicas commit transactions in the same order: the one determined by the shepherd. The main goal of CBS in issuing queries to the secondaries is to explore concurrency while avoiding conflicts between queries belonging to different transactions. Only one scenario can create conflicts and prevent the exploitation of concurrency: when a statement from a certain transaction is executed by the primary after the commit of another transaction. CBS guarantees full concurrency at the primary, relying on the available two-phase commit protocol, and allows many statements to run in parallel on the secondaries.

Ben finally talked briefly about the prototype implementation and delved into the evaluation. He showed that the overhead of running SQL queries through the HRDB shepherd was 17% for the TPC-C benchmark. Also, it was shown that a faulty replica is able to catch up with the system after it recovers. HRDB was also able to unveil a new bug in MySQL. Ben concluded by noting that HRDB was a practical BFT database as well as a tool for finding database bugs.

During the Question and Answer session, a person from the audience asked if it would be appropriate to just send queries that do not modify the database to the replicas, instead of waiting the response from the primary. Ben answered that the main concern of the work was with the ordering correctness and that he did not think more about other ways of relaxing the constraints. Next, **Emery Berger**, from the University of Massachusetts, asked how the authors actually found the bugs. Ben's response was that the bugs were found by accident. **Ken Birman**, from Cornell, asked if the primary could possibly prevent the secondaries from making progress. Ben said that this cannot happen; the only impact is an increased latency.

Low-Overhead Byzantine Fault-Tolerant Storage

by James Hendricks (Carnegie Mellon University), Greg Ganger (Carnegie Mellon University), and Mike Reiter (UNC at Chapel Hill)

Abstract: *This paper presents an erasure-coded Byzantine fault-tolerant block storage protocol that is nearly as efficient as protocols that tolerate only crashes. Previous Byzantine fault-tolerant block storage protocols have either relied upon replication, which is inefficient for large blocks of data when tolerating multiple faults, or a combination of additional servers, extra computation, and versioned storage. To avoid these expensive techniques, our protocol employs novel mechanisms to optimize for the common case when faults and concurrency are rare. In the common case, a write operation completes in two rounds of communication and a read completes in one round. The protocol requires a short checksum comprised of cryptographic hashes and homomorphic fingerprints. It achieves throughput within 10% of the crash-tolerant protocol for writes and reads in failure-free runs when configured to tolerate up to 6 faulty servers and any number of faulty clients.*

Presentation by **James Hendricks**

Summary: James Hendricks motivated the need for a Byzantine fault-tolerant storage system by pointing out that, as modern systems grow in size, they should learn to cope with more faults as well as more types of faults. He proceeded to describe an erasure-coded Byzantine fault-tolerant block storage protocol that is almost as efficient as protocols that tolerate only crashes. 3f+1 servers are required to tolerate f failures. This is an improvement over a PASIS, a previous design that required 4f+1 servers. The low overhead in the new protocol is due to the fact that, in the common case, the protocol uses just two rounds of communication for a write and one round of communication for a read. A write request has a prepare and a commit phase. A read request returns fragments and associated timestamps in the same round. The amount of computation is also reduced by only partially encoding a block for most write operations. This involves the creation of 2f+1 fragments, which is the same number encoded by a non-Byzantine fault-tolerant erasure-coded protocol. Since servers don't see the entire block, a technique called homomorphic fingerprinting is used to ensure that a fragment is consistent. Random nonce values provided by servers and aggregated by clients during a write are used to ensure that reads return the most recently written block. The generation and verification of cryptographic data provides an additional source of overhead for write operations and verification of checksums adds to the overhead of read operations. Experimental evaluation shows that larger fragment sizes can be used to keep write bandwidth close to a benign erasure-coded protocol that only tolerates crash faults. Write latency is less than twice that of the benign protocol when tolerating upto 10 faults. Read throughput is close to the benign protocol. Read latency is again within within twice that of the benign protocol.

Allen Clement, UT Austin
Q: Your performance charts show a significant improvement over state machine replication. Is that comparison fair, given your system is only supporting reads and writes, while state machine replication is supporting arbitrary operations?
A: Your observation is correct. One of the points of this work is to show that BFT storage is a weaker problem than state machine replication, and the performance gain is attained by exploiting that difference in strength.

Attested Append-Only Memory: Making Adversaries Stick to their Word

by Byung-Gon Chun (UC Berkeley), Petros Maniatis (Intel Research, Berkeley), Scott Shenker (UC Berkeley), and John Kubiatowicz (UC Berkeley)

Abstract: *Researchers have made great strides in improving the fault tolerance of both centralized and replicated systems against arbitrary (Byzantine) faults. However, there are hard limits to how much can be done with entirely untrusted components; for example, replicated state machines cannot tolerate more than a third of their replica population being Byzantine. In this paper, we investigate how minimal trusted abstractions can push through these hard limits in practical ways. We propose Attested Append-Only Memory (A2M), a trusted system facility that is small, easy to implement and easy to verify formally. A2M provides the programming abstraction of a trusted log, which leads to protocol designs immune to equivocation —the ability of a faulty host to lie in different ways to different clients or servers —which is a common source of Byzantine headaches. Using A2M, we improve upon the state of the art in Byzantine-fault tolerant replicated state machines, producing A2M-enabled protocols (variants of Castro and Liskov's PBFT) that remain correct (linearizable) and keep making progress (live) even when half the replicas are faulty, in contrast to the previous upper bound. We also present an A2M-enabled single-server shared storage protocol that guarantees linearizability despite server faults. We implement A2M and our protocols, evaluate them experimentally through micro- and macro-benchmarks, and argue that the improved fault tolerance is cost-effective for a broad range of uses, opening up new avenues for practical, more reliable services.*

Presentation by **Byung-Gon Chun**

Summary: Byung-Gon Chun started the talk by presenting the central problem addressed by his work: equivocation in the domain of replicated servers providing a service. In such a system, there is a need for linearizability and liveness in the presence of cases where servers may be equivocating to other servers and clients. The goal is to see if the number of servers required in such a Byzantine faulty environment can be reduced to below 3f+1 where f is the number of faults.

A2M (Attested Append-Only Memory) is a primitive that provides a guard against equivocation. A2M provides a set of trusted, undeniable and numbered logs where each entry has a sequence number, a stored value and an incremental cryptographic digest of all log entries. The A2M primitive may be implemented using several approaches: a 3rd party service, software isolation, a virtual machine or using trusted hardware. A2M can be used to improve fault-tolerance by forcing servers to commit to a single history of operations. As a demonstration, a couple of A2M enabled versions of PBFT were built. A2M-PBFT-E only uses A2M in the last execute phase and A2M-PBFT-EA uses A2M during both agreement and execution. A2M-PBFT-E guarantees safety and liveness using 3f+1 replicas, whereas A2M-PBFT-EA requires only 2f+1 replicas. Macro-benchmark experiments with NFS show that the A2M enabled versions of PBFT using MACs for authentication have performance close to that of plain PBFT. Byung-Gon noted that the broader implication of this work is that small trusted primitives can help make systems better.

In response to a question about the strength of the cryptography used, Byung-Gon replied that the key lengths are the same as used in previous research publications. **Robbert van Renesse** of Cornell University pointed out that simulated authenticated broadcasts attempt to solve the same problem.

SESSION 4: SOFTWARE ROBUSTNESS

Scribes for this session were John McCullough (UCSD) and Rohini Prinja (University of Minnesota).

Bouncer: Securing Software by Blocking Bad Input

by Manuel Costa (Microsoft Research), Miguel Castro (Microsoft Research), Lidong Zhou (Microsoft Research), Lintao Zhang (Microsoft Research), and Marcus Peinado (Microsoft)

Abstract: *Attackers exploit software vulnerabilities to control or crash programs. Bouncer uses existing software instrumentation techniques to detect attacks and it generates filters automatically to block exploits of the target vulnerabilities. The filters are deployed automatically by instrumenting system calls to drop exploit messages. These filters introduce low overhead and they allow programs to keep running correctly under attack. Previous work computes filters using symbolic execution along the path taken by a sample exploit, but attackers can bypass these filters by generating exploits that follow a different execution path. Bouncer introduces* *three techniques to generalize filters so that they are harder to bypass: a new form of program slicing that uses a combination of static and dynamic analysis to remove unnecessary conditions from the filter; symbolic summaries for common library functions that characterize their behavior succinctly as a set of conditions on the input; and generation of alternative exploits guided by symbolic execution. Bouncer filters have low overhead, they do not have false positives by design, and our results show that Bouncer can generate filters that block all exploits of some real-world vulnerabilities.*

Presentation by **Miguel Castro**

George Candea, EPFL
Q: How far in the path do you have to go to trace the failure?
A: Some paths are relatively long, depends on attack. The techniques are still useful to find the root cause: Can find the first place where malloc corrupts the memory structures.
Q: What if the exploit manifests out of something longer? The environment - not the inputs - memory overflow?
A: Answer taken offline.

Eric Eide, Utah
Q: Did you consider trading off the false-positive rate for higher throughput?
A: We don't want to give up on no false positives. It is bad if you deploy this automatically and it blocks something inadvertantly. We need to be stringent.

Vikram Adve, UIUC
Q: When you get false negatives, what is it that the filter cannot handle?
A: There can be many paths from where the message is received to where the vulnerability is. If the data format is variable, static analysis gives an overlarge result and it is difficult to simplify the filter to a practical size.

Ken Birman, Cornell University
Q: Is it reasonable to be rejecting messages instead of correcting them? The assumption that your techniques are triggered by an attack is not obvious.
A: We are working on something for correcting input.

Triage: Diagnosing Production Run Failures at the User's Site

by Joseph Tucek (University of Illinois), Shan Lu (University of Illinois), Chengdu Huang (University of Illinois), Spiros Xanthos (University of Illinois), and Yuanyuan Zhou (University of Illinois)

Abstract: *Diagnosing production run failures is a challenging yet important task. Most previous work focuses on offsite diagnosis, i.e. development site diagnosis with the programmers present. This is insufficient for production-run failures. To address production-run failures, we propose a system, called Triage, that automatically performs onsite software failure diagnosis at the very moment of failure. It provides a detailed diagnosis report, including the failure nature, triggering conditions, related code and variables, the fault propagation chain, and potential fixes. Triage achieves this by leveraging lightweight reexecution support to efficiently capture the failure environment and repeatedly replay the moment of failure, and dynamically—using different diagnosis techniques—analyze an occurring failure. Triage employs a failure diagnosis protocol that mimics the steps a human takes in debugging. This extensible protocol provides a framework to enable the use of various existing and new diagnosis techniques. We also propose a new failure diagnosis technique, delta analysis, to identify failure related conditions, code, and variables.*

Presentation by **Joseph Tucek**

Butler Lampson, MSR
Q: What happens if the program doesn't crash, but just returns wrong answers.
A: It is very hard to detect semantic bugs. If you could include them as part of the failure detector and our technique would work.
Q: If you have a wrong answer, you would have to look back further?
A: These are hard, needs more work.

Marvin Theimer, Amazon
Q: You chose a 200ms checkpoint interval. Is this the right length? Hard bugs are ones that started a long time ago.
A: There have been some studies on bug propagation. Most are very short. In practice, we've found that the worst was 2 checkpoints back. If you have a bug that is a very long propagation, you need to think about how to go back further. Alternatively we could write the checkpoints to disk if the propagation is very long.

Preston Crow, EMC
Q: Is there any assumption that you have source code on the customer's site?
A: We do not need the actual source code, we operate on the binary. Could use obfuscated debugging symbols or send back a binary dump.

Dushyanth Narayanan, MSR
Q: You looked only at the code branches, is this the best way to characterize the failing runs? Some have proposed inserting random predicates.
A: We talked about looking differences in the values. Diffing value variations would be possible, but they are more expensive. You could certainly use random predicates as a diagnosing technique to extend Triage.
Q: If you are using data values, are there any privacy issues?
A: It is harder to represent the data in an error report in a transparent way, but it is better than the core dump, which is hard to read and probably contains private information.

Jay Lepreau, Utah
Q: Why was the time for diffing not included as part of the time for the error reports?
A: It is not part of the active diagnostic process.

/* iComment: Bugs or Bad Comments? */

by Lin Tan (University of Illinois), Ding Yuan (University of Illinois), Gopal Krishna (University of Illinois), and Yuanyuan Zhou (University of Illinois)

Abstract: *Commenting source code has long been a common practice in software development. Compared to source code, comments are more direct, descriptive and easy-to-understand. Comments and source code provide relatively redundant and independent information regarding a program's semantic behavior. As software evolves, they can easily grow out-of-sync, indicating two problems: (1) bugs - the source code does not follow the assumptions and requirements specified by correct program comments; (2) bad comments - comments that are inconsistent with correct code, which can confuse and mislead programmers to introduce bugs in subsequent versions. Unfortunately, as most comments are written in natural language, no solution has been proposed to automatically analyze comments and detect inconsistencies between comments and source code.*

This paper takes the first step in automatically analyzing comments written in natural language to extract implicit program rules and use these rules to automatically detect inconsistencies between comments and source code, indicating either bugs or bad comments. Our solution, iComment, combines Natural Language Processing (NLP), Machine Learning, Statistics and Program Analysis techniques to achieve these goals.

Presentation by **Lin Tan**

Ken Birman, Cornell
Q: Do you view this as a successful methodology? If you compare the time the humans spend on the templates compared to finding the bugs manually.
A: You can use our tool with the built-in templates and do not necessarily need to build your own. It can take developers days or months to find these bugs without assistance, the results are still valuable.

Preston Crow, EMC
Q: Is there spell checking on the comments? Ours have lots of typos.
A: We found this as well, but our analysis techniques were not sensitive to these typos.

Josh Triplet, Portland State U
Q: Would code annotations help this technique?
A: The annotations make the analysis much easier. iComment is important in their absence.

Butler Lampson, MSR
Q: Can you turn it around and generate code annotations from comments?
A: That is future work.

SESSION 5: DISTRIBUTED SYSTEMS

Scribes for this session were Medha Bhadkamkar (Florida International University) and Kiran Muniswamy-Reddy (Harvard University).

Sinfonia: A New Paradigm for Building Scalable Distributed Systems

by Marcos K. Aguilera (HP Labs), Arif Merchant (HP Labs), Mehul Shah (HP Labs), Alistair Veitch (HP Labs), and Christos Karamanolis (VMware)

Abstract: *We propose a new paradigm for building scalable distributed systems. Our approach does not require dealing with message-passing protocols – a major complication in existing distributed systems. Instead, developers just design and manipulate data structures within our service called Sinfonia. Sinfonia keeps data for applications on a set of memory nodes, each exporting a linear address space. At the core of Sinfonia is a novel minitransaction primitive that enables efficient and consistent access to data, while hiding the complexities that arise from concurrency and failures. Using Sinfonia, we implemented two very different and complex applications in a few months: a cluster file system and a group communication service. Our implementations perform well and scale to hundreds of machines.*

Presentation by **Marcos Aguilera**

Butler Lampson, MSR
Q: It seems like your scheme would work just as well if you allowed a minitransaction to be a more or less arbitrary program as long as it has a reasonable bounded execution time and doesn't write any permanent data except at the end. Is there some reason you did not do that?
A: We discuss this in paper. For example, you could implement functionality that doesn't cross a memory node. For example, We can trace pointers within memory nodes. If have to trace pointers across memory nodes, then you have to add network round trips to that. The functionality may be desirable for some apps. The incremental workload that I described here has problems with contention. If you implement increment at the memory node then you don't have that. We've explained this in the paper, we didn't have to use this added functionality for the apps we built.

Marc Shapiro, INRIA
Q: Are there any benefits if you had designed it to be Append-only memory instead of random access memory. You mentioned that it was hard to design data structures in the GCS case. If you had append only memory, it wouldn't be the case.
A: Yes, I guess there are some issues with garbage collection with append only memory. If you don't have to have to worry about that, then you could use append only memory. But we haven't thought about about using append-only memory.

Ken Birman, Cornell University
Q: What do you do when you have a lot of contention? You could have exponential rollback. Have you looked at scenarios like that?
A: Sinfonia doesn't perform well under contention. There is no silver bullet for dealing with contention in large systems. In the Group Communication Service we built, we seem to have a reasonable amount of scalability over what exists currently, which is spread toolkit, not perfect scalability but reasonable scalability.
Q: What would have happened if you have turned on IP multicast for spread?
A: I think that spread would have performed an order of magnitude better, but didn't run the test as we it was not in our environment to support that.

Bryan Ford, MIT
Q: You seem to propose an alternative to message passing, but I don't see any mechanism for notifications?
A: Notifications are good if you are trying to improve latency, otherwise, polling is good. It turns that we are trying to optimize for throughput instead of latency and we don't mind polling not so frequently and that's we did in our GCS application, which is really not optimized for latency.

Dutch Meyer, UBC
Q: : What about the application space made you go with mini transactions as opposed to Federated array of bricks with quorum?
A: FAB is a storage component built out of commodity components and you don't have a notion of transactions. If the same storage component is being accessed by different machines, there is no way to orchestrate a synchronized access as there are no transactions, whereas in sinfonia, you can orchestrate concurrent access.
Q: Is there a big principled reason why transactions are better than a quorum protocol?
A: Quorum mechanisms and mini transactions are not to be compared side to side. They are different mechanisms to do different things.

PeerReview: Practical Accountability for Distributed Systems

by Andreas Haeberlen (MPI-SWS), Petr Kouznetsov (MPI-SWS), and Peter Druschel (MPI-SWS)

Abstract: *We describe PeerReview, a system that provides accountability in distributed systems. PeerReview ensures that Byzantine faults whose effects are observed by a correct node are eventually detected and irrefutably linked to a faulty node. At the same time, PeerReview ensures that a correct node can always defend itself against false accusations. These guarantees are particularly important for systems that span multiple administrative domains, which may not trust each other.*

PeerReview works by maintaining a secure record of the messages sent and received by each node. The record is used to automatically detect when a node's behavior deviates from that of a given reference implementation, thus exposing faulty nodes. PeerReview is widely applicable: it only requires that a correct node's actions are deterministic, that nodes can sign messages, and that each node is periodically checked by a correct node. We demonstrate that Peer- Review is practical by applying it to three different types of distributed systems: a network filesystem, a peer-to-peer system, and an overlay multicast system.

Presentation by **Andreas Haeberlen**
Indranil Gupta, UIUC
Q: How do you select Witnesses? How do you prevent nodes from colluding among each other?
A: In NFS example, the membership was static and we used a static config file that mapped every node to its witness. We used consistent hashing to choose a witness. In distributed email system, we could not do this as the membership was dynamic. There, we used consistent hashing but we resolve it dynamically using secure routing. That ensures that you cannot pick your witness nodes yourself.
Q: What is to prevent a node's friend from doing a sybil kind of attack, where it tries out all port numbers until it becomes a witness?
A: We assume that we have a solution to sybil attacks.

Robbert Van Renesee, Cornell
Q: How do you deal with nodes that selectively talk to some nodes and not to the others?
A: Suppose I ignore a message from you, but talk to others, you would generate verifiable evidence against me. You would give this evidence to my witness nodes, then my witness nodes would give it to everyone else and it would also come to me and challenge me and say "please acknowledge that you received this message". If I did not get the original message due to communication problem, I can acknowledge that got the message. But if I am malicious and don't reply, all other nodes will know that I am malicious and will stop communicating with me. So I have no choice but to talk to other people.

Jay Lepreau, Utah
Q: Because of the requirement of another instance of the RSM, it seems like it will be useful for malicious adversary instead of general faults. The question is of Wide applicability. Is this work is useful for failing H/W and S/W where you can't or is not practical to replicate. The state machine, it'll have the same fault.
A: They decided to take it offline.

Jeff Chase, Duke
Q: The cats system which does strong accountability for a particular system - network storage. You said, unlike peer review, cats depends on a trusted publishing medium for the integrity of the state logs. For clarification, each node requires access to other node's state log. But the publishing medium doesn't have to be central and it isn't trusted. It can mount a denial of accountability attack but it can't subvert the system. It seems like witnesses play the same role and the replication of witness is important. How do you make deal with the assumption of making the logs visible in a secure way?
A: This is solved by the requirement that every single witness set has one correct node. This correct node will make the evidence available to other nodes in the system. It really depends on your failure assumption. How many nodes do you need for this witness set, what kind of config do you have. In a general setting, you might use a probabilistic argument to see how large the witness set has to be. If you have some special nodes, you might use those.

Milan Milenkovic, Intel
Q: You cannot figure out if the fault is with the sender or receiver if a message not received. There were some papers in the 70s regarding digital non-repudiation that addressed this. Why couldn't you use this?
A: Answer taken offline.

Dynamo: Amazon's Highly Available Key-Value Store

by Guiseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swami Sivasubramanian, Peter Vosshall, and Werner Vogels (Amazon.com)

Abstract: *Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.*

This paper presents the design and implementation of Dynamo, a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience. To achieve this level of availability, Dynamo sacrifices consistency under certain failure scenarios. It makes extensive use of object versioning and application-assisted conflict resolution in a manner that provides a novel interface for developers to use.

Presentation by **Peter Vosshall**

Mike Schroeder, MSR
Q: When failure occurs in some of the nodes, it adds load. What kind of MTTF is needed to avoid loading up parts of keyspace so that that part of the key space becomes unavailable?
A: We do have guidelines to over provision to accommodate typical failure scenarios (not sure I can give out the exact formulas). We sufficiently over provision.

Bill Bolosky, MSR
Q: There's an interaction between 99.9 SLA and adding a node: you got to make all nodes have their load reduced rather than just a few. Do you chop it up into so many pieces so that if you add one node it takes load off everybody?
A: By introducing virtual nodes into the system, when you add a new node, it takes many positions on the ring and it ends up having a neighbor relationship with many other nodes in the system. Instead of transferring load from one node, you're pulling small segments of load from all the nodes with a high probability.
Q: With a whole lot of small database files that you have to split things into?
A: Yes.

Goeff Kuenning, Harvey Mudd College
Q: When a node recovers, you use eta entropy, do you use log based or comparison based method? Why?
A: There's a number of mechanisms. There's hinted handoff: when there are failures, writes spill over to another node. When the node recovers, the spill over data is handed off to the correct node. Another technique is to use merkel hash trees to do replica synchronization between the two nodes.

George Candea, EPFL
Q: At this scale, how do you collect the data, analyze, and verify and prove to your users that you met the three 9's?
A: We have a monitoring infrastructure to do all for all our services, but I was not involved with it, so I don't know the details but everyone does use it verify if the SLA is being met or not.
Q: Do you know if you sample or not?
A: I don't know.

Ashvin Goel, University of Toronto
Q: What kind of conflicts do you see and what kind of conflict management do you have to do?
A: There's some results in the paper about the rate at which we are seeing multiple results. 99.94% of reads return a single value, so there's not conflict. The other times, reads were returning two versions of the value. We don't know if they are conflicts or retries of the same write.

Chris Lesniewski-Laas, MIT
Q: How often do you miss your write quorum?
A: We have not had failures where we fail to gather a write quorum.
Q: Do you ever have a partition where you have only one node on one side?
A: That's possible ... if a switch on a rack fails.
Q: Would a client ever see that?
A: The client would not see that.

David Anderson, CMU
Q: Trying to understand why you abandoned pure consistent hashing. You said it took too long for the seeks to do replication. That seems like a classic seek vs scan tradeoff in a database. So why didn't you partition your data into relatively coarse extents on disk and simply scan them sequentially?
A: We're not claiming that we did the best thing, we just did something.

Andrew Myers, Cornell
Q: Trying to understand the programming model. How do clients do a good job of merging conflicts, because it seems like you are not giving the common ancestors of the versions back, which things like cvs or rcs really need to do a good job of merging?
A: yes, its hard for apps. Depending on the application, how they approach it can vary. Carts already use the model and were already good at reconciling conflicts. From an application perspective, that's sufficient. History might help in some cases. other applications we might punt and use a last write wins rule. You can tune your dynamo cluster to do that for you. We also thought about keeping all versions around and garbage collect them later on, so we could present previous versions, but we didn't need it for the apps we were considering. But, if the apps really wanted it, we could think about doing that.
Q: Are there guarantees that you will eventually converge on one version?
A: There are no guarantees in life, but we believe that the system will converge.

Emit Sit, MIT
Q: Why did you use consistency hashing vs. a GFS like approach that has a random set of nodes and a coordinator?

A: We wanted extremely high availability. We wanted a symmetric system where there was no master. The decentralized that you got from consistent hashing was very compelling. You could add a node and it could be its own master for a particular read/write operation and also have their own view of the system. That was the main reason.

Hari Balakrishnan, MIT
Q: This is regarding load balancing. What happens if some keys are more popular than others and the distribution is heavily skewed and do you see that in real applications?
A: In some apps, we don't really see it so often. Even if there are robots, it falls away into noise as there are so many users. In some apps, you do see it and it is bit of a challenge. Unless you have a control over how keys are placed, that could be a challenge. Because we control over how nodes are placed, we could start controlling how that load impacts the system.
Q: So you have log(n) virt nodes and any given key is replicated that many times?
A: Yes.

SESSION 6: SYSTEM MAINTENANCE

Scribes for this session were Purvi Shah (University of Houston) and Qing Zhang (UCSD).

Staged Deployment in Mirage, an Integrated Software Upgrade Testing and Distribution System

by Olivier Crameri (EPFL), Nikola Knezevic (EPFL), Dejan Kostic (EPFL), Ricardo Bianchini (Rutgers), and Willy Zwaenepoel (EPFL)

Abstract: *Despite major advances in the engineering of maintainable and robust software over the years, upgrading software remains a primitive and error-prone activity. In this paper, we argue that several problems with upgrading software are caused by a poor integration between upgrade deployment, user-machine testing, and problem reporting. To support this argument, we present a characterization of software upgrades resulting from a survey we conducted of 50 system administrators. Motivated by the survey results, we present Mirage, a distributed framework for integrating upgrade deployment, user-machine testing, and problem reporting into the overall upgrade development process. Our evaluation focuses on the most novel aspect of Mirage, namely its staged upgrade deployment based on the clustering of user machines according to their environments and configurations. Our results suggest that Mirage's staged deployment is effective for real upgrade problems.*

Presentation by **Olivier Crameri**

Bill Bolosky, MSR
Q: Adversaries tend to design exploits based on released patches. Hence there is an urgency to get the patches out asap. Does this system pose any limitations by extending the amount of time the patches are released until it reaches the users?
A: We give the control to the vender for the speed of deployment. If time is crucial the vendors can bypass some of the stages. Trade off between speed and the usibility of the users of the updates.

Joshua Triplett, Portland State University
Q: Finger printing between different clusters Can you apply fingerprinting to note differences between clusters, to see why updates work for one and not another? What if representative in a cluster fails? Also, can you diff them for debugging purposes?
A: Yes, our future work is to diff different clusters to see why how they differ and what the similarities are for debugging.

Micah Brodsky, MIT
Q: Are clusters non-overlapping?
A: Yes, two clusters can have same problems The goal is that the machines in the same cluster behave the same.
Q: With the possible combinations of problems, it.s a combinatorial problem and wouldn.t there be a risk of explosion of clusters?
A: We try to study the population of applications and limit the update by configurations that make sense. We want the update, which depend on the specific application.

Daniel Peek, Michigan
Q: Are there other ways of evaluate the system?
A: Our paper evaluates the system in one way, but there are other avenues of evaluation techniques that we can deploy.

AutoBash: Improving Configuration Management with Operating System Causality Analysis

by Ya-Yunn Su (University of Michigan), Mona Attariyan (University of Michigan), and Jason Flinn (University of Michigan)

Abstract: *AutoBash is a set of interactive tools that helps users and system administrators manage con gurations. AutoBash leverages causal tracking support implemented within our modi ed Linux kernel to understand the inputs (causal dependencies) and outputs (causal effects) of configuration actions. It uses OS-level speculative execution to try possible actions, examine their effects, and roll them back when necessary. AutoBash automates many of the tedious parts of trying to fix a misconfiguration, including searching through possible solutions, testing whether a particular solution fixes a problem, and undoing changes to persistent and transient state when a solution fails. Our results show that AutoBash correctly identifies the solution to several CVS, gcc cross-compiler, and Apache configuration errors. We also show that causal analysis reduces AutoBash's search time by an average of 35% and solution verification time by an average of 70%.*

Presentation by **Ya-Yunn Su**

Jacob Hansen, VMware
Q: I like the part of rollback transactions. Have you considered adding transactional packages to Linux. E.g., Integrating redo into Linux.

A: Yes, rolling back would be useful. It will provide versioning management.

Michael Swift, University of Wisconsin
Q: How relevant would it work in distributed file system environment?
A: We don.t handle distributed system yet. We are thinking about extending our work to support distributed systems.
Q: How far back can you go with the predicates. How incrementally deployable is this system?
A: The users will add their own predicates. It will be good to have a database of predicates.

John Wilkes, HP labs
Q: Writing predicates is a real pain, takes forever. Can we look at applications automatically generate predicates similar to installation system. Where else do you get these predicates from?
A: The vendor will have these predicates, and they can provide them to the users. How will the SW vendor provides these predicates.

Micah Brodsky, MIT
Q: I think this is a really cool system. How well would it work with persistent state transactions. How does it compare with speculative systems?
A: The benefit from doing the rollback in memory is more flexible, and you wouldn.t roll back to have bad data in a persistent state.

Derrick Coetzee, MSR
Q: Can we provide diagnostic information as oppose to just fail/pass. Infer information about what is fixed and what failed from the predicates. It would help users to debug configuration problems.
A: We can make it into a more interactive tool.

Josh Triplett, Portland State
Q: What happens if you need to apply multiple solutions? Can you deploy multiple solutions?
A: Currently we can only try one solution at a time. In the future we want to try solutions which allow multiple deployments of predicate.

SESSION 7: ENERGY

Scribes for this session were Sarah Diesburg (Florida State University) and Jan Stoess (University of Karlsruhe).

Integrating Concurrency Control and Energy Management in Device Drivers

by Kevin Klues (Stanford University, Washington University in St. Louis, Technical University of Berlin), Vlado Handziski (Technical University of Berlin), Chenyang Lu (Washington University in St. Louis), Adam Wolisz (Technical University of Berlin, University of California Berkeley), David Culler (Arch Rock Co., University of California Berkeley), David Gay (Intel Research Berkeley), and Philip Levis (Stanford University)

Abstract: *Energy management is a critical concern in wireless sensornets. Despite its importance, sensor network operating systems today provide minimal energy management support, requiring applications to explicitly manage system power states. To address this problem, we present ICEM, a device driver architecture that enables simple, energy efficient wireless sensornet applications. The key insight behind ICEM is that the most valuable information an application can give the OS for energy management is its concurrency. Using ICEM, a low-rate sensing application requires only a single line of energy management code and has an efficiency within 1.6% of a hand-tuned implementation. ICEM's effectiveness questions the assumption that sensornet applications must be responsible for all power management and sensornets cannot have a standardized OS with a simple API.*

Presentation by **Kevin Klues**

Matt Welsh from Harvard University stated that this work was really about amoritizing power when powering on or off devices. In general, though, it would be nice if the applications would be somewhat adaptive to their circumstances. Can the applicationss adapt their behavior? Kevin responded that they thought about ways to allow applications to submit application requests. There has been work done on an energy management system called "Concurrency", and they have thought about combining these tools toghether.

Mike Swift from The University of Wisconsin asked if there are any lessons that apply to main-stream OSes such as Linux or Windows. Kevin responded that right now, there would be nothing on stage for general-purposes OSes. They had plans for later integrating their approach into OSes for embedded systems such as Symbian, to make, e.g., mobile phones more energy efficient.

Mike then followed up by asking how the programming model would change for programmers implementing this approach. Kevin responded that the proposed operations are very similar to the common notion of asynchronous I/O. Still, it would require application developers to somewhat change their programming paradigms. It would be nice to have a scheme to allow applications to use the energy management mechanisms if possible, but still allow the application to proceed with lower energy efficiency if not.

Mike finally asked if one could come up with a transaction-style language construct for batching all these I/O requests directly, and if one could leverage work combining events and threads? Kevin answered that they hadn't thought about that much, but he wouldn't see anything preventing an API.

Milan Milenkovic from Intel finally noted that, given a send operation via the radio was particularly expensive, one could apply special tricks such as to reduce overhead via data compression, e.g., as soon as a certain threshold has been reached. Kevin stated that all they had specified was the architecture and that you could use that architecture very well to specify different implementations of power compenents. The power lock semantics weren't really limited to the three interfaces presented, and one could imagine special policies for different components and hardware devices.

VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems

by Ripal Nathuji (Georgia Institute of Technology) and Karsten Schwan (Georgia Institute of Technology)

Abstract: *Power management has become increasingly necessary in large-scale datacenters to address costs and limitations in cooling or power delivery. This paper explores how to integrate power management mechanisms and policies with the virtualization technologies being actively deployed in these environments. The goals of the proposed VirtualPower approach to online power management are (i) to support the isolated and independent operation assumed by guest virtual machines (VMs) running on virtualized platforms and (ii) to make it possible to control and globally coordinate the effects of the diverse power management policies applied by these VMs to virtualized resources. To attain these goals, VirtualPower extends to guest VMs 'soft' versions of the hardware power states for which their policies are designed. The resulting technical challenge is to appropriately map VM-level updates made to soft power states to actual changes in the states or in the allocation of underlying virtualized hardware. An implementation of VirtualPower Management (VPM) for the Xen hypervisor addresses this challenge by provision of multiple system-level abstractions including VPM states, channels, mechanisms, and rules. Experimental evaluations on modern multicore platforms highlight resulting improvements in online power management capabilities, including minimization of power consumption with little or no performance penalties and the ability to throttle power consumption while still meeting application requirements. Finally, coordination of online methods for server consolidation with VPM management techniques in heterogeneous server systems is shown to provide up to 34% improvements in power consumption.*

Presentation by **Ripal Nathuji**

Petros Maniatis from Intel research noted that one of the design choices of their approach had been a high-level requirement to remain transparent to applications. He asked if and how much adherence to that choice would create overhead. Ripal responded that they were looking into that problem, and how it could possibly be mitigated using lightweight paravirtualization.

Milan Milenkovic from Intel stated that another possible source to save power would be the consolidation across nodes, which could be established by connecting the power management subsystem with the cross-node management subsystem. Ripal responded that their work evaluated exactly that scenario, and that their ultimate goal was to do this dynamically. In this context they would alleviate a need for low workload analysis by getting the information on the fly.

SESSION 8: STORAGE

Scribes for this session were Vishakha Gupta (Georgia Institute of Technology) and Swapnil Patil (Carnegie Mellon).

DejaView: A Personal Virtual Computer Recorder

by Oren Laadan, Ricardo A. Baratto, Dan Phung, Shaya Potter, and Jason Nieh (Columbia University)

Abstract: *As users interact with the world and their peers through their computers, it is becoming important to archive and later search the information that they have viewed. We present DejaView, a personal virtual computer recorder that provides a complete record of a desktop computing experience that a user can playback, browse, search, and revive seamlessly. DejaView records visual output, checkpoints corresponding application and file system state, and captures displayed text with contextual information to index the record. A user can then browse and search the record for any visual information that has been displayed on the desktop, and revive and interact with the desktop computing state corresponding to any point in the record. DejaView combines display, operating system, and file system virtualization to provide its functionality transparently without any modifications to applications, window systems, or operating system kernels. We have implemented DejaView and evaluated its performance on real-world desktop applications. Our results demonstrate that DejaView can provide continuous low-overhead recording without any user noticeable performance degradation, and allows browsing, search and playback of records fast enough for interactive use.*

Presentation by **Oren Laadan**

Hank Levy, U of Washington
Q: What state is not available when you go back and revive, and how does it impact the user?
A: When you revive the computer, the main piece missing is the network. We have to look at network for different application; whatever connection a user had before for state full protocol is lost but stateless protocols dont care. We don't want to auto revive the network access because it is better that network access be handled on a per-application basis (e-mail apps use network differently from browsing).

Geoff Keunning, Harvey Mudd College
Q: Rather odd level of saving information. When working with Photoshop, I care about not about what I see on the screen but the complex information underneath that is more important...same for video, dvd insertion would be more important. Then why this instead of more application specific?
A: Two reasons: First, our systems can search and extract information displayed on screen is mainly textual information. Not aware of tools for searching video and images. And if you have to search through text, then you need to capture text. Second, if you have to work through application, then have to build very specific features for every application. E.g. for photoshop, I might want to remember when I was working on photoshop and what was in a browser. I could use these hints to revive the data later.

Winfried Kuehnhauser, TUI
Q: What about privacy and confidentiality?
A: Not the focus of current paper – but future work. In current implementation, we don't capture key board input, so things like passwd but show ***. Not done much beyond basic in this aspect.

Cynthia Taylor, UCSD
Q: How do you handle video when you are capturing video?

A: The display recording is based on previous SOSP paper. We only save display updates. In case of full screen video play, we have about 30 display updates which is entire screen and is one single command that is saved.

Improving File System Reliability with I/O Shepherding

by Haryadi S. Gunawi (University Wisconsin - Madison), Vijayan Prabhakaran (Microsoft Research), Swetha Krishnan (University Wisconsin - Madison), Andrea C. Arpaci-Dusseau (University Wisconsin - Madison), and Remzi H. Arpaci-Dusseau (University Wisconsin - Madison)

Abstract: *We introduce a new reliability infrastructure for file systems called I/O shepherding. I/O shepherding allows a file system developer to craft nuanced reliability policies to detect and recover from a wide range of storage system failures. We incorporate shepherding into the Linux ext3 file system through a set of changes to the consistency management subsystem, layout engine, disk scheduler, and buffer cache. The resulting file system, CrookFS, enables a broad class of policies to be easily and correctly specified. We implement numerous policies, incorporating data protection techniques such as retry, parity, mirrors, checksums, sanity checks, and data structure repairs; even complex policies can be implemented in less than 100 lines of code, confirming the power and simplicity of the shepherding framework. We also demonstrate that shepherding is properly integrated, adding less than 5% overhead to the I/O path.*

Presentation by **Haryadi Gunawi**

Preston Crow, EMC
Q: If you want to have policy propagate errors back to applications, the file system ignores error, and you are wedged in beneath the FS, there is no code path to send error back. How do you handle that?
A: Currently working on it. There are certain issues in correctly guaranteeing consistency and error propagation. Currently we are doing static analysis to propagate to the error to application (work under submission)

Colin Dixon, Univ of Washington
Q: In practice, when you run into reliability problem, things seem to have different, unknown effects. How do account for that?
A: There is not much work done right now on how individual block failures occur. There is a paper in Sigmetrics which has statistics on how individual block failures happen. We depend on such data for showing how policies can improve reliability.

Bill Bolosky, MSR
Q: When you are doing mirroring, are you writing on one copy, complete the write, and continue letting the others run asynchronously?
A: When we write to block D and replicate it to R, we ask file system to allocate block R to shepherd and shepherd can synchronously write to D&R in parallel. The checkpoint is success, if both blocks are written to disk. This causes lots of changes, like modified bitmap, modified shepherd data etc., that need chained transactions. So all these state changes in the memory happen with respect to policies for checkpoint. And to guarantee that these changes complete successfully, we need transaction semantics.
Q: By doing that do you get good IO parallelism by avoiding to seek back and forth on the disk?
A: We have tested several benchmarks till now; so far, its not that prohibitive. In general reliability policies are not very much in line with performance. For instance, usually file systems preallocate stuff and you cannot allocate too much. I agree with you that performance might not be good always and it depends on workload.

Amit Gud, VMware
Q: Have you thought of the IO shepherd not having the need to know disk layout of file system?
A: It depends on what policies you want to support. For example, if you want to repair inode or sanity checking at file system level, not at shepherd layer, then shepherd doesn't need to know the layout but then there is a problem of policy. As a file system developer, if you have policies accordingly, then it is better for the FS developed to provide this support with the shepherd.

Generalized File System Dependencies

by Christopher Frost (UCLA), Mike Mammarella (UCLA), Eddie Kohler (UCLA), Andrew de los Reyes (Google), Shant Hovsepian (UCLA), Andrew Matsuoka (UT Austin), and Lei Zhang (Google)

Abstract: *Reliable storage systems depend in part on "write-before" relationships where some changes to stable storage are delayed until other changes commit. A journaled file system, for example, must commit a journal transaction before applying that transaction's changes, and soft updates and other consistency enforcement mechanisms have similar constraints, implemented in each case in systemdependent ways. We present a general abstraction, the patch, that makes write-before relationships explicit and file system agnostic. A patch-based file system implementation expresses dependencies among writes, leaving lower system layers to determine write orders that satisfy those dependencies. Storage system modules can examine and modify the dependency structure, and generalized file system dependencies are naturally exportable to user level. Our patch-based storage system, Featherstitch, includes several important optimizations that reduce patch overheads by orders of magnitude. Our ext2 prototype runs in the Linux kernel and supports asynchronous writes, soft updates-like dependencies, and journaling. It outperforms similarly reliable ext2 and ext3 configurations on some, but not all, benchmarks. It also supports unusual configurations, such as correct dependency enforcement within a loopback file system, and lets applications define consistency requirements without micromanaging how those requirements are satisfied.*

Presentation by **Christopher Frost**

Marc Shapiro, INRIA
Q: What is your undo data?
A: Copy of the data before patch is created (these are file-system agnostic)

Q: How does this work when several applications are writing to the same block?
A: Get layer of patches; write() are atomic w.r.t to other applications.

Petros Maniatis, Intel Research
Q: Comparison between your xsyncfs and ext3. The performance approach in ext3 under the two approaches are different . Where are the differences in performance?
A: Two levels of difference. One, featherstitch with patches can used to implement provide xsyncfs and take advantage of several optimizations that patches provide. On the other hand, xsyncs can provide similar guarantees with patch groups, with a simpler API; but PG provides additional flexibility by not requiring to implement features other techniques.

Surendra Verma, Microsoft
Q: How will patches help build a better file system? For instance, how will patches help improve journaling? Not sure if IO gets better in the way it goes down to the disk as opposed to journal where data can go most optimally? For instance, It's hard to order the I/Os in your system.
A: Patches can be used to implement a model like journal – exact same block writes like journal. In constructing a new system, they can help reduce complexity and ease developing new consistency models. Our approach is a good starting point for new file system for applications to have more control on consistency models.

SESSION 9: OPERATING SYSTEM SECURITY

Scribes for this session were Yu-En Lu (University of Cambridge) and Xiaohui Yang (George Mason University).

Information Flow Control For Standard OS Abstractions

by Maxwell Krohn (MIT), Alex Yip (MIT), Micah Brodsky (MIT), Natan Cliffer (MIT), M. Frans Kaashoek (MIT), Eddie Kohler (UCLA), and Robert Morris (MIT)

Abstract: *Decentralized Information Flow Control (DIFC) is an approach to security that allows application writers to control how data flows between the pieces of an application and the outside world. As applied to privacy, DIFC allows untrusted software to compute with private data while trusted security code controls the release of that data. As applied to integrity, DIFC allows trusted code to protect untrusted software from unexpected malicious inputs. In either case, only bugs in the trusted code, which tends to be small and isolated, can lead to security violations.*

We present Flume, a new DIFC model and system that applies at the granularity of operating system processes and standard OS abstractions (e.g., pipes and file descriptors). Flume eases DIFC's use in existing applications and allows safe interaction between conventional and DIFC-aware processes. Flume runs as a user-level reference monitor on Linux. A process confined by Flume cannot perform most system calls directly; instead, an interposition layer replaces system calls with IPC to the reference monitor, which enforces data flow policies and performs safe operations on the process's behalf. We ported a complex Web application (MoinMoin wiki) to Flume, changing only 2% of the original code. The Flume version is roughly 30-40% slower due to overheads in our current implementation but supports additional security policies impossible without DIFC.

Presentation by **Max Krohn**

Ki Yung Ahn, Portland State
Q: On the example showing how endpoints are made, and secrecy labels may be changed, what if a malicious plugin attempts to perform declassification?
A: This isn't a problem, because, if the secrecy were critical to other applications, it wouldn't be able to declassify the tag (because the tag would have been created by another application). Untrusted software would not have the elevated privilege to declassify tags.

Unidentified, Unknown
Q: Can the described policies be implemented inside SELinux?
A: For a single application, probably yes, but the policies would become complex if it were possible to add applications to the change. Fluke simplifies system administration and transfers these decisions to the content provider.

SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes

by Arvind Seshadri, Mark Luk, Ning Qu, and Adrian Perrig (Carnegie Mellon University)

Abstract: *We propose SecVisor, a tiny hypervisor that ensures code integrity for commodity OS kernels. In particular, SecVisor ensures that only approved code can execute in kernel mode over the entire system lifetime. This protects the kernel against code injection attacks, such as kernel rootkits. SecVisor can achieve this property even against an attacker who controls everything but the CPU, the memory controller, and system memory. Further, SecVisor the attacker could have the knowledge of zero-day kernel exploits.*

Our design goals for SecVisor are small code size, small external interface, and ease of porting OS kernels. We rely onmemory virtualization to build SecVisor and implement two versions, one using software memory virtualization and the other using CPU-supported memory virtualization. The code sizes of the runtime portions of these versions measure 1739 and 1112 lines, respectively. The size of the external interface for both versions of SecVisor is 2 hypercalls. We also port the Linux kernel version 2.6.20 to execute on SecVisor. This requires us to add 12 lines of code to the kernel and delete 81 lines, out of a total of approximately 4.3 million lines of code.

Presentation by **Arvind Seshadri**

Andrea Bittau, University College London
Q: Many past kernel exploits change the uid of user processes to overwrite kernel memory; does SecVisor address

this?
A: It doesn't protect the integrity of kernel data structures. This is future work.

Christos Karamanolis, VMware
Q: Performance effect of nested page tables?
A: Shadow overhead goes away, but you still have to check and protect the kernel page tables.

Mike Schroeder, Microsoft Research
Q: Knowing everything you know about the system, how would you attack it as an intruder?
A: Try to attack the Trusted Computing Base, but it is very small, so we don't expect that this will be a problem.

Andy Tucker, VMware
Q: Deal with loadable modules?
A: Paper discusses this: you could have a hash-based approval policy, and SecVisor performs code relocation on behalf of the kernel.

Richard Black, Microsoft Research
Q: How to protect the stable storage off which SecVisor is loaded?
A: Use the same hash technique and store it in PCR block.

Secure Virtual Architecture: A Safe Execution Environment for Commodity Operating Systems

by John Criswell (University of Illinois), Andrew Lenharth (University of Illinois), Dinakar Dhurjati (DoCoMo Labs USA), and Vikram Adve (University of Illinois)

Abstract: *This paper describes an efficient and robust approach to provide a safe execution environment for an entire operating system, such as Linux, and all its applications. The approach, which we call Secure Virtual Architecture (SVA), defines a virtual, low-level, typed instruction set suitable for executing all code on a system, including kernel and application code. SVA code is translated for execution by a virtual machine transparently, offline or online. SVA aims to enforce fine-grain (object level) memory safety, control-flow integrity, type safety for a subset of objects, and sound analysis. A virtual machine implementing SVA achieves these goals by using a novel approach that exploits properties of existing memory pools in the kernel and by preserving the kernel's explicit control over memory, including custom allocators and explicit deallocation. Furthermore, the safety properties can be encoded compactly as extensions to the SVA type system, allowing the (complex) safety checking compiler to be outside the trusted computing base. SVA also defines a set of OS interface operations that abstract all privileged hardware instructions, allowing the virtual machine to monitor all privileged operations and control the physical resources on a given hardware platform. We have ported the Linux kernel to SVA, treating it as a new architecture, and made only minimal code changes (less than 300 lines of code) to the machine-independent parts of the kernel and device drivers. SVA is able to prevent 4 out of 5 memory safety exploits previously reported for the Linux 2.4.22 kernel for which exploit code is available, and would prevent the fifth one simply by compiling an additional kernel library.*

Presentation by **John Criswell**

Bryan Ford, MIT
Q: To what extent does the analysis equate to type safety?
A: Do not guarantee that pointers point to the correct object, but that it is a valid object.

Nickolai Zeldovich, Stanford University
Q: Must the number of memory pools be statically allocated?
A: No.
Q: What if memory pools are allocated at run-time?
A: Even then, its use is statically identifiable.

Jay Lepreau, Utah
Q: Is the overhead really small enough to make this approach suitable for use today?
A: Improving the overheads is future work.