HotOS XI: 11th Workshop on Hot Topics in Operating Systems

San Diego, CA
May 7–9, 2007

KEYNOTE ADDRESS

■ ***Transactional Memory: What's the OS Got to Do with It?***

David A. Wood, University of Wisconsin—Madison

Summarized by Ian Sin Kwok Wong (iansin@eecg.toronto.edu)

Multicore processors are here but we do not have the parallelism we need in applications to take advantage of this new architecture. Like Dave, most of us agree that parallel programming is hard because people think sequentially and as a community we have been using the same programming models for the past 30 years and thus have not acquired enough experience with parallel programs. An application can be parallelized through the use of threads. However, accesses to shared data must be carefully synchronized to maintain application correctness. Otherwise, deadlocks, live locks, or data races might result.

Transactional memory (TM) is one way to ease this burden while allowing concurrent execution of a program. TM originates from database systems and its declarative model makes it an attractive proposition. The programmer says what he or she wants and the system deals with the "how," while maintaining ACID properties. Software TM is slow and Dave believes that the next logical step is a hybrid implementation, which is basically a best-effort software TM with hardware acceleration for the common case. However, besides performance, the goals that TM systems are trying to achieve include unlimited transactions, long-running transactions, and unlimited closed nesting. In order to be successful, these facilities should be provided with modest hardware support.

Dave then gave the audience an overview of TM terminology and introduced LogTM-SE, which can be dubbed an "almost" virtualizable TM system. LogTM-SE is a hybrid TM that explores eager version management and eager conflict detection in the design space. LogTM-SE uses simple hardware support and exposes the interface to the software, which in turn implements the required policies.

In the Q&A period, Emin Gün Sirer (Cornell) argued that he did not believe TM to be the solution. In his opinion, locking instructions are a simple sequence of 12 instructions but the main problems are what students are being taught and what and how systems are being built (in reference to Linux). His question was, "Now that we have TM, what does this do for the average programmer?" Dave's response was that programming language experts would be required and that this subject was not his domain of expertise. Another interesting question from Kai Shen (Rochester) dealt with the state of the art in "transactionalizing" large systems. Dave replied that the main problem behind such an effort is that the simple close-nested abstraction was not powerful enough for highly complex systems and dirty tricks were required because of the lack of open-nested transaction support. He continued by arguing that the open challenge in transactionalizing large complex applications is to learn when simple abstractions are not sufficient and come up with extensions that will be usable by the average programmer.

COPING WITH CONCURRENCY

Session Chair: Armando Fox, University of California, Berkeley

Summarized by Ian Sin Kwok Wong (iansin@eecg.toronto.edu)

■ ***Is the Optimism in Optimistic Concurrency Warranted?***

Donald E. Porter, Owen S. Hofmann, and Emmett Witchel, The University of Texas at Austin

Donald Porter argued that the conservative mutual exclusion provided by locks, especially when locking is coarse-grained, is detrimental to performance. However, using a fine-grained locking scheme for better concurrency is very complex. Optimistic concurrency, achievable through transactional memory, removes the serialization points that locks suffer from. Donald argued that porting systems to leverage optimistic concurrency is a lot of work for potentially marginal benefits and his talk focused on quantifying the benefits of optimistic concurrency on multicore platforms.

He explained how his tool, called Syncchar, measures data independence in applications by analyzing the conflicts in the address sets that are accessed within critical sections. The tool was then used in a case study to measure data independence in a standard Linux 2.6 kernel. Although they found that most locks used in Linux were fine-grained, they also found 95% data independence on the dcache lock—which indicates a good opportunity for improvement. The study also compared Linux against TxLinux, an implementation that converts 32% of spinlocks to use transactions. Donald outlined some limitations, such as the pathological behavior of linked lists, which caused many conflicts and was due to the way the data structure was organized. The talk concluded with the question of whether optimistic concurrency will help your average system and the answer was, "It depends."

In the Q&A session, Dave Wood (Wisconsin) asked whether the linked-list pathological case was similar to open-nested transactions and whether it would be beneficial to raise the level of abstraction throughout Linux. Donald indicated that he looked into this and found the best way was to get rid of the data structure altogether to avoid the conflict. Although this might not be the best solution, it would be a first step before building nesting and correctness conditions. Ding Yuan (Illinois) followed with a question regarding how Syncchar dealt with multiple locks being held for a critical section. The answer was that each lock is treated individually for now, but ideally Syncchar should coalesce related locks. Sandhya Dwarkadas (Rochester) then asked whether Linux was the best place to use transactional memory and whether this would make the system simpler. Donald argued that transactional memory gives more options for tuning the system for performance and that is part of the reason for using transactional memory on the Linux kernel.

■ ***Thread Scheduling for Multi-Core Platforms***

Mohan Rajagopalan, Brian T. Lewis, and Todd A. Anderson, Programming Systems Lab, Intel

With significant architectural differences in many-core processors compared to traditional SMT architectures, Mohan argued that we need to rethink the way threads are scheduled for better performance. Most important, we need to know what threads to run, and on which cores to run them such that they benefit from cache locality. The goal of this work is to develop an automatic solution that is both portable and easily programmable, although it might not perform as well as a hand-tuned application.

The contribution of this work is an automatic scheduling framework that makes it relatively easy for the average programmer to achieve good performance without an in-depth knowledge of the underlying multiprocessor architecture. This is achieved through minimal programmer annotations, whereby related threads are tagged with a Related Thread ID (RTID). The system uses the RTIDs and their attributes for a best-effort placement of threads and updates them at runtime. Thread placement can also be explicitly guided by the expert programmer.

Timothy Roscoe (ETH Zürich) asked Mohan where the operating system (OS) was in this work. To Timothy, this is a runtime system. Mohan responded by saying the design was a runtime one but the scheduling problem is the same as traditional OS scheduling. A hanging question from Michael Isard (Microsoft Research) was whether threads were the right level of abstraction.

■ ***Automatic Mutual Exclusion***

Michael Isard and Andrew Birrell, Microsoft Research, Silicon Valley

Andrew Birrell argued that in software development we want correctness, efficiency, and maintainability. Increasingly, people want to program for parallel architectures and threads, together with locks as synchronizing primitives, as a popular way to implement parallel programs. He argued that the use of locks gets harder as a project grows in size and relies on the programmer to make decisions to maintain application correctness (e.g., to preserve the acyclic locking order). Failure to program locks correctly results in deadlocks, data races, and oversynchronization, among other problems. An alternative is the transactional memory paradigm. The complexity is moved from the programmer to the experts, but the programmer still needs to reason about the concurrent parts of a program.

They propose to solve these problems by using automatic mutual exclusion (AME). AME is similar to an event-based system augmented with transactions that can run concurrently. Transactions are presented differently to the programmer. Essentially the whole program is run as a transaction and the programmer must explicitly indicate when a code fragment is expected to run outside of a transaction. AME is composed of a thread pool and asynchronous methods. Each async method runs as a transaction and transactions execute concurrently with the help of threads. What is produced is a correctly synchronized concurrent program that the runtime system needs to execute efficiently. The responsibility of making intelligent decisions is thus moved from the programmer to the intelligent runtime system. To deal with blocking IO calls, AME implements a yield system method that breaks a method into atomic fragments. Any method may call one or more yields but the caller must be aware of this, since their state becomes visible.

After the talk, Jon Howell (Microsoft Research) agreed that labeling things to be outside transactions makes sense but wondered about yield propagation. From previous experience, he argued that most functions end up having the yield annotation, making them practically useless. Andrew replied by saying that they don't know yet and are in the process of implementing the system.

MODERN ABSTRACTIONS

Session Chair: Landon Cox, Duke University

Summarized by Ramakrishna Gummadi (ramki@cata-rina.usc.edu)

■ ***Hype and Virtue***

Timothy Roscoe, ETH Zürich; Kevin Elphinstone and Gernot Heiser, National ICT Australia

Timothy Roscoe presented a call to think about virtual machines differently for research purposes than what is done today. He argued that research on virtual machines has not provided new insights into operating system abstractions or structures; instead, it has focused more on building better hypervisors or on developing new uses or applications around them. However, the work of building hypervisors

involves reimplementing many of the abstractions of traditional OSes, such as protection and sharing of hardware resources, and efficient communication. So, the VMM research in this space has not been productive from a research perspective. Moreover, VMMs suffer from problems such as implementation complexity, poor performance, and large TCBs (Trusted Computing Blocks).

However, Roscoe noted that a key area in which VMMs represent a clear advance is in providing an application-level abstraction, since one can bundle entire applications as ready-to-run packages. So, OS designers can leverage this benefit of VMMs to save themselves the burden of porting applications while building new OS abstractions and facilities. The speaker outlined some really new research directions for disruptive virtualization, such as new kernel and OS API designs based on transactional memory, concurrent hardware, and high-level languages, as well as new kernel implementation techniques based on verifiable languages and machine-checkable formal specifications. He also called for OS and VMM designers to carefully examine the performance of resulting systems using improved and more meaningful metrics for measuring VMM isolation and scalability. The ultimate outcome of such efforts would thus be new and disruptive OS research that actually has the chance to succeed in the real world because of the availability of crucial application support provided by VMMs.

Gün Sirer from Cornell asked what to do with applications where one wants to keep some of the old POSIX interface yet include a new API. Roscoe answered that we might need to decide between having a legacy or a new interface. Margo Seltzer from Harvard remarked that the Program Committees at conferences should become more open-minded about accepting OSes that didn't necessarily support the usual suite of compatible applications and/or present performance numbers for such applications. Jeff Mogul from HP asked whether there is an analogy between the narrow waist occupied by IP in the Internet stack and VMMs in OSes. Roscoe replied that we shouldn't standardize a thin waist and that not having a thin waist in OSes is not a problem.

■ ***Relaxed Determinism: Making Redundant Execution on Multiprocessors Practical***

Jesse Pool, Ian Sin Kwok Wong, and David Lie, University of Toronto

Jesse Pool suggested a system that provides relaxed determinism guarantees in order to practically allow redundant execution of threaded processes on multiprocessors. The motivation is that future multiprocessors will have enough resources to allow processes to be executed redundantly in order to provide guarantees such as reliability, as well as security through diversity. Unfortunately, today's systems don't allow practical multithreaded applications to run redundantly because of various nondeterministic execution scenarios encountered under real-world settings.

Jesse described their system called Replicant, which provides reasonable performance while tolerating nondeterminism. Their key insight is that, in many cases, the event ordering seen by applications will not result in significant differences in application behavior, so some event orderings can be profitably relaxed. This approach is similar to the relaxed memory consistency models used in modern processors to achieve respectable performance. Thus, Replicant loosely replicates the order of events among the executing replicas, and it relies on determinism hints inserted by the developer in order to enforce a precise ordering of event delivery across replicas.

Replicant incorporates several system facilities to achieve such a nondeterministic but correct execution. First, each replica is executed in an OS sandbox called a harness, which captures the process-specific OS state. Second, a matcher component in the kernel is used to fetch and replicate inputs from the external world and deliver them to the harness. It is also responsible for determining when outputs from the harness should be made externally visible. The Replicant system has been implemented for Linux, and it has successfully managed to run several applications in the SPLASH-2 benchmark suite.

Jason Flinn from Michigan asked whether one can also use annotations as performance hints. For example, the replicas could all use shared memory, and the locks in the code can be thought of as annotations that indicate this possibility. Jesse responded that this was likely to slow down performance. Diwaker Gupta from UCSD asked whether the replicas can be thought of as state machines, and the speaker said that the replicas have only to provide identical outputs, so their implementation need not conform to the state-machine execution model. Diwaker also wondered whether Replicant would scale to more than two replicas and Jesse responded that they had tried more than two replicants. Emmett Witchel from Texas asked how the matcher could accurately deliver answers to inherently nondeterministic system calls such as gettimeofday. Jesse replied that the same answers to all replicas have to be returned for such system calls only when such calls were used in sequential regions of threaded code.

■ ***Compatibility Is Not Transparency: VMM Detection Myths and Realities***

Tal Garfinkel, Stanford University; Keith Adams, VMware; Andrew Warfield, University of British Columbia/XenSource; Jason Franklin, Carnegie Mellon University

Tal Garfinkel pointed out that recent worries about being vulnerable to stealthier rootkits are unfounded, because building transparent VMMs is effectively impractical. This is because there are numerous easily detectable anomalies between real and virtual hardware, as described in the

paper, and possible countermeasures to such anomalies are demonstrated to be infeasible. The take-away conclusion of the talk and the paper was therefore that transparent VMMs are unrealizable from both a performance and an engineering standpoint.

Tal talked about various discrepancies that an application running inside a VM can detect. For example, some CPU instructions are nonvirtualizable, so applications can execute them and observe their side effects in order to detect whether they are running in a virtualized state. Second, the emulated hardware is out of date with the CPU capabilities, because a VM typically emulates old but well-understood hardware resources such as chipsets and disks from the 1990s. Also, applications can easily detect features of emulated hardware resources such as TLB sizes that behave differently on virtualized hardware than on native hardware. Finally, there are a lot of timing discrepancies exposed through both local and remote time sources that allow an application to construct covert channels for detecting that it is running on a VMM. Although it is theoretically possible for a VMM to provide perfect transparency to applications through techniques such as time dilation, the resulting emulation overhead and the overall performance impact would make implementations impractical, while potentially opening up further, more subtle sources of vulnerabilities. Tal concluded that virtual machines can never truly approximate real hardware and that there are both good and bad outcomes as a result.

Timothy Roscoe from ETH Zürich pondered whether another form of nontransparency in VMs is when nonvirtualized programs running on virtualized hardware end up automatically exploiting the nontransparent behavior of VMs to do bad things, such as polymorphic viruses using nondeterministic cycle counts. Tal said that was an interesting idea, pretending to be a VM to fool malware into believing it is being run in a sandbox, at which point it exits. Tal mentioned some work at Symantec along these lines. John Wilkes from HP wondered whether there exists hardware to detect rootkits, and whether such hardware would be practical. Tal suggested that you write something that permits only authorized VMs to run. Emin Gün Sirer declared that he didn't believe in digital signature schemes, referring to authorizing VMs.

ALGORITHMS FOR PROFIT

Session Chair: Emmett Witchel, University of Texas at Austin

Summarized by Vinod Ganapathy (vg@cs.wisc.edu)

■ *Don't Settle for Less Than the Best: Use Optimization to Make Decisions*

Kimberly Keeton, Terence Kelly, Arif Merchant, Cipriano Santos, Janet Wiener, and Xiaoyun Zhu, Hewlett-Packard Laboratories; Dirk Beyer, M-Factor

Kim Keeton said that complex systems problems often present a large search space, with complex tradeoffs, with the best and worst solutions differing by as much as an order of magnitude. Currently, ad hoc domain-specific solutions are used to arrive at solutions to these optimization problems.

Keeton then went on to argue that the approach to this problem should be to use mathematical programming to solve these optimization problems. A math program has input parameters, objective functions, and constraints. The first step should be to formally describe the problem, following which commercial solvers can be employed to arrive at a solution. An alternative is to use meta heuristics, such as genetic algorithms, to arrive at solutions to optimization problems. Keeton described the generic structure of a genetic algorithm and showed how it can be used to, for example, minimize total penalty in an optimization problem.

Keeton also exploded several myths regarding optimization. (1) Myth: "Simple heuristics are good enough." Reality: Simple heuristics may be good enough, but what does "good enough" mean? (2) Myth: "One may have to oversimplify the problem to make it amenable to a math program solver." Reality: One can use alternative optimization techniques; one does not have to shoehorn the problem into the solver available. (3) Myth: "Optimization is too slow." Reality: This claim cannot be made in general; the cost of optimization depends on the specific problem to be solved. (4) Myth: "Inaccurate data can lead to bad decisions." Reality: This cannot be helped, and so a sensitivity study is necessary.

Someone asked whether there are any guidelines on when math programming is useful, or when machine learning is useful. Kim replied that you should use statistical machine learning to uncover what variables are important. Brian Knoll of Michigan wondered whether the main challenge was in trying to express all costs in the same currency. Will this not result in bad data, which results in inaccurate results? Kim answered that we must express everything in the same currency. In their paper they did so using dollars (real currency) to quantify the costs of various options.

■ ***Hyperspaces for Object Clustering and Approximate Matching in Peer-to-Peer Overlays***

Bernard Wong, Ymir Vigfússon, and Emin Gün Sirer, Cornell University

Bernard Wong explained that the motivation for this work is that services such as Gnutella provide a search primitive that can conduct approximate search (so a search for "Bratney Spears" will still yield files related to the real goal of the search "Britney Spears"). However, Gnutella is slow. One solution is provided by systems such as Chord, Pastry, etc., all of which use distributed hash tables (DHTs). However, DHTs do not support approximate search, but need the exact key.

Wong then went on to present the hyperspace model, which achieves the best of both worlds by supporting approximate search on P2P overlays. A hyperspace is a high-dimensional space in which objects that are "close by" in the sense of having small edit distance are located close together in the hyperspace (i.e., the Euclidean distance between these objects is small). The main challenge in a hyperspace is to correctly choose the basis for the high dimensional space, and the cost of a poor selection of labels is that it can lead to poor clustering, and thus poor search results.

The questions mainly concerned how difficult it was to choose an appropriate basis. Wong mentioned that basis selection must be repeated over time, depending on the current search queries that were popular.

■ ***Optimizing Power Consumption in Large Scale Storage Systems***

Lakshmi Ganesh, Hakim Weatherspoon, Mahesh Balakrishnan, and Ken Birman, Cornell University

Lakshmi Ganesh began by saying that much money and energy get wasted at data centers (e.g., $7.2 billion in a recent year, of which $2.4 billion was spent to cool disks). Thus, a technique is needed to reduce this amount (e.g., by spinning down disks from which data is not immediately needed).

Ganesh then went on to present a file system–level solution to the problem by using a log structured file system for this purpose. Using a log structured file system ensures that new data is appended only to the end of the log. Thus, all the disks that are not currently being written to can be spun down provided that the data that is accessed most often from these disks get cached. In addition, log cleaning can be used to concentrate popular data on the same disk.

Q&A: In terms of how many reads hit the cache, someone mentioned that server disks are more sensitive to power than laptop disks and wanted to know if using laptop disks would mitigate this problem. Another audience member suggested that this problem might be mitigated by buffering before actually performing writes to the disk.

One questioner asked whether the power reduction numbers presented in the talk and in the paper take into account the full machine or just the disk subsystem. The answer was that they only take into account the disk subsystem.

GUARANTEES FOR THE FUTURE

Session Chair: Amin Vahdat, University of California, San Diego

Summarized by Ramakrishna Gummadi (ramki@catarina.usc.edu)

■ ***Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era***

Ben Greenstein, Intel Research Seattle; Ramakrishna Gummadi, University of Southern California; Jeffrey Pang, Carnegie Mellon University; Mike Y. Chen, Intel Research Seattle; Tadayoshi Kohno, University of Washington; Srinivasan Seshan, Carnegie Mellon University; David Wetherall, University of Washington and Intel Research Seattle

Ben Greenstein argued that today's wireless devices severely compromise a user's privacy because of various limitations in the design and implementation of wireless protocols. Such privacy threats should be seen as imposing an economic cost on the threatened users, because the exposed information includes data about browsing history, location history of previously visited access points, information about applications running on a user's computer, and details about capabilities and features of the hardware on the user's computer.

Using measurements from a publicly available SIGCOMM '04 trace of 802.11 network usage, Ben pointed out that exploiting these vulnerabilities allows an adversary to identify and track the locations of more than 25% of the user population with an accuracy of 99% or better. He then outlined the systems challenges involved in building privacy-preserving wireless protocols, such as a privacy-enhanced 802.11 MAC. Such research challenges fall into three main categories: building a naming architecture with anonymity properties better than those afforded by pseudonyms; building resource discovery and binding protocols that let a user search for, select, bind, and then migrate to resources such as access points; and limiting information leakage by preventing the inadvertent exposure of implicit identifiers that allow an attacker to identify and classify the hardware and software being used by a user with high accuracy.

Jason Flinn from Michigan asked whether the privacy and performance metrics in the paper represent commonly accepted metrics, and the speaker replied that he hoped so. Steve Hand from Cambridge asked what the most identifying feature in 802.11 was, and Ben replied that it was the set of IP destinations accessed by a user. Margo Seltzer from Harvard asked whether there is a difference between

her generation and the current one in terms of privacy expectations: Younger people today seem to be more willing to openly reveal their locations and activities on sites such as MySpace. The audience thought that adverse impact in terms of future study and employment opportunities owing to lax attention to privacy could soon cause people to take their privacy more seriously. The final question dealt with the implications of changing the MAC address frequently to provide anonymity, because many protocols use MAC addresses for various functions such as authentication. Ben replied that one could change the MAC address slowly, such as each time you associate to an AP, so that the changed address could be used for authentication, in conjunction with a more permanent address.

■ ***Auditing to Keep Online Storage Services Honest***

Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, and Ram Swaminathan, HP Labs

Mehul Shah talked about the importance of providing a third-party auditing facility for online services such as storage and the challenges involved in provisioning reliable internal and external audit facilities for such services.

Mehul described a catastrophic data-loss scenario for a user using an online storage service provider. The user could not make a well-guided selection of the storage service she used because of lack of reliable information about the relative service qualities offered by competing storage providers. The speaker pointed out that, in the real world, there is already considerable appreciation of the functionality provided by auditors and insurance agents who contract them. He said that there are two main approaches to auditing: external and internal. In external auditing, externally visible interfaces are used to measure and predict the properties of a service. In internal auditing, information about the extent to which a service follows best practices and processes internally to ultimately meet its service objectives is assessed. Both types of audits are needed, and they complement each other. Mehul pointed out several goals and properties of audits and said that auditing was motivated by demands placed by insurance providers or by government regulations.

Mehul then explained the interfaces and hooks that are necessary for storage services to maintain the service's SLAs. A main challenge is to preserve privacy while ensuring data longevity and integrity. The auditing process must also be bandwidth-efficient. Mehul then proposed a privacy-preserving approach for auditors to verify that the data stored by the service providers is correct. It uses hashes on encrypted data and ensures that the key used to encrypt the data need not leave the service provider, while simultaneously guaranteeing that the service provider has not lost any of the user's data. He concluded by saying that there is therefore both a need and a mechanism for efficiently and privately verifying the performance of online storage providers.

David Lie from Toronto asked how one can audit rapidly changing data. Mehul proposed the use of batching. Landon Cox from Duke asked whether an alternative approach would be to keep a separate copy of the data with the customers themselves, or, alternatively, to keep a copy with the auditors. Mehul replied that one of the assumptions of the work is that the customers are not expected to keep any copy of the data they originated with themselves, and that auditors should not necessarily be trusted with the original data. Hakim Weatherspoon from Cornell asked whether it is possible for the auditor to be anonymous. Mehul replied that the solution is then to use data sampling. Gün Sirer from Cornell asked whether the reputation provided by auditing is really meaningful, and Mehul replied that auditing indeed helps develop reputation. Finally, the audience wondered whether moving toward data assessment through auditing or more simply toward better data backup is ultimately the right thing to do.

■ ***A Web Based Covert File System***

Arati Baliga, Joe Kilian, and Liviu Iftode, Rutgers University

Liviu Iftode presented a Web-based covert file system called CovertFS. Liviu first outlined the requirements for a steganographic file system centered on Web services for media sharing and storage: providing plausible deniability, allowing online access and sharing, and providing information hiding for confidential documents and information. He then presented the main design concepts of CovertFS and the challenges involved in providing a file system abstraction build on top of a Web-based system for sharing photos.

In CovertFS, both data and metadata such as inodes are stored in photos. The root of the file system is then accessed through a hash of the encryption passphrase entered by the user. CovertFS includes techniques to hide access patterns that may reveal its hidden purpose behind photo accesses. They include ways to manage frequently changing image data owing to file system writes by using immutable allocation maps and avoiding photo access hotspots resulting from metadata accesses, which could tip off an external party, by using image chains. CovertFS also includes facilities for access control, such as allowing read-only access on publicly shared photos, and replication to manage Web site unavailability. Finally, plausible deniability can be provided by having multiple levels of CovertFS, the top few of which are potentially less incriminating. Finally, Liviu talked about how to manage both active adversaries who can perform steganalysis and passive adversaries who can mount traffic analysis. He concluded by saying that they are currently building a working prototype that they hope to evaluate in terms of latency, scalability, security, and privacy.

Mary Baker from HP wondered whether write accesses can be more covertly managed by pretending that images are being manipulated for common operations such as red-eye

removal. The speaker agreed. Margo Seltzer from Harvard asked whether one can build a cooperative steganographic service and whether the threat model would be different. Gün Sirer from Cornell wondered whether it is possible to use file systems optimized for write-once, read-many workload, such as an ISOFS. Finally, Jason Flinn from Michigan wondered why a file system should be used for covert sharing in the first place. Liviu answered by saying that a file system is an abstraction familiar to users.

PANEL: PUTTING THE SCIENCE IN COMPUTER SCIENCE

Session Chair: Margo Seltzer, Harvard University

Summarized by Vinod Ganapathy (vg@cs.wisc.edu)

Panel Members: Dawson Engler, Stanford University; Butler Lampson, Microsoft; Jay Lepreau, University of Utah, virtualized by Jeff Mogul, HP Labs; Brian Noble, University of Michigan, virtualized by Yuanyuan Zhou, UIUC

Brian Noble started by saying that the good news was that the OS community had begun to think about usability and not just performance. However, the bad news is that we are really poor at evaluating our work. Most work proceeds by conducting a toy user study or presenting anecdotal evidence with excuses such as "We don't need a user study," "Our colleagues thought that our system was neat," and "User studies are too hard!" Noble went on to conclude that the main reason we don't conduct user studies was because of fear of the unknown.

Noble mentioned the need to collaborate and work with experts in the HCI area to conduct meaningful user studies. He encouraged the community to learn the rules, as IRBs are typically not the enemy. He did, however, mention the long cycle times needed to conduct a user study and also the need to overprovision the resources needed [because a subject, once used in a study, cannot be used in that same (redesigned) study once again]. Therefore there is a need to conduct several little pilot studies before the final study.

The second speaker, YY Zhou, spoke about the need to put "Nerdiness" into "Hackers." She had began in Princeton as a theory student, and therefore as what she believed was a "nerd." Her advisor then convinced her to become a "Hacker," and so she is now what she calls a "Neker."

Zhou then described a typical research cycle, in which we select a problem, either based on demands, trends, and challenges in the real world or as a response to other papers from the community, then abstract it, solve it in the abstract, and develop a proof-of-concept solution. The next step is to conduct a user study to evaluate these proofs of concept. She then mentioned that it is time for hackers to admit that nerds can be cool too. For example, in her own work she uses machine learning and data-mining techniques, and she said that the community must consider publishing in venues such as SysML and control theory conferences.

The third speaker, Jeff Mogul, presenting Lapreau's slides, mentioned the need to not just have "reproducible research" but "replayable results." The goal is to have the ability to replay the entire system, i.e., software plus hardware, so that we can fiddle around with parameters and see how the system responds. This is possible using a virtual machine infrastructure, data repositories, experimental management systems, and grids.

The fourth speaker, Butler Lampson, proposed that to bring science to computer systems research, it was very important to write precise specifications for the systems that we build. The techniques and tools needed to write specs have all been developed in research over 10–15 years ago. The basic idea is to build a system and write a simulation proof that the system built indeed conforms to the spec.

Lampson said that by writing a spec, it is often possible to learn things about the system that the designers couldn't have learned otherwise. He mentioned the example of a student whose thesis committee he was on. This student was designing a CVS-like system on P2P over a DHT but had a flaw in his design that became obvious when he encouraged the student to write a spec of his system. Lampson then switched gears to say that scientists, contrary to popular belief, do *not* replicate experiments.

The audience then joined the discussion. One member asked whether a bad user study is better than no user study at all, to which Noble replied that what we currently have are only bad user studies, which is why we need to collaborate with HCI folks. Another audience member mentioned that the time and effort needed to conduct a detailed study such as the one that the panelists were referring to takes a lot of time, which means that the students will graduate without enough papers—students nowadays need at least three or four decent papers in top venues to get good jobs, so isn't there a conflict here? One panelist strengthened this view by saying that even universities nowadays don't hire such people, so we don't breed that kind of culture in our universities now. Another audience member questioned the need for writing specs by asking why specs are important. Isn't impact on the real world more important?

NEW SOLUTIONS, OLD PROBLEMS

Session Chair: Yuanyuan Zhou, University of Illinois at Urbana-Champaign

Summarized by Diwaker Gupta (dgupta@cs.ucsd.edu)

■ ***Purely Functional System Configuration Management***

Eelco Dolstra, Utrecht University; Armijn Hemel, Loohuis Consulting

Keeping software up to date poses a huge system administration challenge, in large part owing to problems with existing configuration management tools: Dependency handling is not perfect, having multiple versions of software on a system is often problematic, configuration files often undergo destructive modifications on updates, etc. In this talk Eelco Dolstra presented a new Linux distribution—NixOS (http://nix.cs.uu.nl/nixos)—built around a purely functional system configuration manager called Nix.

The key insight is that most of these problems arise because of the imperative model used by most configuration tools—users need to describe *how* to get things done as opposed to *what* needs to be done. Drawing motivation from the programming language world, NixOS uses functional programming paradigms such as *referential transparency* to manage system configuration. In particular, Nix is completely stateless, and the entire system configuration is rebuildable from a single, declarative specification. Once built, packages are immutable. Multiple versions of packages are supported and dependencies are propagated in the build system.

A consequence of this design is that if one of the core libraries (say, libc6) undergoes modification, pretty much the entire system has to be rebuilt, resulting in significant storage overhead. Another practical problem arises because NixOS generates configuration files in a declarative manner as well, so there is no easy way to manually modify a configuration file, primarily because tracing the place in the build system where it was generated is nontrivial. Nix does provide repositories with prebuilt packages to save build time for end users.

■ ***Processor Hardware Counter Statistics as a First-Class System Resource***

Xiao Zhang, Sandhya Dwarkadas, Girts Folkmanis, and Kai Shen, University of Rochester

Processor hardware performance counters started out as verification and debugging aids, but they have evolved into a rich source of statistical information invaluable for several applications such as CPU scheduling, self-managing applications, and benchmarking tools. These performance counters are usually managed in hardware and read using some low-level interface by the operating system. In this talk, Sandhya Dwarkadas made a case for hardware counters to be managed as first-class entities by the OS.

Management means providing a high-level API for applications to use, and also virtualizing the counters on a per-process basis. This would allow things such as measuring the number of cache misses for a particular application. There are several applications that would benefit from such facilities: Resource-aware OS schedulers could use hardware counters as input to a counter-based resource model; hardware counters could also be used to distinguish between CPU-intensive and memory-intensive requests to do online workload modeling.

However, there are several issues to be hashed out. Security is a concern: Can counters act as covert channels to leak information? Another concern is performance: Currently, counters are attractive because they are extremely efficient. However, OS management might make counters significantly slower, nullifying the benefits of management. The main push of the paper was to encourage a dialog between hardware vendors and OS developers.

■ ***Microdrivers: A New Architecture for Device Drivers***

Vinod Ganapathy, Arini Balakrishnan, Michael M. Swift, and Somesh Jha, University of Wisconsin—Madison

It is well known that the bulk of the bugs in OS code come from device drivers, primarily because device drivers are hard to get right, extremely hard to debug, and often written by those who are not kernel experts. Vinod pointed out, however, that the fundamental problem was the architecture of monolithic kernels (e.g., Linux). Since the device driver runs in the kernel's address space, a faulty driver can easily take the whole system down. Earlier projects have tried to address this issue by moving device drivers to user space, but these approaches suffer from either poor performance or incompatibility with commodity OSes.

Microdrivers is a new approach to building device drivers that is both efficient and backwards-compatible. The key idea is to split each driver into a kernel driver and a user-space driver interacting over a driver runtime. Performance-critical functionality sits in the kernel driver; the rest is delegated to user space. However, the real clincher for this approach is that existing device drivers can be semi-automatically converted to microdrivers.

The code generation takes place in two stages. First, a "splitter" detects function-level split in driver code. Second, the "code generator" takes as input marshaling annotations required for serializing complex data structures and outputs code for the different components.

This talk sparked a lot of discussion and questions. One of the biggest complaints was that the evaluation in this paper doesn't actually implement the microdriver approach (since everything executes in the kernel). Another observation was that many bugs usually occur in corner cases and a split based on functionality might not be able to capture it. Clean maintenance of split drivers as the upstream driver code evolves also poses a challenge.

4-MINUTE MADNESS

Session Chair: Rebecca Isaacs, Microsoft Research Cambridge

No summary available, but see the Sirer-Farrow article in this issue, based on a talk from this session.

WEB 2.0

Session Chair: David Wetherall, Intel Research and University of Washington

Summarized by Vinod Ganapathy (vg@cs.wisc.edu)

■ ***MashupOS: Operating System Abstractions for Client Mashups***

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

Howell presented background on Web 2.0 technologies such as AJAX and also browser policies, such as the Same Origin Policy. He then presented Mashups, for example, where housingmaps.com uses information from both Google Maps and Craigslist to present a listing of available houses.

Howell mentioned that the problem was the binary security model of the Web, where content from domains can be isolated perfectly using IFRAMES, or has no security at all (e.g., a SCRIPT executes in the context of the page that includes it). Howell thus argued for the need for sophisticated interaction among various components of a page. The solution is a new abstraction called a ServiceInstance, which is akin to a process on an OS. Each service instance is associated with a single domain, and each resource has its own service instance. Service instances are created by using the FRIV tag (a new construct introduced in HTML). Service instances allow for limited communication, and they are thus a hybrid of FRAMES and DIVS. Howell also went on to mention that MashUpOS can be implemented with script rewriting (e.g., using the Browsershield framework).

The questioners asked whether service instances were akin to adding a new element to the process hierarchy, and if so, why this was indeed a right abstraction. Another questioner also asked how to label pages from different domains: Should the browser do it, or should applications do so cryptographically? Howell said this was an issue they were currently examining.

■ ***Live Monitoring: Using Adaptive Instrumentation and Analysis to Debug and Maintain Web Applications***

Emre Kıcıman and Helen J. Wang, Microsoft Research

Emre Kıcıman said the motivation for this work was that huge amounts of code were downloaded on the client side in today's Web 2.0 sites, with sites such as Google Maps downloading up to 50,000 lines of code to the client side to improve user experience. Thus lots of code executes on the browser, and there are third-party dependencies, too (e.g., using Mashups).

Kiciman explained the need for end-to-end visibility, which would help Web application developers better understand and tune their applications. He argued for an on-the-fly rewriting technique that would be deployed with the code that is downloaded on clients, which would help the application developer with issues such as performance and correctness debugging of these applications. The key was that this offers a different deployability model, where deployment is immediate, with very fine-grained control over who's using what instrumentation. Also, all this is possible in their system without any changes to the server or the client (and is done via Javascript rewrites).

In the Q&A session, one questioner expressed concern about the "willy-nilly" rewriting being performed by their system, and about all this code executing in the browser, which he mentioned did not make him feel very comfortable from a security standpoint. A second questioner asked whether different instrumentation at different sites would cause problems in understanding bugs.

■ ***End-to-End Web Application Security***

Úlfar Erlingsson, Benjamin Livshits, and Yinglian Xie, Microsoft Research

Úlfar Erlingsson said that today's data transmitted over the Web is rich data; that is, it can contain embedded scripts, which can be used to launch cross-site scripting attacks. The standard solution adopted nowadays is serverside sanitization (e.g, by disallowing scripts). However, this is a hard problem to solve, because the server must now parse the scripts in exactly the same way that browsers do. Bugs in doing so can result in security holes and worms (e.g., the Yamanner Yahoo! mail worm, the Samy Myspace worm).

Erlingsson went on to argue that security applies to both servers and clients, and he described a mechanism called METS that achieves this. METS allows expression of security policies. The idea is that these will be specified by the server but will be enforced at the client (within the browser). Thus, METS ensures high-fidelity enforcement of security policies. The basic idea behind METS is not new—they are much like inline reference monitors.

One questioner mentioned that currently the onus of enforcement was on the browser, so METS relies heavily on browser manufacturers to ensure security. Is this practical? A second questioner said that the problem was that today's Web languages such as HTML lack a real specification and that several current problems could go away if we have a precise specification for HTML. A third person questioned the practicality of having Web server designers writing METS policies. Web server designers in today's environ-

ment can be as clueless as an end user. How practical is it to assume that they will be able to write meaningful policies?

FINDING A BETTER WAY

Session Chair: George Candea, EPFL

Summarized by Diwaker Gupta (dgupta@cs.ucsd.edu)

- ***HotComments: How to Make Program Comments More Useful?***

  Lin Tan, Ding Yuan, and Yuanyuan Zhou, University of Illinois at Urbana-Champaign

All of us who have programmed understand the value and pitfalls of source code comments. In this talk, Lin Tan discussed the feasibility of analyzing comments and detecting inconsistencies between code and comments. This is a fairly tall order: Bear in mind that comments are imprecise, unstructured, and often written in prose and cannot be tested or verified. However, empirical evidence suggests that often programmer assumptions and expected usage (for instance, code that requires a lock to be held) are most succinctly captured in source code comments. Failure to convey these assumptions to users and other developers often leads to bugs.

The goal of this work is to leverage these assumptions and example usages to detect inconsistencies between code and comments (and thereby detect potential bugs). A code-comment mismatch indicates either a bug or a wrong comment, both of which can lead to bugs, so detecting mismatches is certainly useful. But how feasible is it to extract structure from comments? This work uses Natural Language Processing (NLP) along with clustering on topics such as "locks" to analyze comments: Comments are mapped to predefined templates.

The authors extracted 530 rules from 5 different Linux subsystems and detected 12 new bugs (2 of which have been confirmed by Linux developers). For lock-related and call-related topics, the system "just works" and almost no user intervention is required. For other kinds of comments, more templates need to be defined.

- ***Towards a Practical, Verified Kernel***

  Kevin Elphinstone and Gerwin Klein, National ICT Australia and the University of New South Wales; Philip Derrin, National ICT Australia; Timothy Roscoe, ETH Zürich; Gernot Heiser, National ICT Australia, the University of New South Wales, and Open Kernel Labs

As a foundation for building secure systems, researchers have advocated a small Trusted Computing Base (TCB) that can be manually audited. This paper takes a much stronger stand: Kevin presented their efforts at building a formally verified kernel about which properties can be proved and a guaranteed correct implementation can be provided. The idea is to start with an abstract model and transform it all the way down to a high-performance implementation in C and then assembly code.

One obvious implication of this approach is that any code changes will invalidate proofs. To avoid this, the kernel model is made as detailed as possible—this is done in Literate Haskell. The documentation is embedded in the code of this abstract model, so a single spec can generate the reference manual, as well as a kernel prototype in Haskell that can be run through a user-level simulator. The kernel is modeled as a big state machine with events as inputs—things such as manipulation of low-level state (page tables) as well as preemption have also been modeled.

The kernel prototype in Haskell is about 3,000 lines of code with an accompanying 53,000 lines of code of proof! There exist a proof of termination and a proof of correctness for all but one system call. The authors stated that the transformation from Haskell to an actual C version would be manual, since automatic code transformation would miss significant opportunities for optimization.

- ***Beyond Bug-Finding: Sound Program Analysis for Linux***

  Zachary Anderson, Eric Brewer, and Jeremy Condit, University of California, Berkeley; Robert Ennals and David Gay, Intel Research Berkeley; Matthew Harren, George C. Necula, and Feng Zhou, University of California, Berkeley

Jeremy began by noting the difference between bug finding and soundness analysis: Bug finding involves heuristics and approximations; soundness, in contrast, ensures the *complete absence* of a particular class of bugs. Soundness has been perceived to be extremely hard to attain in practice, and the goal of this work is to make soundness analysis practical for a large system such as Linux. There is some confusion in terminology, since soundness and completeness have slightly different meanings in the literature.

The authors use a combination of lightweight annotations and hybrid checking to make incremental progress toward reliable software. The new goal for system evaluation is to "minimize the amount of untrusted code" as opposed to "maximizing the number of bugs found." They began with a limited subset of the Linux kernel (around 400,000 lines of code) and subjected it to three different analyses:

- Deputy: memory and type safety checks
- Count: deallocation safety
- BlockStop: call graph analysis to identify interrupt handlers that may block

Overall the authors conclude that these analyses can be done efficiently and new tests can be added to make the tests more conclusive. All the code and findings are available at http://ivy.cs.berkeley.edu. Surprisingly, this approach is not effective at *finding* bugs.