conference reports

THANKS TO OUR SUMMARIZERS

2009 USENIX Annual Technical Conference

San Diego, CA
June 14–19, 2009

OPENING REMARKS

Summarized by Rik Farrow

After thanking the program committee and the USENIX staff, co-chairs Geoffrey M. Voelker and Alec Wolman announced the Best Paper awards: "Satori, Enlightened Page Sharing" by Grzegorz Miłoś, Derek G. Murray, Steven Hand, and Michael A. Fetterman, and "Tolerating File-System Mistakes with EnvyFS," by Lakshmi N. Bairavasundaram, Swaminathan Sundararaman, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Next, Alva Couch, Secretary of the USENIX Board of Directors, presented the Software Tools User Group Award to Jean-Loup Gailly and Mark Adler for their work on file compression (see http://www.usenix.org/about/stug.html for details). Gailly said, "I feel I have received more from the OS community than I gave," and Adler then said "Ditto," in a couple of the shortest acceptance speeches ever.

Couch then presented the Lifetime Achievement Award to the late Professor Gerald J. Popek (you can read more at http://www.usenix.org/about/flame.html). Two of his past students, Bruce Waller of HP Labs and Geoff Kuenning of Harvey Mudd College, described Popek's long history in CS research, with many contributions in systems, including the first mention of clusters. Both men also spoke of Popek's dedication to his students. Kuenning explained that Popek had taken a long leave from academia to start Locus Computing, which is why his name disappeared from publications sometime during the '90s.

KEYNOTE ADDRESS

- ***Where Does the Power Go in High-Scale Data Centers?***
  James Hamilton, VP & Distinguished Engineer, Amazon Web Services

  Summarized by Stephen P. Tarzia (starzia@northwestern.edu)

James Hamilton gave a fresh appraisal of electrical power's role as a primary design consideration in data centers. The fundamental issue is that high-scale data centers such as those managed by Amazon and Google are very different from conventional enterprise data centers. Due to management complexity introduced by heterogeneity, people costs dominate the total enterprise datacenter costs. By contrast, a high-scale data center typically has more than 1000 servers per administrator, so people costs are almost negligible. In this keynote, Hamilton outlined the true costs of such data centers as well as their engineering implications.

Hamilton gave a total cost analysis for operating a theoretical 15 megawatt high-scale data center. He showed that servers accounted for 53% of costs, power and cooling infrastructure for 23%, and power usage for 19%. Since server prices are falling, he forecast power-related costs accounting for over half of total costs in the future. However, it is important to note that the majority of power-related costs are due to infrastructure, not utility charges.

To drive cost-cutting efforts, Hamilton advocated measuring Total Power Usage Efficiency (tPUE), the ratio of total facility power to power delivered to server components. His blog, in particular the entry at http://perspectives.mvdirona.com/2009/06/15/PUEAndTotalPowerUsageEfficiencyTPUE.aspx, has more details on tPUE. This measure differs from the traditional metric, PUE, in that it includes energy waste within IT equipment. In particular, motherboard voltage regulation circuits and case fans are often unnecessarily inefficient. He showed all of the steps in the power distribution chain, which has over 90% end-to-end efficiency.

To get the maximum return from the data center's power and cooling infrastructure investment, the operator must run as many servers on top of that infrastructure as possible without overloading it during peak periods. To achieve that, Hamilton suggested a combination of both cooling and server-utilization optimizations.

Regarding cooling, Hamilton first promoted isolating hot and cool air flows and running data centers at much higher temperatures. Hamilton showed that popular server warranties typically cover equipment that is run at up to 95°F, much hotter than a typical data center. Based on this observation, Hamilton proposed using outdoor air instead of AC for cooling. Some worry that airborne particles from outdoors might damage IT equipment, so detailed studies are needed to test this and to evaluate filtration techniques.

Finally, Hamilton discussed resource consumption shaping. This means reducing peak load at the expense of increased trough load. In other words, smooth out the load curve by pushing some of the peak workload into idle times. He suggested following the airline industry's model of overbooking and then shedding excess load when necessary to maximize capacity utilization. Hamilton also suggested using the same load-smoothing approach with links. Further gains can be had by increasing average server utilization, a figure that is typically only around 15%.

Rik Farrow asked why datacenter operators don't have servers custom-built to work most efficiently in their facility. Hamilton responded that the big datacenter operators do work closely with custom design groups in computer manufacturing companies. He also mentioned that big-impeller fans and shared power supplies are typical requests. Can server traffic can be pushed by hours, since this is what would be needed to smooth out daily user cycles? There is lots of work to be done at night, in particular data analysis and data mining. Still, Hamilton acknowledged that operators will have to pay for peak-time responsiveness.

Is humidity an issue in the data center at higher temperatures? Everyone fears humidity, but concrete data is lacking. How do networking costs figure into the total and will ISPs change their pricing model if link utilization increases? WAN costs were not included in Hamilton's analysis, but they are minor: only a few percent. Hamilton was not prepared to comment on ISP pricing. David Petrow asked about the role of server water cooling now and in the future. Hamilton observed that the industry loves density, while floor space costs are negligible, so water cooling is unnecessary. When asked by Dan Klein how to shed light on the right people to promote his agenda, Hamilton suggested focusing on those with the biggest R&D budget.

The final two questions returned to server utilization. When asked for an example of software inefficiency, Hamilton noted that some software inefficiency must be tolerated, such as using high-level languages to increase developer productivity and thus drive innovation. Someone asked how tPUE included the actual work done. Hamilton acknowledged that it does not, but it is valuable since it is generalizable across different applications and industries. He recommended additional industry-specific calculation of work done per dollar.

VIRTUALIZATION

Summarized by John McCullough (jmccullo@cs.ucsd.edu)

- ***Satori: Enlightened Page Sharing***
  Grzegorz Miłoś, Derek G. Murray, and Steven Hand, University of Cambridge Computer Laboratory; Michael A. Fetterman, NVIDIA Corporation

  Awarded Best Paper!

Miłoś described a system to leverage page sharing without the overheads of VMM page scanning. Memory can be one of the most limited resources in virtual machines, and in the common situation of homogeneous virtual machines there can be a large amount of redundant data. Typical approaches involve the VMM scanning all pages, creating fingerprints, and then initiating page sharing. This is a heavyweight operation whose periodicity is limited. Miłoś observed that many shared pieces of data arise from I/O devices and that by instrumenting the virtual I/O devices we can capture that page sharing and avoid periodic scanning. An additional benefit of this approach is that when using copy-on-write disk images, VMs can bypass the disk read and share the data if it is resident in memory elsewhere. Satori implements I/O-based page sharing behavior in the Xen hypervisor.

While typical page sharing approaches release pages into a global pool, Satori credits fractions of the freed pages to the VMs participating in the sharing. These credits can be taken from a type of inverted-balloon driver, but to prepare for share-breaking the VM must maintain a list of volatile pages that can be evicted at any time. These pages can typically be used for additional page cache. The system performs well in general, with less than 1% overhead for random reads and meta-benchmarks. However, for sequential reads there is a 35% slowdown due to hypercall overheads. Miłoś believes this overhead can be alleviated through a shared memory approach. Satori outperforms VMware's fastest—yet still infrequent—similarity scans, even though Satori cannot perform sharing for pages not loaded through I/O, including the kernel, which is pre-loaded by the hypervisor.

An audience member asked whether the VMs can steal memory from other machines by reading extra shared data. Miłoś responded that the machines cannot gain any additional memory this way. Do the costs of sharing and detection outweigh the potentially short duration of the potential sharing? The aggregate of the short-lived sharing opportunities can still provide a great benefit. Does it make more sense to explicitly share the page cache? The goal is to provide the benefits of page sharing with minimal modification to the guest operating system. Transcendent memory implements the shared page-cache behavior.

- ***vNUMA: A Virtual Shared-Memory Multiprocessor***
  Matthew Chapman, The University of New South Wales and NICTA; Gernot Heiser, The University of New South Wales, NICTA, and Open Kernel Labs

Chapman observed that when you need more computational power than a single processor, you typically turn to a shared memory multiprocessor or a cluster of workstations. Large shared memory multiprocessing systems are often very expensive, and workstation clusters are often awkward to program. Typical approaches that join a workstation cluster into a single machine image use language-specific middleware or narrowly supported distributed operating systems. Chapman proposed vNUMA, where a virtual machine monitor presents an unmodified operating system with a single machine image spanning a workstation cluster.

vNUMA addresses a number of challenges in faithfully reproducing the SMP programming environment. Unlike many distributed shared memory systems, all data in an SMP system is shared, including locks, and read-modify-write and memory-fence behavior must be respected. Because no particular write invalidation technique is performant across all access patterns, vNUMA uses an adaptive protocol selecting among three approaches for particular memory pages. In one update case, trap-emulation is required, but a simple write-invalidate is used in most cases. Chapman showed that vNUMA can out-perform the distributed shared memory library, Treadmarks, across compute-intensive HPC benchmarks and that vNUMA performs comparably to distcc for compilation. However, for I/O intensive database workloads, vNUMA performs poorly. Overall, vNUMA provides a single system image for free with good computation performance.

An audience member asked how devices are handled. Chapman responded that the work focused primarily on memory behavior rather than devices. In the current implementation, network and disk I/O are routed through node zero. Future work could introduce striping across nodes and improve performance.

- ***ShadowNet: A Platform for Rapid and Safe Network Evolution***
  Xu Chen and Z. Morley Mao, University of Michigan; Jacobus Van der Merwe, AT&T Labs—Research

Chen observed that alternative configurations on carrier-grade networks can have negative effects. However, existing modeling and emulation testbeds cannot get the same fidelity and hardware implementation as the production network. Chen proposed ShadowNet, a system that provides a network that is connected to but separate from the production network. This provides an environment in between the lab and the production environment and allows multiple service trials to run simultaneously, sharing the same physical resources but in isolation.

ShadowNet is implemented on top of Juniper-based virtual routers, which are, in turn, attached to a ShadowNet node hosting virtual machines. Each virtual router provides the full functionality of the original routers, while connected to each other and VM instances via a variety of connectivity options, keeping traffic isolated and routing updates regulated. At the experimental level, ShadowNet provides configuration management across experimental configurations. Chen demonstrated that ShadowNet is able to get the desired bandwidth allocations, although the virtual routers have some interaction with the other routing elements under high load. Chen also demonstrated that ShadowNet can achieve failover to an alternate configuration.

An audience member observed that in PlanetLab-style deployments it is very easy to add nodes and asked how feasible it is to add new nodes in ShadowNet. Chen noted that the controller takes care of adding the nodes and that they should be easy to add. Do any cloud vendors provide similar systems? Most cloud infrastructures only provide the virtual hosts, and ShadowNet has richer networking support.

INVITED TALK

- ***Teaching Computer Science in the Cloud***
  David J. Malan, Harvard University

Summarized by Matthew Renzelmann (mjr@cs.wisc.edu)

Professor David Malan presented his work on reinvigorating Harvard's introductory computer science course, CS 50, in an effort to increase enrollment in the university's computer science program. Enrollment figures for the last decade and a half showed a significant decline after the dot-com bubble burst in early 2000. Malan suspects that this decline stemmed from misconceptions about computer science and the relatively uninteresting nature of many introductory programming projects (e.g., writing programs with a command-line interface vs. a GUI).

To make the course more interesting, Malan discussed using more languages than just C (e.g., PHP) and providing students with frameworks to write more sophisticated programs. These frameworks also serve to acquaint students with reading code. In addition, Malan emphasized the importance of assigning programming projects that solve more interesting problems, such as implementing a Vigenère cipher with arrays or a competition to come up with the fastest spelling checker.

After outlining his approach to teaching the course, Malan began discussing the role of cloud computing. Malan's goal was to acquire a set of machines with unfettered root access, which he could then configure for the students in the course. Although his group examined the possibility of operating their own cluster, they concluded that because of limited space, power, and cooling, it would be easier to offload everything to Amazon's EC2 service. In Malan's experience, launching a group of virtual machines on EC2 was much easier than setting up the infrastructure themselves.

Observed benefits of using Amazon's EC2 cloud infrastructure for course work were numerous. The number of virtual machines assigned to the cloud was scalable, and it was easy to start additional virtual machines during periods of high activity, such as the night before an assignment was due. The students found that using the virtual machines was straightforward because access was available through the host name cloud.cs50.net. This single host would pseudo-randomly assign each user to one of the cloud's virtual machines.

Using Amazon's EC2 also involved some costs. Malan estimated a cost of $15/student for the semester, or $5000 in all, but believed that additional work on his part could drive this cost down to $2000–$2500. Bandwidth was a particular concern, because it can be expensive. Learning EC2's idiosyncrasies was also troublesome; in the past, the department's IT staff took care of infrastructure issues, but with EC2, the onus was on Malan and his staff to keep things running smoothly.

One audience member asked whether the term "sprites" was a spoof or pun after Osterhout's Sprite research. The question was in reference to Malan's use of MIT's Scratch programming environment, which used objects called sprites. Malan replied that the name was entirely courtesy of MIT's Media Lab. Someone else pointed out that Malan's results showed an increase in course enrollment during the first week, but it wasn't clear whether these students were doing any better later in the course. Malan responded that there was not yet enough data to answer definitively, but that there has been an uptick in the number of students selecting computer science as a major.

NETWORKING

Summarized by John McCullough (jmccullo@cs.ucsd.edu)

- ***Design and Implementation of TCP Data Probes for Reliable and Metric-Rich Network Path Monitoring***
  Xiapu Luo, Edmond W.W. Chan, and Rocky K.C. Chang, The Hong Kong Polytechnic University, Hong Kong

Luo observed that Internet measurement can be very challenging. ICMP packets frequently have different behavior and can only measure limited metrics. He introduced OneProbe, which enables the measurement of TCP-based applications' specific behavior and can capture RTT, directional packet loss, and packet reordering. The current incarnation operates over HTTP, and the approach should be extensible to additional TCP-based applications.

OneProbe operates using a pair of probing packets. By observing the sequence and acknowledgment numbers in TCP packets and distinguishing TCP data packets and TCP control packets through packets' payload size, the responses can be classified into one of 18 cases to determine reordering and loss on both forward path and reverse path. Thus, OneProbe is able to achieve more expressive measurements against almost any Web server and provide more accurate results than httping. Luo showed results of latency measurements for the Web servers of the 2008 Olympic Games: they were able to observe diurnal RTT and loss behavior, and a significant difference between OneProbe and ICMP echo result on some paths. See http://www.oneprobe.org for more information.

An audience member asked how the RTT tests can be accurate for forward and reverse paths while using TCP. Another audience member inquired about the requirements on the application protocol. Luo answered that servers must send back some data packets and that clients need to be able to send data back to the server.

- ***StrobeLight: Lightweight Availability Mapping and Anomaly Detection***
  James W. Mickens, John R. Douceur, and William J. Bolosky, Microsoft Research; Brian D. Noble, University of Michigan

Mickens observed that we typically like to know the status of hosts in our networks. This can sometimes be achieved using distributed systems or other monitoring mechanisms, but it requires host modifications and can sometimes lead to scalability concerns. Mickens introduced StrobeLight, a system targeted to measuring networks of a few hundred thousand hosts. StrobeLight simply sends ICMP probes to every host on the network every 30 seconds, providing fine-grained fingerprints for availability data. This data can be used to guide choices in building multicast trees, task allocation, or identifying misbehaving networks or network hosts.

StrobeLight was designed to be simple and unintrusive without requiring infinite scaling. It operates by extracting the list of hosts from the DNS server and pinging each of them. The availability data is used to construct a per-subnet bit-vector where each position represents the availability of each host. Using a similarity metric, Mickens shows that most subnets do not change in character over time unless there is an anomaly. In one case, this assisted in identifying subnets that lost connectivity, and it can be used in general to identify routing anomalies such as BGP hijacking. Using an external wide-area prober, Mickens demonstrates that the fingerprints are generally similar and that in simulation they were effective in identifying hijacking attempts.

An audience member asked whether the hijacker could prevent detection by mimicking the host availability of the target network. Mickens replied that you'd be relatively powerless if the attacker could duplicate the availability profile, but that access to the network availability can be restricted to designated probers. Overall, the task is challenging, but StrobeLight is a simple first cut.

- ***Hashing Round-down Prefixes for Rapid Packet Classification***
  Fong Pong, Broadcom Corp.; Nian-Feng Tzeng, Center for Advanced Computer Studies, University of Louisiana at Lafayette

Pong established the need for fast packet classification that is both dynamic and compact. Typical approaches use either decision trees or hash tables. Decision trees can be tall and take a while to traverse, and the addition or deletion of a rule can necessitate a reconstruction of the entire tree. Hash-table approaches often require many probes to determine the correct prefix length and, in some cases, use supplementary decision trees to select the correct prefix length. Instead of storing a single address and mask pair at each entry in the hash table, HaRP (hashing round-down prefixes) lumps groups of prefixes into sets that can be searched in parallel. This reduces the number of hash lookups and reduces memory requirements.

Because HaRP probes all prefix-group buckets and because prefixes are transitive, it is possible to load-balance the hash table by placing shorter prefixes in longer buckets. HaRP also allows for either source IP hashing or destination IP hashing. Pong demonstrated results for six rule sets: three from practice and three artificially enlarged. Although HaRP does not hash-load-balance quite as well with many short prefixes, it enables a compact representation that can fit in cache and overall achieves approximately a 5x speedup. Data structure updates are much quicker than the several minutes that can be required for a large decision tree.

An audience member questioned the applicability of rapidly changing firewall rules. Pong pointed out that in VPN settings there can be frequent creation and deletion of firewall rules as clients enter and leave the system. Does the order of rule insertion affect the layout and hash-table load? Their first cut of choosing the first fit has worked well, and he also notes that finding the optimal fit is an NP problem.

FILE AND STORAGE SYSTEMS

Summarized by Alex Rasmussen (alexras@acm.org)

- ***Tolerating File-System Mistakes with EnvyFS***
  Lakshmi N. Bairavasundaram, NetApp, Inc.; Swaminathan Sundararaman, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

  Awarded Best Paper!

Swaminathan Sundararaman presented EnvyFS, a file system based on N-version programming that is designed to tolerate silent file system failures. Sundararaman argued that modern file systems are very complex and that this complexity, combined with the increasing depth of the stack between the file system and the disk (imposed by virtualization, networked file systems, etc.), admits the possibility of a wide range of failures, including so-called "fail-silent" failures in which the file system doesn't detect an error and continues to work on the wrong data, causing data corruption and returning bad data to the user. EnvyFS copes with fail-silent failures through N-version programming, which involves executing several different functionally equivalent programs (in this case, file systems) and using majority consensus to agree on the federated program's output. To reduce the storage and performance overheads imposed by using several child file systems at once, the authors created a single-instance store called SubSIST that de-duplicates data while retaining most of the reliability benefits the N-version file system provides.

Sundararaman presented several examples of individual fail-silent errors that resulted in corruption in the ext3 file system, but that EnvyFS (using ext3, JFS, and ReiserFS as its child file systems) is able to tolerate. In one case, EnvyFS masked an ext3 error that would otherwise have caused a kernel panic.

One audience member wondered whether EnvyFS assumes that all file systems have the same block size and how EnvyFS would deal with an extent-oriented file system. Sundararaman replied that EnvyFS assumes 4 KB blocks and that, if extent-oriented file systems wrote at non-block-aligned offsets, a more sophisticated scheme such as fingerprinting would have to be used to identify duplicate blocks. Had the file system authors had been informed of the bugs uncovered during the evaluation? Yes, they had. Did correlated failures relate to file systems having copied code from one another? They had not noticed any instances of code copying, but different file systems can be chosen to minimize the presence of duplicate code if such code is observed.

- ***Decentralized Deduplication in SAN Cluster File Systems***
  Austin T. Clements, MIT CSAIL; Irfan Ahmad, Murali Vilayannur, and Jinyuan Li, VMware, Inc.

Austin Clements presented a new method of de-duplication in storage area networks (SANs). De-duplication prevents duplicate data from being stored on disk by tracking the locations of written blocks in an index and bypassing writes to disk if the block to be written is already in the index. Clements asserted that classical methods of de-duplication do not work well in a decentralized setting due to cache coherence problems, the need for coordinated allocation of disk space for new blocks, loss of disk locality on individual disks, and a shared index structure to which access must be coordinated using locks.

To solve these problems, the authors have developed DeDe, which breaks de-duplication into three stages: write monitoring, local de-duplication, and cross-host de-duplication. DeDe can de-duplicate live storage devices out-of-band and in large batches. The system is designed to minimize contention on the shared index and communication between hosts, is resilient to stale index information, and improves access to unique blocks by allowing them to be mutable and to remain sequential on disk.

The authors evaluated DeDe on a corporate virtual desktop infrastructure and found that it was able to compress 1.3 TB of non-zero data to 237 GB while using only 2.7 GB of disk space for its data structures and causing no additional I/O overhead.

An audience member wanted to know how effective DeDe would be if de-duplication were done at the file level as opposed to the block level. DeDe (and de-duplication in general) would not be as effective in this case, since the opportunity for savings decreases as the size of the unit of replication increases. Might DeDe cause fragmentation and interfere with linear read-ahead? While any de-duplication system has these issues to some extent, DeDe suffers less from these problems, because it keeps blocks in their sequential location on disk whenever possible and permits in-place updates to blocks without duplicates. Would certain kinds of access patterns lead to poor performance with a de-duplication system that uses fixed-size chunking, as DeDe does? Many systems in this space use variable-size Rabin fingerprinting to overcome this issue, but Clements speculated that such fingerprinting is unlikely to be worth the performance penalty of managing variable-size blocks in a live, shared file system. What might happen when a lot of duplicate data is injected into the system suddenly, such as when all VMs in the network are patched in rapid succession? Increased duplication would trigger de-duplication more frequently, but such temporary increases in duplicate data are hard to deal with in general and some extra storage space must be allocated to deal with this eventuality.

- ***FlexFS: A Flexible Flash File System for MLC NAND Flash Memory***
  Sungjin Lee, Keonsoo Ha, Kangwon Zhang, and Jihong Kim, Seoul National University, Korea; Junghwan Kim, Samsung Electronics, Korea

Today's NAND flash memory comes in two main varieties: SLC (single-level cell) and MLC (multi-level cell). SLC has higher performance and lasts longer than MLC, but MLC has higher capacity. However, MLC flash memory can be programmed dynamically as either MLC or SLC through use of a special writing method. Sungjin Lee described FlexFS, a new file system that combines the performance of SLC flash with the capacity of MLC flash. It does this by managing disk blocks as three separate pools for SLC, MLC, and free blocks. Blocks are dynamically allocated and migrated between regions in the background. New data is written to the SLC region and blocks are migrated to the MLC region in the background as the SLC region becomes full. FlexFS also takes advantage of idle time to generate free blocks for the SLC region and avoids migrating "hot" (recently referenced) pages. In addition, FlexFS's wear manager controls the rate at which erase operations occur, to maximize the device's lifetime.

An audience member wanted to know if FlexFS, which was targeted at mobile systems, could be applied to large disks, where capacity is less of an issue, and to environments where the system could take advantage of write caching. Lee responded that FlexFS could certainly be extended to support such environments.

- ***Layering in Provenance Systems***
  Kiran-Kumar Muniswamy-Reddy, Uri Braun, David A. Holland, Peter Macko, Diana Maclean, Daniel Margo, Margo Seltzer, and Robin Smogor, Harvard School of Engineering and Applied Sciences

Provenance is metadata that describes the history of an object. Such data is useful for scientific reproducibility, business compliance, and security. Previously, the authors constructed PASS, which observes file system calls to infer relationships between objects. However, if an application such as a Web browser also tracks provenance, no method exists to link the provenance tracked by the application with that tracked by the kernel. Kiran-Kumar Muniswamy-Reddy discussed the Disclosed Provenance API (DPAPI), through which software that tracks provenance can disclose that provenance to lower layers of the software stack in a secure, modular way. DPAPI can pass abstract provenance-containing objects between programs through use of opaque handles and has functions to associate provenance with reads and writes, thus ensuring that provenance is consistent with the data it describes. Muniswamy-Reddy then described the use of DPAPI in Kepler (a provenance-aware workflow engine) that links the Web browser and the Python interpreter. He concluded with some lessons learned by the authors in writing DPAPI. Among these lessons learned are that it is not easy to make applications provenance-aware and that making platforms provenance-aware does not necessarily provide provenance awareness to all applications running on that platform.

One audience member wondered whether there was some notion of nested provenance, where, for example, each tab tracks its provenance and the browser tracks the "meta-provenance" of the collection of tabs. Muniswamy-Reddy replied that the browser knows where each URL came from and the chain of URLs the user viewed in the past and so can do some logical separation within itself, but that the clarity of the separation isn't clear. He mentioned that they are looking at Google Chrome as a better target for DPAPI than Firefox, due to Chrome's use of a separate process per tab. Why is the file system the right location at which to focus provenance tracking, since the application is so much more aware of what is actually being done at a high level? The file system is a single point with which all processes must eventually interact and, while its provenance is incomplete, is a piece of the larger provenance puzzle; DPAPI helps to integrate it with the rest of the software stack.

INVITED TALK

- ***Project SunSPOT***
  Roger Meike, Sun Microsystems

Summarized by Alex Rasmussen (arasmuss@cs.ucsd.edu)

Roger Meike from Sun Labs provided an overview of SunSPOT, Sun's research and development platform for studying the entire embedded systems software and hardware stack as part of its Internet of Things Actualized initiative. SunSPOTs run Java, come equipped with a variety of sensors and affectors, and can communicate with one another wirelessly. Additionally, the SunSPOT platform comes with a large number of libraries, and the device itself is modular, allowing users to install custom or preconfigured sensor boards to fit their application.

The majority of the talk focused on projects that have used SunSPOTs. Meike gave examples ranging over several domains, including toys, research-oriented sensor networks for environmental monitoring, autonomous robots, and art installations. He also discussed some work that has been done to build a community around the SunSPOT platform; Meike believes that allowing the SunSPOT community to largely integrate itself into existing social networks (through use of the #spaught Twitter tag, for example), rather than creating a domain-specific social network, has helped the community grow beyond a core group of SunSPOT enthusiasts.

When asked about the weirdest thing anyone has ever done with a SunSPOT, Meike replied that they once taught a SunSPOT Morse code and created a translator that would receive Morse code sent wirelessly by one SunSPOT and translate it into semaphore.

Meike provided a number of pointers to more information about the SunSPOT platform. sensor.network.com provides information about various SunSPOT installations. See http://sunspotworld.com and http://spots.dev.java.net for more information about the platform and existing applications.

POSTER SESSION

Summarized by Chris Frost (chris@frostnet.net) and Rik Farrow (rik@usenix.org)

- ***SPROV: A Library for Secure Provenance***
  Ragib Hasan, University of Illinois at Urbana-Champaign; Radu Sion, Stony Brook University; Marianne Winslett, University of Illinois at Urbana-Champaign

SPROV, an application-layer library for secure provenance, intercepts file-system system calls and logs file modifications and other lineage information. Cryptographic components of the provenance chain allow verification of the integrity of these provenance records at any point in the future. This capability allows one to verify the edit history of a document. For most common real-life workloads, SPROV imposes runtime overheads of 1–13%. For more information see http://www.usenix.org/publications/login/2009-06/openpdfs/hasan.pdf and http://tinyurl.com/secprov.

- ***Towards a Formally Verifiable Multiprocessor Microkernel***
  Michael von Tessin, NICTA, University of New South Wales

Michael von Tessin presented his work on formal verification of the seL4 microkernel. He is working to extend the completed proofs for the uniprocessor version and make them work in a multiprocessor setup. To reduce complexity introduced by concurrency, he identified two orthogonal approaches: The first is to use one big lock around the kernel to reduce parallelism. The second is to reduce sharing by having a multikernel architecture.

- ***Sonar-Based Measurement of User Attention***
  Stephen P. Tarzia, Northwestern University; Robert P. Dick, University of Michigan/Northwestern University; Peter A. Dinda and Gokhan Memik, Northwestern University

Stephen Tarzia explained how they are using sonar to monitor user activity. Unlike typical activity monitors, such as mouse or keyboard event monitoring, sonar can detect if there is a person sitting in front of a keyboard. The sonar data is easy to analyze and will be used in power management, such as screen dimming.

- ***Including the Network View in Application Response Time Diagnostics using Netflow***
  Jochen Kögel, University of Stuttgart

Jochen Kögel presented a use of router-provided flow-level data, Netflow, to diagnose network issues and their impact on application response time in global enterprise networks. Today Netflow data is only used for reporting, accounting, and security, in part because of its incompleteness caused by hardware logging limitations. However, Jochen showed how network round-trip times can be separated from server response times, how packet loss can be traced to particular network segments, and how one-way network delays can be measured with Netflow data.

- ***Dynamic Resource Management Through Transparent Interaction Monitoring***
  Igor Crk, Mingsong Bi, and Chris Gniady, University of Arizona

In this work, the presenters include context while monitoring user behavior. Simply monitoring user events, such as mouse and keyboard events, doesn't provide enough information to predict when a hard drive or network interface should be suspended or awakened, or the CPU run at a slower clock rate. Using the mouse to open a File dialog suggests that the hard disk should be spun up in anticipation of a read or write. This work builds on their 2008 USENIX Annual Technical Conference paper (http://www.usenix.org/event/usenix08/tech/full_papers/crk/crk_html/index.html).

- ***Software Configuration by the Masses***
  Wei Zheng, Ricardo Bianchini, and Thu D. Nguyen, Rutgers University

This poster focused on early research trying to help new users configure our increasingly flexible software systems. They plan to collect existing users' configurations and the corresponding effects (i.e., performance metrics) to automatically recommend configurations for new deployments. They hope this will help new deployments by determining the sequence of configurations to try whose values are most likely to achieve the target performance on the new deployment in a descending order. They also estimate the number of experiments for a target performance to enable explicit tradeoff between performance target and configuration tuning. Open questions include how to obtain existing configurations, how varied configurations are, how to combine existing configurations with expert data, and how to deal with software evolution. For more information see http://vivo.cs.rutgers.edu/.

- ***FlexFS: A Flexible Flash File System for MLC NAND Flash Memory***
  Sungjin Lee, Keonsoo Ha, Kangwon Zhang, and Jihong Kim, Seoul National University; Junghwan Kim, Samsung Electronics

There exist two types of NAND flash in today's products: SLC is fast and supports a large number of block erases, and MLC supports large capacities. FlexFS is a file system for embedded mobile systems that can use MLC hardware to provide the benefits of both MLC and SLC flash by treating the hardware as MLC or SLC on a per flash-block basis. FlexFS provides the larger capacity of MLC flash to end users, but strives to write as much data as possible to SLC flash blocks to maximize I/O performance. FlexFS also provides a mechanism that mitigates the poor wear characteristics of MLC flash. Their paper was presented during USENIX Annual Tech '09 (see above).

- ***Zephyr: Efficient Incremental Reprogramming of Sensor Nodes using Function Call Indirections and Difference Computation***
  Rajesh Krishna Panta, Saurabh Bagchi, and Samuel P. Midkiff, Purdue University

Zephyr reduces the size of software updates for sensor nodes with a goal of improving battery life. By updating using a modified rsync to reduce the amount of program data required for patching, and using function call indirection, Zephyr requires much less energy for both network and flash storage. Panta also presented on Zephyr during the conference.

DISTRIBUTED SYSTEMS

Summarized by Kiran-Kumar Muniswamy-Reddy (kiran@eecs.harvard.edu)

- ***Object Storage on CRAQ: High-Throughput Chain Replication for Read-Mostly Workloads***
  Jeff Terrace and Michael J. Freedman, Princeton University

Internet storage providers have started providing object storage with eventual consistency semantics. Eventual consistency, said Jeff Terrace, is difficult to program, as it can return stale data on reads to users. The traditional strong consistency is easy to program but hard to scale. This work introduces CRAQ (chain replication with apportioned queries), a storage system that provides strong consistency while also ensuring high scale and availability. CRAQ is an improvement over the chain replication (CR) method. CR organizes all nodes storing an object in a chain, with the head of the chain processing writes and the tail of the chain processing reads. On a write, the head propagates modifications along the chain and acknowledges the write to users once the write has propagated to the tail. On a read, the tail returns the value it has stored for the object, thus ensuring strong consistency. Since all reads are served by the tail node in CR, the tail can be a potential hotspot. CRAQ improves on CR by taking advantage of the fact that all the nodes on the chain have replicas of the data and can serve read requests. CRAQ uses the following scheme to ensure strong consistency in the event that reads are issued while an update is propagating through the chain. Each node that has dirty data (i.e., data that has not yet been propagated to the tail) queries the tail for the current version number of the data and returns that version of the data to the user. The overhead on the tail in CRAQ is smaller than in CR, due to the fact that the tail has to reply with metadata, as opposed to the whole object in CR.

CRAQ can also provide eventual consistency if it is sufficient for the applications, reducing the number of operations across data centers compared to CR. Since one can look up objects from any node in the chain in CRAQ, one can look up objects from the nodes in the local data center, whereas in CR, one has to send the request to the data center that has the tail node. Users can also configure CRAQ in a variety of ways. For example, to ensure datacenter diversity, users can specify the data centers to be used for a chain and the chain size in each data center. Terrace presented evaluation results comparing CRAQ and CR on Emulab: the results confirm that CRAQ scales better than CR.

One audience member commented that Hadoop could really use this scheme for appends and requested that the authors consider contributing this to the Hadoop project. Sourav Bagchi asked what happens if two separate writes originated for the same object in two data centers. Terrace replied that there is a statically defined master data center that coordinates the writes. How might the system change if the operations were persistent as opposed to in-memory (as it is now)? The system would change but the protocol would still be effective.

- ***Census: Location-Aware Membership Management for Large-Scale Distributed Systems***
  James Cowling, Dan R.K. Ports, Barbara Liskov, and Raluca Ada Popa, MIT CSAIL; Abhijeet Gaikwad, École Centrale Paris

Dan Ports presented Census, a membership service designed to work in wide area locality-aware large-scale systems—hence, a platform for building large-scale distributed systems that have to deal with constant churn. Census divides time into epochs and provides consistent membership views to all nodes in a single epoch. Many of the previous membership services were restrictive in that they provided only partial views of system membership, whereas Census provides stronger consistent semantics, thus making applications easier to develop. The basic approach of Census is to designate one node as a leader. The other nodes then report any membership changes to the leader, and the leader aggregates these changes and multicasts the updated membership to members. In the next epoch, members update their membership views.

In order to reduce load on the leader, Census divides the nodes into a hierarchical structure based on network coordinate locality of the nodes. The hierarchical structure is constructed by exploiting the membership knowledge of the system. Since there is a consistent membership view, nodes can reconstruct the tree on the fly, and there is no protocol overhead even during churn. For very large networks, nodes are grouped into regions according to their network coordinates. For such networks, Census makes an exception, and the membership knowledge of nodes is restricted to the nodes in their region. Each node has a summary of membership in other regions. Census uses standard state replication techniques for fault tolerance and can optionally deal with Byzantine faults. An evaluation of Census shows that it imposes low bandwidth overhead per node, reacts quickly to churn, and scales well.

One audience member questioned whether it is possible for branches to occur due to two nodes having different views of the membership tree. Ports replied that there can be temporal inconsistencies, but the nodes can use the version number in each epoch to resolve inconsistencies. Does Census require nodes to store old versions of the views? It helps to have a few recent membership views. How does Virtual Synchrony compare with Census? Ports replied that Virtual Synchrony was more rigorous, but it is similar to their scheme. Does the duration of the epoch affect the scalability? If the epoch is small, their scheme can have slightly higher overhead, but it does not affect scale.

- ***Veracity: Practical Secure Network Coordinates via Vote-based Agreements***
  Micah Sherr, Matt Blaze, and Boon Thau Loo, University of Pennsylvania

Network coordinates (NC) are a decentralized mechanism to estimate approximate network distances between hosts without performing a pairwise measurement. However, NC systems are easy to manipulate. If 10% of the nodes are malicious, there is a 4.9x decrease in accuracy, and if 30% of the nodes are malicious, there is an 11x decrease. Micah Sherr presented Veracity, a security protection layer for NC systems. It differs from existing solutions in that it assumes no triangular invariants, is fully distributed (other schemes assume *a priori* trusted nodes), supports dynamic neighborsets, and does not assume temporal locality.

Participants in Veracity are either publishers or investigators. An investigator is a node that wants to use the publisher's coordinate to update its own. When a publisher returns its coordinate to the investigator, the coordinate is verified by a set of verification nodes (a deterministic set of peers of the node) before the investigator uses it. A malicious node that tries to publish incorrect coordinates will fail this step. After verification, the investigator updates its own coordinate based on the publisher coordinate and the RTT between it and the publisher. The publisher can delay its response to the investigator's probe and induce an error in the investigator's coordinate computation. In the second step, the investigator updates its coordinate only if the new coordinate results in an error below a threshold when computed against a random set of peers. Veracity is implemented by modifying the Vivaldi implementation that is packaged with Bamboo. The authors demonstrated the effectiveness of Veracity under a variety of attacks.

John Dunagan commended Sherr for the thoroughness of their evaluation, but wondered if random delay is the worst an attacker can do. Sherr replied that they looked up all attacks in the literature and came up with some of their own attacks. Further, Sherr agreed that it is hard to assert that Veracity is resilient against all attacks, so they are trying to formalize and verify their system. Could jitter on the WAN affect Veracity? NC systems handle many of these issues. Veracity doesn't distinguish between malice and temporary effects. Veracity does allow users to tweak knobs so that they can handle corner cases in network behavior.

KERNEL DEVELOPMENT

Summarized by Ragib Hasan (rhasan@uiuc.edu)

- ***Decaf: Moving Device Drivers to a Modern Language***
  Matthew J. Renzelmann and Michael M. Swift, University of Wisconsin—Madison

Matthew Renzelmann presented Decaf, a tool for moving device drivers from the kernel to user space. Driver programming and debugging are difficult tasks, and complications can lead to driver unreliability. Renzelmann argued that writing drivers in type-safe high-level languages such as Java can alleviate these problems, but may introduce performance degradation. Decaf solves this by moving most of the driver functionality to user-mode code written in Java, with a only small amount of code running inside the kernel. Decaf provides a migration path for porting existing kernel drivers to user mode. This also makes writing patches easy, to evolve drivers over time.

Decaf builds on the authors' previous work on microdrivers, allowing one to write drivers from scratch and migrate existing drivers. The programmer annotates legacy drivers and then uses the tool DriverSlicer to split the driver code into a nucleus (which runs in kernel mode) and a user-mode library. The developer can then migrate code from the driver library to the Java Decaf driver one function at a time. For example, in porting the ENS1371 sound card driver, Decaf uses Jeannie to allow C and Java code to be mixed in the same file. Complex Java/C transfers are handled using XPC. The authors evaluated Decaf by migrating five existing drivers, showing that porting most of the functionality to user-mode Java code can still provide reasonably good performance.

A member of the audience asked if the authors had studied the memory overhead. Renzelmann replied that there is roughly a 3x memory overhead. Since there was no Java code invoked during the benchmarks, how can the authors be sure that Decaf will correctly handle bugs in the driver? The Java code did run during driver initialization, and the only code left in the kernel mode in C is for faster performance. To a question about refactoring, Renzelmann mentioned that the object-oriented features of Java allowed reduction of the code size for the e1000 driver by 6.5 KB. To a question about the observed bug distributions, Renzelmann referred to their earlier microdrivers study, where they found bugs to be uniformly distributed between kernel and user mode. Alan Thai inquired about performance optimization, and Renzelmann said that they did not use any multi-threading or other optimizations. An audience member who recently discovered a bug in the e1000 driver asked if the authors performed any detailed bug analysis. They had not. How much performance degradation occurred by rewriting the C driver code into Java? Performance loss was not substantial, largely because most of the overhead is in control and in data transfer between user and kernel modes.

- ***Rump File Systems: Kernel Code Reborn***
  Antti Kantee, Helsinki University of Technology

Antti Kantee presented his work on reusing kernel code in user space. A large portion of the kernel code can be run in the user mode with no modifications. Reusing kernel code in user-space applications saves reimplementation time. The Rump File system runs on NetBSD, where it allows different file system codes as user-space processes. The author defined a Rump as a "runnable userspace meta program," i.e., a user-space program which runs kernel code, and a framework that allows this. The Rump kernel runs inside the host OS kernel. The author's goal was to make kernel development simpler. Currently, kernel developers need to use user-mode OS, virtual machines, or emulators to develop and debug kernel code. By allowing unmodified kernel code to be run from user space, debugging and development become easier.

Rump works by using as much kernel code directly as possible. Rump has two modes: a mounted server mode, which is transparent to the applications but where mount privileges are required, and an application library mode, where the application needs explicit modifications but can run with no special privileges. Kantee gave an example scenario in which a corrupt file system on a USB stick can cause a system crash or have exploits. This can be avoided by mounting the device as a Rump file system in user space, thereby isolating the damage to a user-mode process. Rump also makes kernel debugging easier, as various debuggers can be used to give even non-experts control over the debugging process. Kantee talked about a Google Summer of Code project that implemented an application suite providing mtools-like functionality for all file systems supported by Rump. He also showed that Rump is maintainable, with only a small number of Rump breakage commits in the NetBSD repository.

Is the buffer management layer still kept inside the kernel? Kantee answered that Rump uses double buffering, with both the kernel and the application maintaining its own buffer. However, the double buffering is a temporary workaround, which Kantee plans to fix. Is it obvious which part of the interface to port and which one to rewrite? It is not obvious—it's mostly gut feeling.

- ***CiAO: An Aspect-Oriented Operating-System Family for Resource-Constrained Embedded Systems***
  Daniel Lohmann, Wanja Hofer, and Wolfgang Schröder-Preikschat, FAU Erlangen—Nuremberg; Jochen Streicher and Olaf Spinczyk, TU Dortmund

Wolfgang Schröder-Preikschat presented their paper on using aspect-oriented programming in embedded operating systems. Embedded operating systems are widely used in different devices, but to handle different architectures, the code becomes very complex with the use of #ifdef. In the eCos operating system, for example, only two lines of fundamental code requires 34 lines of ifdef blocks, spread over seventeen functions and data structures in four implementation units. This problem, also known as "ifdef hell," makes software complex and difficult to maintain. Schröder-Preikschat argued that aspect-oriented programming (AOP) can solve this by modularizing cross-cutting concerns. After a brief introduction to AOP, the presenter showed how they used AspectC++ (an extension to C++) along with the source-to-source weaver tools to generate normal C++ code from a given aspect specification.

AOP can help here by eliminating the ifdefs. They used this approach in the CiAO system to design configurable embedded-system software. CiAO has loose coupling, visible transitions, and minimal extensions. Important state transitions are captured by a point-cut expression in the aspect. The resulting base system is designed with classes, and most functionality is provided by optional aspects—extension aspects, policy aspects, and upcall aspects. Schröder-Preikschat gave an example of an extension aspect that uses task scheduling. For evaluation, he mentioned their collaboration with Audi and Elektrobit, where they showed that CiAO runs on a number of microcontroller-based systems. He also compared CiAO with OSEK. Finally, Schröder-Preikschat talked about how using aspects for low-level code can break fragile join points and about other issues related to aspect-aspect interdependencies, such as join point traceability and granularity.

Remzi Arpaci-Dusseau from the University of Wisconsin asked whether the ordering of advice procedures has to be considered in the structure definitions. Schröder-Preikschat replied that advice ordering tells in what order the aspects are going to be intermixed in the source code, and so the use of a C pointer may cause some problems. Can aspects be applied at the level of statements? Fine-grained aspect-oriented programming would require using empty functions around statements, but there are not very many cases like that. Sourabh Bagchi from Purdue asked whether using this will be cost-effective, in terms of the effort spent in defining aspects, in real-life scenarios. Configurability of a system will forever be a problem, for example, in the auto industry, which needs very deterministic behavior from their embedded systems.

INVITED TALK

- ***Towards Designing Usable Languages***
  Matthew Jadud, Allegheny College in Meadville, PA, and Christian L. Jacobsen, Untyped Ltd.

Summarized by Michael von Tessin (mtvt@cse.unsw.edu.au)

Edit, compile . . . edit, compile. This tireless cycle dates back to the 1960s, when the cost of editing and compiling was substantial.

Despite this by now long-standing interaction between human and computer, the observable behavior of novice programmers has only recently been linked with negative affective states. That's a fancy way of saying, "We can detect when students are frustrated." The short-term goal in this study is to support the learner with sensible interventions based on automated observation of their interactions with the compiler and their environment. The longer-term goal is to help provide a human-centered foundation for the design of languages and their environments.

To this end, the language design target is ambitious: parallel languages for robotic control. The authors have built a small virtual machine to support message-passing parallel languages (the Transterpreter) and have begun exploring its use on small, microcontroller-based mobile robotics platforms. They felt this was good engineering: begin by exploring, understanding, and reusing well-tested and formally verified languages with a rich 20-year history. They also thought more people might use the tools if they could play with them on robots made out of shiny yellow plastic. In short, this is a story about people trying to do some cool stuff at the intersection of usability research and the design and implementation of parallel-programming languages.

BlueJ is a development environment used at the University of Kent to teach novice students OO programming in Java. It allows classes, object instances, and invocations to be created/executed via a graphical interface. Skeleton code is automatically created and helps students to start writing code instead of just presenting them with a blank page. BlueJ can be used to trace all compiler invocations, i.e., record the time of invocations and their results (warnings, errors). Their study covers about 42,000 programs in 2000 sessions of 120 students over two years. Overall, 56% of all compiler invocations resulted in a syntax error. The top errors were: unknown variable, missing semicolon, missing bracket, unknown method, app. error, illegal start of expression. They were able to spot compiler error messages that can completely confuse students and frustrate them.

Midway through the talk, the speakers opened the floor to questions. If students know their activities are monitored, does it affect their behavior? They haven't explicitly looked at that, but if monitoring makes them think harder before they hit "compile," that would be an interesting outcome. Someone else asked about variation in traces over the course of a semester. They did see error rates vary and in the types of errors change as students' skills evolved.

When teaching parallelism to students, the speakers continued, they don't want to use an existing sequential language, because the compiler doesn't know how to help students (or even confuses them). Thus, they have chosen occam, which is used a lot at the University of Kent. They want learning parallelism to be fun, but also authentic. "Authentic" in robotics means that although there is a basic sequence—sense → think → act—this is never a strict sequence; you can have multiple inputs (sense), multiple outputs (act), and multiple tasks/calculations (think phases) running in parallel and interconnected.

The authors want the ability to reach out to a community of tinkerers and explorers and the hardware they use in the classroom to be affordable. A good choice is Arduino (http://arduino.cc), which only costs $20 and allows you to buy, download, or build your own software and has a large community behind it. The authors know that designing a new language is hard to do well and to get right. On the other hand, Java was not designed for novice programmers. There were some good examples (Logo and Lego Mindstorms), but they eventually want to get rid of the standard edit/compile cycle.

One speaker said, "In 10 years from now, I don't want my son (now 13 weeks) to learn how to program embedded systems in C. I want tools to be designed for him. So please engage and have a look at baseplate.org and transterpreter.org."

There were some concluding questions. Have you contacted psychologists or other experts in child development? Not yet. That would be future work and very interesting. What motivated you to choose occam? Occam has a long British tradition (Tony Hoare, Bristol) and Kent has a long tradition in that space. Erlang (which is heavily used in telco) has a huge runtime, so we moved away from Lego Mindstorms and started to use occam, which has a very small runtime and memory footprint.

AUTOMATED MANAGEMENT

Summarized by Xu Chen (chenxu@umich.edu)

■ ***Automatically Generating Predicates and Solutions for Configuration Troubleshooting***
Ya-Yunn Su, NEC Laboratories America; Jason Flinn, University of Michigan

Su observed that troubleshooting computing systems is hard. There have been automated troubleshooting tools proposed, but they rely on a given set of predicates that can be used to determine good or bad system states. In this talk, Su proposed methodologies for automatically generating predicates. Existing approaches analyze source code or configuration files, but Su showed that predicates can be extracted from previous user or expert troubleshooting behaviors.

A modified shell is used to record human troubleshooting behavior, including commands executed and resulting kernel-object modifications, while being unobtrusive to the users. The basic assumption is that users usually use repeated commands as predicates, one for recreating the failure and one for evaluating the troubleshooting outcome. The results of these repeated commands should be different, in terms of exit code, screen output, or kernel objects modified. Causal dependencies of different commands are tracked by the modified shell such that the command that solved the problem can be identified, while pruning unrelated commands. To sanitize the user-submitted predicates, they are ranked according to popularity and merged based on the associated state delta.

Su demonstrated through a user study with 12 participants solving four configuration problems that the proposed method can extract correct predicates, with very few false positives (wrong predicates). The false positives are introduced because the users did not perform repeated predicates or did not solve the problem.

The audience raised questions regarding how to pinpoint the exact solution when the user changes a lot of different things. Su emphasized that their methodology currently works at kernel object level (e.g., a file). So the exact change made, such as which line in the file, cannot be determined. How are generated predicates applied to other environments? The generated predicates and solutions are canonicalized so that they can be applied to different users, as shown in their prior work, AutoBash.

■ ***JustRunIt: Experiment-Based Management of Virtualized Data Centers***
Wei Zheng and Ricardo Bianchini, Rutgers University; G. John Janakiraman, Jose Renato Santos, and Yoshio Turner, HP Labs

Managing a data center is hard. To predict system behavior resulting from configuration changes, Zheng proposed an experiment-based approach called JustRunIt. The assumption is that data centers usually run services in tiers, with many instances of VMs for each tier. The basic idea is to create sandboxed VMs to run alongside production VMs so that different parameters can be explored in the sandbox without affecting production services. The sandboxed VMs will be running clones of production VMs. For each tier, an in-proxy and an out-proxy are used to hand over to sandboxed VMs the traffic from production VMs. The management entities usually specify the ranges for different parameters and some time limit for experiments. JustRunIt will try to search the parameter space as much as possible within the time limit. Since the search space is large, interpolation is used if not all combinations are tested. As a simple heuristic, JustRunIt prioritizes the combination of the upper and lower bounds of different parameters.

JustRunIt is implemented in Xen and has been tested on a cluster of machines running multi-tiered Internet services. Evaluation results demonstrate that the overhead of JustRunIt is small. The accuracy of JustRunIt is very good: it can accurately predicate production service behavior in terms of mean response time and throughput.

Zheng presented two usage scenarios in which JustRunIt was used to estimate the impact of hardware upgrades and to derive a better resource allocation strategy in the context of an SLA violation.

Audience members raised several questions. How practical is JustRunIt when the parameter search space is very large? Even though JustRunIt can give an accurate estimate on mean response time, would the variance or exhibited distribution of response time be accurate as well? How does

JustRunIt compare to some greedy approaches in resource management, which, for example, just allocate more VMs if SLA is violated?

- ***vPath: Precise Discovery of Request Processing Paths from Black-Box Observations of Thread and Network Activities***
  Byung Chul Tak, Pennsylvania State University; Chunqiang Tang and Chun Zhang, IBM T.J. Watson Research Center; Sriram Govindan and Bhuvan Urgaonkar, Pennsylvania State University; Rong N. Chang, IBM T.J. Watson Research Center

Tak observed that enterprise services are usually multi-tiered, and a user request usually traverses the system after being processed by a variety of threads that communicate with each other. How each request moves through the system can be characterized by request-processing paths, which Tak tried to discover in his work. There are two types of existing approaches: statistical inference, which is flexible but may not be as accurate; and instrumentation-based, which is accurate but requires source code.

Tak proposed vPath, which resides in a virtual machine monitor and thus is transparent to the application running, yet does not impose too much overhead. vPath identifies the request-processing path by tracking internal and external causality between thread activities within and across machines. The VMM identifies threads and tracks their TCP behavior to identify causalities. This is possible because most commercial applications follow a multi-threaded structure and synchronous communication patterns. vPath seeks to demonstrate that the application model can be exploited to solve the problem of path discovery, presenting a new direction and paradigm.

Implementation-wise, vPath modifies the Xen VMM to intercept some system calls made within each VM. For the related system calls, threads are identified by inspecting the EBP register, while network socket information is delivered by hypercalls. For current support, a guest VM needs to be modified to invoke hypercalls, but in the future such functionality will be merged into the VMM, as Tak pointed out. While delivering accurate processing-path results, vPath exhibited about 6% overhead in increased response time and throughput reduction in a TPC-W test.

There were concerns from the audience about how applicable vPath is to complicated applications whose workload model deviates from vPath's assumptions, and about the benefits vPath could deliver compared to existing inference-based approaches.

SHORT PAPERS

Summarized by Stephen P. Tarzia (starzia@northwestern.edu)

- ***The Restoration of Early UNIX Artifacts***
  Warren Toomey, Bond University

Warren Toomey described efforts by himself and others at the UNIX Heritage Society to restore the first edition of UNIX from 1971. In addition to its historical interest, this case study serves as a lesson in preserving code for use in the distant future. Toomey's story began with the discovery of a printed assembly code listing for the first edition of UNIX. However, much more than the source code was needed. Running the code required several reconstructions, including correcting lines of code, adding peripheral hardware devices to the PDP-11 emulator, and rewriting system utility binaries.

Toomey observed that although software does not physically decay like hardware, it cannot run in a vacuum. The software and hardware environment that it requires is, of course, constantly evolving. This phenomenon is often called "bit rot." The key to preserving code is to preserve the environment as well. This includes keeping contemporary libraries, compilers, configuration files, hardware (or, preferably, a hardware emulator), documentation, anecdotes, and publications.

Toomey reported that a few first-edition UNIX bugs had been discovered during the restoration and had been jokingly posted as security advisories. An attendee asked how to best preserve this kind of restoration work. Toomey encouraged thorough documentation and replication.

- ***Block Management in Solid-State Devices***
  Abhishek Rajimwale, University of Wisconsin, Madison; Vijayan Prabhakaran and John D. Davis, Microsoft Research, Silicon Valley

Abhishek Rajimwale presented an analysis of how Solid State Drive (SSD) characteristics break file systems' long-standing assumptions, and he prescribed appropriate storage-stack changes aimed at improving SSD performance and longevity. In contrast with conventional, spinning magnetic disks, SSDs have low random-access latency, significant background activity (for cleaning and wear-leveling), significant media wear-down, and no seek delay. Additionally, they exhibit write amplification, meaning a small write results in the entire, much larger, block being rewritten.

Rajimwale and colleagues measured characteristics of several different SSD samples, gathered real file-system traces, and modified the SSD module extension for the DiskSim (PDL) simulator. He showed results quantifying the above characteristics and presented three optimizations based on these results. First, the device should merge write requests when possible; second, background maintenance operations should be stalled during high-priority I/O; finally, the SSD should be prevented from cleaning free disk blocks. To implement the above performance and longevity optimizations, Rajimwale called for a richer storage device interface. In particular, he proposed a higher-level interface such as object-based storage, to allow the SSD to manage block-level details itself.

Were there cases when the SSD should provide dynamic status information to the OS? If cells are being switched between multi-level (MLC) and single-level (SLC) modes, that information could be shared with the OS. Do conventional

RAID disk arrays share the same issues? Rajimwale agreed that write amplification has also existed in RAID, but said that the background activity found in RAID (namely, rebuilding) is relatively infrequent; SSD background activity may be continuous. Another attendee suggested letting the OS control low-level block management on the SSD. Rajimwale agreed that OS control is a viable alternative, but suggested that, as SSD devices get more complex, internal control is desired.

- ***Linux Kernel Developer Responses to Static Analysis Bug Reports***
  Philip J. Guo and Dawson Engler, Stanford University

Philip J. Guo presented an analysis of Linux kernel developer responses to bug reports generated by the Coverity static code analysis tool. Their basic goal is to evaluate such tools and to make them more useful by automatically prioritizing the thousands of generated bug reports by correlating bug reports from a single source snapshot to subsequent developer actions in the bug tracker and repository. Such bugs are either ignored or triaged, and the assumption was that developers ignored bugs because they were identified as less important or meaningful. They found correlations between triage rate and several factors such as error type, other static bugs, user-reported bugs, and file age, size, and location.

Guo argued that static analysis is indeed useful. Although static bugs are shallow in nature, he believes that their correlation results show that developers can be led by static bugs to deeper bugs. Quoted reactions from kernel developers support this hypothesis.

The first audience question was whether static-code-analysis tool use results in fewer bugs over time. Guo responded that, since the kernel source is growing, it is difficult to determine such trends. Can a code-complexity metric be used to find deep bugs? Guo supported this idea and pointed out that static analysis bugs often result from code complexity, so these ideas are actually complementary. Had the authors contacted lead kernel developers? Although their developer quotes were anonymized, some were from veterans.

- ***Hardware Execution Throttling for Multi-core Resource Management***
  Xiao Zhang, Sandhya Dwarkadas, and Kai Shen, University of Rochester

Xiao Zhang presented a new software-based multicore resource management mechanism called Hardware Execution Throttling. The general problem is core performance isolation. Adjacent cores typically share a last-level cache, so one core can slow down other cores by overusing the cache. Hardware Execution Throttling cleverly uses two features of Intel Core-series processors: duty cycle modulation (DCM) and cache prefetcher disabling. These settings can be quickly adjusted (within a few hundred CPU cycles) to reduce a core's shared-cache access rate. This fine-grained control is their primary contribution relative to previous mechanisms. Of course, core performance control policy is a separate problem which this work does not address.

Zhang described a fairness metric for concurrent applications. This measures the extent to which all applications are running at the same level of performance. They used a set of standard benchmark applications to favorably compare Hardware Execution Throttling's fairness to previous mechanisms.

An attendee noted that hyper-threading would cause pairs of threads to be throttled together. Zhang agreed and suggested pressuring CPU vendors for more flexible control. Would kernel activity on a throttled core be affected? The kernel could quickly de-throttle that core. Were there cases in which one of the two CPU features worked better for throttling? Zhang didn't have a concrete example, but reported that DCM has a more predictable effect than disabling prefetching. Responding to another question, Zhang said performance was not very sensitive to the particular choice of DCM setting used. Finally, an attendee pointed out that the throttling policy would have to follow a process as it migrated to different cores.

INVITED TALK

- ***The Antikythera Mechanism: Hacking with Gears***
  Diomidis Spinellis, Athens University of Economics and Business

  Summarized by Ragib Hasan (rhasan@uiuc.edu)

Diomidis Spinellis presented the history and the functionality of the Antikythera Mechanism, an ancient mechanical computer used for astronomy. Spinellis started with the history of the discovery of the mechanism. In 1900, Greek sponge divers on a fishing trip took shelter from a storm on the island of Antikythera, between the Peloponnese and Crete. One of the divers found some relics when he dived near the island. Subsequent archaeological expeditions found a large number of ancient artifacts, the result of a shipwreck almost 2000 years ago.

Among these was a heavily corroded bronze object consisting of multiple gears. Initially it was thought to be an astronomical toy. Later, in the 1950s, Derek J. de Solla Price, a Yale professor, presented a theory that this mechanism was used to compute the motions of celestial objects. By studying an X-ray of the object, he created a model of how it worked using a combination of many intertwined gears. Price's model was later found to be incorrect, but it formed the basis of more thorough studies published in the journal *Nature* in 2006 and 2008 (see http://www.antikythera-mechanism.gr).

A total of 82 fragments of the mechanism are available. Recently, researchers studied the mechanism using digital radiographs, X-ray tomography, and a 3D lighting model. Inscriptions on the mechanism serve as a user manual. The researchers found that it could calculate days in the Egyptian calendar. The device could also compute the motion and phases of the moon and predict solar and lunar eclipses. As a comparison, Spinellis showed that the equivalent code in BSD's moon phase program is very complex. The Antikythera Mechanism used the Metonic calendar and computed complex astronomical predictions, as well as the year of the Olympic games.

Spinellis created a complete Squeak EToys emulator for showing how the mechanism worked. The emulator is available at http://spinellis.gr/sw/ameso. He demonstrated the working of the mechanism with animations from his emulator. Finally, Spinellis pondered the purpose of the device. He said that we may never know the real purpose of the tool, but it might have had some political and strategic value, since prediction of eclipses was important at that time. This mechanism was too delicate to carry on a ship to aid in navigation. This could also have been an educational tool, or simply something someone (possibly an ancient hacker) built for fun.

A member of the audience asked what was the complexity of the mechanism compared to other machines. Spinellis said that no other such mechanism from that time has been found. Another person remarked that there was a clock built during the Renaissance to simulate the motion of the planets, and asked if this could have done the same thing. Spinellis said experts have posited that, with some gears, it could have calculated planetary positions too. The mechanism was expensive to build just for moon positions, so it probably also computed the paths of others celestial bodies. Did the device have any bugs? The device was designed in a very clever manner, and there are no bugs in the mechanism.

SYSTEM OPTIMIZATION

Summarized by Abhishek Rajimwale (abhi@cs.wisc.edu)

- ***Reducing Seek Overhead with Application-Directed Prefetching***
  Steve VanDeBogart, Christopher Frost, and Eddie Kohler, UCLA

Steve VanDeBogart addressed the problem in prefetching arising from non-sequential accesses by applications in systems using disks. He introduced a new prefetching algorithm which uses applications' knowledge of future accesses and is implemented in the form of a user-space library called "libprefetch." libprefetch provides a convenient application interface, needs little modification to the kernel, and handles resource sharing.

Steve then presented the details of the intuition behind libprefetch. He showed that for seeks above 1 MB there is a very gradual increase in cost; however, reducing seeks larger than 1 MB to less than 32 KB (on average) can result in significant performance gain. He also showed that using larger reorder buffers greatly helps to reduce average seek cost as well as number of disk passes. Next, he explained the libprefetch's interface, which uses a callback mechanism, and also talked about how libprefetch solves the problem of contention in memory by using a TCP-like mechanism (i.e., additive increase and multiplicative decrease). Finally, libprefetch showed an up to 20x improvement in some benchmarks using real applications.

Someone in the audience asked about whether applications need to use "pread" only to be able to use libprefetch. Steve clarified that they intercept read, pread, and other variants. How much gain do the authors expect if they don't use spinning disk and use SSDs (or RAID arrays) on these benchmarks with libprefetch? Disks will still exist as long as SSDs are expensive; for RAID arrays, it's possible to extend this work by pushing information up from RAID arrays about the layout; as far as direct performance gains on SSDs are concerned, there may be some gains. What is the generality of the non-linear relation between seek time and seek distance, and what are the reasons behind it? Although they had limited samples of disks in their results, similar results have been shown in previous works. The reason is due to rotational latency and seeks. Had they tried to use selective joins (queries) to see gains? As long as access patterns are known, performance will improve with libprefetch. For more information see http://libprefetch.cs.ucla.edu.

- ***Fido: Fast Inter-Virtual-Machine Communication for Enterprise Appliances***
  Anton Burtsev, University of Utah; Kiran Srinivasan, Prashanth Radhakrishnan, Lakshmi N. Bairavasundaram, Kaladhar Voruganti, and Garth R. Goodson, NetApp, Inc.

Fido is targeted to enterprise-class server appliances such as NAS, with the aim of addressing the main problem of performance in virtualizing NAS in order to exploit the natural benefits of virtualization. The main insight, Anton Burtsev said, was to use the relaxed trust model in these appliances to design fast inter-VM communication by sharing memory read-only across VMs in the same appliance.

He then detailed how their fast inter-VM communication works using a large pseudo-global virtual address space and mapping the address of all VMs in one single large address space. Transitive zero copy is achieved by mapping the communicated read-only data into this global virtual address space; a shared memory ring is used to pass pointers. Two interfaces are used to access Fido: MMNet (network device interface) and MMBlk (block device interface). Anton then presented some evaluation figures demonstrating that MMNet outperforms XenLoop and Netfront, sometimes also outperforming the monolithic kernel due to inefficiencies in the TCP stack. Further, MMBlk outperforms XenBlk and also the monolithic kernel due to contention in tmpfs and ext3. Finally, Anton presented the case study with NAS in order to give a more realistic performance evaluation. He showed that with Fido, NAS can be virtualized with little performance overhead on micro-benchmarks and TPC-C macro-benchmarks by exploiting pipelined parallelism between VMs and by eliminating copy and page-mapping overheads in the critical path.

An audience member asked what mechanism was used to reclaim memory that is shared read-only with other VMs. Anton replied that they didn't have any special mechanism to reclaim memory; for TCP they reclaim memory voluntarily when the other VM frees it, but for file systems they have to copy out memory in the other VM. He had already acknowledged this limitation in his conclusions. Someone from VMware also expressed concern about using this relaxed trust model in the face of users pushing malicious content into servers. Anton suggested that this model is a required assumption for this work and, at least for the networking stack, is a reasonable assumption. What are the benefits of virtualizing NAS if there is no fault isolation because of the relaxed trust model? VMs are a pragmatic approach with benefits of migration, cleaner hardware support, and better isolation with very little performance overhead. Future work might include implementing some micro-rebooting technique to provide fault isolation.

- ***STOW: A Spatially and Temporally Optimized Write Caching Algorithm***

  Binny S. Gill and Michael Ko, IBM Almaden Research Center; Biplob Debnath, University of Minnesota; Wendy Belluomini, IBM Almaden Research Center

Binny Gill presented a new writ-caching algorithm called STOW. He explained the need for the algorithm by showing that the destaging rate from write caches is important apart from just destaging order. He pointed out that earlier work, including his own WOW algorithm, had only focused on destaging order.

Binny presented the intuition behind the new algorithm in steps. He first pointed out problems with simplistic techniques for destaging, such as destaging as quickly as possible or having a fixed destage threshold. He suggested that destaging with linear thresholding (high and low) is required to control the destage rate, but even with linear thresholding, the destaging occurs in spikes due to the long time spent in sequential and random regions. With this, he introduced the notion of separating random and sequential data streams. However, this leads to low throughputs because mixing sequential and random streams hurts disk throughput by forcing the disk head to service two separate regions instead of one. He further explained that this can be controlled by adding hysteresis to the destages. The last important thing in the algorithm is to adapt the sizes of the sequential and random queues to be responsive to the workload. He then presented a thorough evaluation of his algorithm, comparing it with CSCAN, LRW, and WOW. STOW outperforms all the other algorithms in throughput and response time for both full back-end and partial back-end experiments. STOW gives an average of 18% improvement over the previous best algorithm (WOW), which is substantial because of the slow nature of disk I/O improvements in hardware.

Someone asked why adaptation was required with sequential queues, particularly because there is little temporal locality with sequential writes. Binny replied that this adaptation was required for multiple streams. In order to get maximum throughput, each stream must have some amount of data to say it's sequential, for at least a stripe or two stripes worth of data. So we need to adapt the size of the sequential queue according to the number of concurrent sequential streams. Why did he choose a simple hysteresis rather than some more complex mechanism? He likes to keep things simple so that they are actually used. He acknowledged that there could be some further gain in throughput (around 5% more) by using more complicated mechanisms, but he wouldn't worry about that more than the real applications of his work. Could the technique of using separate queues with hysteresis be used to dynamically adjust the sizes of read and write caches? This was an open and complex problem he hadn't dealt with.

WEB, INTERNET, DATA CENTER

Summarized by Wei Zheng (wzheng@cs.rutgers.edu)

- ***Black-Box Performance Control for High-Volume Non-Interactive Systems***

  Chunqiang Tang, IBM T.J. Watson Research Center; Sunjit Tara, IBM Software Group, Tivoli; Rong N. Chang and Chun Zhang, IBM T.J. Watson Research Center

Chunqiang Tang provided examples of systems that process requests generated by automated software tools, in addition to requests generated by interactive users, e.g., Twitter, WebCrawler, and IT monitoring and management systems. Systems that have non-interactive workloads generally benefit more from high throughput than from short response time.

Tang proposed a general black-box performance control named TCC (throughput-guided concurrency control), which varies the number of event-processing threads to maximize throughput. TCC keeps adding more threads and observes whether throughput increases. After finding peak throughput, TCC decreases the number of threads so long as throughput does not decrease significantly. Tang also described how to measure throughput accurately and efficiently through sampling and noise removal.

TCC was demonstrated to maximize throughput and control resource to near-saturate level by analyzing different event-processing queue models. The control is also evaluated in a real implementation to demonstrate the scalability of TCC and its effectiveness under various bottleneck scenarios. Tang said future work might include applying TCP-style flow control to general distributed systems. Emphasizing that performance control for non-interactive systems is an interesting problem.

Someone asked whether the time to measure throughput is deterministic. Tang said it is dynamically adjusted. Is the increase and decrease of thread number by percentage? Tang said yes. Can TCC deal with multiple pools of threads?

That is a limitation of this approach and should be studied for future work.

- ***Server Workload Analysis for Power Minimization using Consolidation***
  Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari, IBM India Research Lab

Akshat Verma described the characteristics of the workloads collected from the *Fortune* Global 500 over a period of 90 days in 2007. Based on the observation, Verma proposed two new consolidation methodologies, Correlation-Based Placement (CBP) and Peak Clustering-Based Placement (PCP).

The idea behind CBP is to separate positively correlated applications across servers. However, it cannot capture both the body and the tail of the workload distribution. PCP addresses this problem by using two parameters to decide collocation. The first can be mean or percentile for body and the second can be a tail-based metric. All correlated peaks will be separated across active servers, and the off-peak reservation can be equal to the body value. CBP and PCP are compared against peak-based and mode-based sizing approaches. The results show PCP consuming as little power as the mode-based approach while having very few capacity violations across different application suites.

Someone asked how the power is measured in this work. Verma responded that the power was calculated from a model instead of by measurement. Given that dynamic load balancing/consolidation is popular in industry, how much room is left for static consolidation? Even in that situation, PCP will help to decrease the number of migrations. Chunqiang Tang from IBM Research asked if other resources are considered. Verma said no, but believed that PCP can be extended to deal with other resources.

- ***RCB: A Simple and Practical Framework for Real-time Collaborative Browsing***
  Chuan Yue, Zi Chu, and Haining Wang, The College of William and Mary

Chuan Yue presented RCB, a pure browser-based collaborative browsing framework. A co-browsing host starts RCB-Agent, a Web browser extension. Later, co-browsing participants connect to the host using regular Web browsers. Any of them can visit a Web page and interact there, with all Web page contents automatically synchronized between the host and participants.

The authors implemented RCB as a Firefox extension and evaluated it with real-time performance in LAN and WAN settings. The results showed that a Web page can be synchronized between a host and a participant in a reasonable amount of time. RCB worked with Google Maps and Amazon Checkout. User studies indicated that most people like it and tend to continue using it.

The audience responded actively. One person wondered why RCB is better than screen sharing. Yue pointed out that RCB puts less stress on bandwidth and secure assurance. Would a password be transmitted to others? RCB can enforce a policy on the host side. If multiple participants submit changes simultaneously, what will happen to a Web page? The host can write a policy to control what action to perform if multiple changes are received. Will RCB work if one has limited access to the Internet? As long as a participant can connect to the host, RCB should work.

INVITED TALK

- ***A Computer Scientist Looks at the Energy Problem***
  Randy H. Katz, University of California, Berkeley

Summarized by Stephen P. Tarzia (starzia@northwestern.edu)

Randy Katz described how his group and others are applying computing-inspired solutions to the electrical power piece of the global energy problem. First, he described computing's role as a major energy consumer and proposed strategies increasing efficiency. Second, he addressed electrical generation and distribution. He used the Internet as a model for how the power grid might be re-engineered as a more efficient and robust distributed system.

Katz presented an overview of energy sources and sinks in the US. Currently, most electricity is generated from coal. Renewable energy sources have severe limitations, so limiting demand is important. For example, the country with the highest percentage of electricity from renewables is Denmark, with only 28%. Looking at IT energy consumption, Katz referred to the Smart 2020 report, available at http://smart2020.org. Despite the anticipated invention of new efficiency technologies, IT power drain is projected to reach 4% of total energy consumption by the year 2020.

Katz also touched on some of the same themes as the keynote, showing the energy demand breakdown in Internet data centers and describing some possible optimizations. He suggested that containerized data centers (racks of servers built into shipping containers) may be more efficient, since their internal layout and airflow can be highly optimized.

Power Usage Efficiency (PUE) optimization at the data center is already approaching optimality, so future optimizations will be within IT equipment. One of Katz's slogans is that computers must be made to "do nothing well." This goal, also called energy proportionality, means that idle computers should drain nearly zero current; this is currently not common. However, Katz showed that low-power CPUs such as Intel's Atom are more energy proportional. Since average server utilization tends to be low, one might replace each high-power CPU with several slower Atom CPUs. However, other system components such as RAM, disk, etc., currently have high idle power consumption, so optimizing the CPU is not enough.

In the second part of the talk, Katz described his plan for upgrading the power grid. He drew an extended analogy with the telephone network's evolution in the Internet era.

Because weather variations make renewable energy sources such as wind and solar both unreliable and distributed in nature, the need for a power grid upgrade is essential. Like the Internet, an effective renewable-source grid must have local buffers (energy stores) and adaptive routing. Energy storage is tricky. On large scales, one can pump water uphill or compress gasses. On the small scale, for example in homes, batteries would work but are expensive. Anticipatory work such as cooling a building earlier in the day can actually be thought of as a type of energy storage.

Katz proposed building a smart power grid by augmenting the existing grid with Internet-connected Intelligent Power Switches (IPSes). He also emphasized that an intelligent power grid with open access would bring "power to the people" in both senses of the word; it would allow enterprising individuals to contribute excess generated electricity to the grid.

Rik Farrow asked about energy storage, and Katz noted that there are many innovative options for energy storage, including using water temperature differentials. He described storage as an information management problem. Responding to a question about laptop versus desktop computer power usage, Katz noted that power management policies are currently lacking. He would like to see faster transitions from low-power to operating states and back again. Alan Thal suggested cooling data centers by building them underground. Katz responded that people are looking at innovative data center locations and that architects do have a role to play in the optimization process. Another attendee noted that room cooling is often overlooked when calculating the power drain of computers. Katz replied that PUE measurements in data centers include this and that significant energy savings can be had by allowing server rooms to reach higher temperatures (but monitoring them more carefully). As a follow-up, another attendee noted that much hardware lacks good air flow, so we may have to lean on vendors to improve this aspect of their products. Katz agreed that systems packaging is an issue and mentioned that the containerized design brings airflow to the forefront of design.

An attendee asked whether the next optimization step is choosing what we are willing to compute at all. Katz acknowledged this as the ultimate goal and a broader challenge, although it is rarely mentioned. He suggested modeling user desires and behavior and then structuring the entire system to meet those demands.

BUGS AND SOFTWARE UPDATES

Summarized by Michael von Tessin (mtvt@cse.unsw.edu.au)

- ***The Beauty and the Beast: Vulnerabilities in Red Hat's Packages***
  Stephan Neuhaus, Universita degli Studi di Trento, and Thomas Zimmermann, Microsoft Research

Stephan Neuhaus explained that they used Red Hat Security Advisories to determine which Red Hat packages had vulnerabilities reported. The distribution shows that two-thirds of all packages didn't have any vulnerabilities, whereas the kernel had the highest number of vulnerabilities. They asked if there are package properties that correlate with vulnerabilities and found that there were.

If A → B (A depends on B), then A is vulnerable if: (1) A is in an "insecure domain" ("domain" characterized by dependencies, e.g., "Internet browsers," "image manipulation programs"); (2) B is difficult to use securely (e.g., SSL); (3) a fix in B spills over to A (e.g., a change in the API).

Next they created a dependency subtree of packages, with each node having an attached risk equal to the number of packages that depend on this package. They could then find the "beauties" and the "beasts" by comparing this risk between a parent and a child in the dependency subtree. If the risk of the child is significantly higher ($p < 0.01$) than the parent's, that means that the child is the "beast," and vice versa. They then used machine learning (Support Vector Machine, SVM) to predict the future vulnerability of a package. Their model correctly predicted 10 of the 25 packages found to have vulnerabilities over the next eight months. Programmers can use this research to help choose less risky packages to depend on.

An audience member observed that they looked at binary packages instead of source packages, which could give them some strange anomalies when multiple binary packages are generated from one source package (e.g., OpenOffice). Stephen admitted that looking at source packages would make sense as well. Someone wondered whether there was a correlation vs. causation problem here; instead of implying vulnerability, might the correlation instead say something about the carefulness of the programmers in choosing which packages to use? Neuhaus responded that this was a good question, but that they tested the model on real data and were able to successfully predict the future. So there has to be some truth in it. But even if it came down to choice of packages, it would still be a useful outcome, because a programmer would now know which packages a clever programmer prefers to use. For more information see http://research.microsoft.com/projects/esm and http://www.artdecode.de.

■ ***Immediate Multi-Threaded Dynamic Software Updates Using Stack Reconstruction***
Kristis Makris and Rida A. Bazzi, Arizona State University

Kristis Makris said that software updates to patch critical security holes are arriving with increasing frequency and need to be applied as soon as possible. In a live system, downtime should be as limited as possible (milliseconds instead of minutes). This means that the system needs to be updated dynamically while it is running. In order to do DSU (Dynamic Software Update), we need to know when a system is in a state that allows updating and the mapping that maps old data structures to new data structures. But this problem is undecidable, i.e., there is no algorithm that can always find a correct solution.

Many DSU systems allow old and new code to be executed at the same time (and to use adaptors for accessing data structures), making it very difficult for users to determine valid update points. The authors' work had three design goals:

1. Atomic update: The entire state of an application has to be transferred (from old to new representation) in a single step, before which only old code was running and after which only new code is running.

2. Transaction safety: In certain cases, you need to execute a transaction (critical section of code) without allowing updates in between. This means that such code is either fully executed as old code or as new code, but not mixed.

3. Thread safety: If there are multiple threads sharing a state (e.g., in a Web server), an "immediate update" is needed. Immediate means atomic and with bounded delay (no blocking).

UpStare consists of a compiler, a patch generator, and a runtime. UpStare saves stack frames, updates global state, then reconstructs stack frames. Update points can be automatically set or set manually by the user. Thread safety is implemented by forcing all threads to block in case of an update request, safely detecting that they are blocked, and only then performing the update. Overheads during evaluation ranged from 38% to 97% with throughput decreases of none to 26%. Future work involves moving cold code to the end of image (improves cache locality), adding runtime safety checking, using semantic analysis, and updating in-transit data.

The session chair, John Dunagan, asked the only question, wondering about future work that aims to reduce the amount of user involvement. Couldn't the opposite be useful, i.e., to force programmers to annotate their programs sufficiently? Kristis said that would be a very good idea. It would help to automatically generate the patches fully, without user involvement (except for the annotations, of course).

■ ***Zephyr: Efficient Incremental Reprogramming of Sensor Nodes using Function Call Indirections and Difference Computation***
Rajesh Krishna Panta, Saurabh Bagchi, and Samuel P. Midkiff, Purdue University

Rajesh Krishna Panta explained that the goal of this work is to enable software updates on wireless sensors on the fly and "in situ." Because these devices are battery-powered, reprogramming must be fast and energy-efficient. Their approach achieves energy efficiency by sending as small a diff (delta script) as possible to the sensor, which then applies it to the current code to get the updated code. The computationally intensive part of this setup is done on the host (finding the delta script), whereas applying it is straightforward. They used rsync to calculate a delta script between the old and the new binary (on byte-level). Because of some shortcomings, they had to improve rsync quite a lot (e.g., merging superblocks).

This approach only works fine if functions in the new binary have the same addresses as in the old binary; otherwise, all jump addresses will change, making it very difficult to find a small binary diff (delta script). Solutions to this problem include: (1) leaving space after each function to avoid having to shift parts of the image if functions grow, or (2) using position-independent code, which is available only on certain architectures. Their solution uses function call indirections. All calls are performed into a fixed indirection table, with each function having its predefined slot which doesn't change on updates. Thus, only the contents of the table change, but all jump addresses in the binary stay the same. In all benchmarks, Zephyr is multiple times smaller/faster/more efficient than Deluge or Stream. Future work is to remove function-call latency (due to indirection table) by having the loader relocate the binary according to the indirection table.

Someone pointed out that this is very similar to what a dynamic linker has to do (e.g., indirection table). John Dunagan asked how often updates are typically needed in a wireless sensor environment. Rajesh didn't have numbers, but his experience told him that updates occur quite frequently. Most updates are quite small, e.g., because the environment changes and sensors might have to be reprogrammed to behave a little bit differently. Very small bug fixes or very large updates are rare.

CLOSING SESSION

■ ***Third Millennium Problem-Solving: Can New Visualization and Collaboration Tools Make a Difference?***
David Brin, Hugo Award-winning author

Summarized by Rik Farrow (rik@usenix.org)

David Brin introduced himself as an astrophysicist by training who is also a book author and futurist. Brin headed off into his title theme, but quickly took off in several intriguing directions.

Brin explained that the horns depicted on Moses' head in Renaissance paintings weren't really horns but "lamps on his brow." These lamps are, in turn, a metaphor for the frontal lobes of the human brain that allow us to plan for the future and "discover the troubles in front of you before you stumble into the pit." As a futurist, I have no doubt that Brin uses his horns a lot.

Brin, like other futurists, is very interested in the singularity, the point when humans have computer-enhanced intelligence, or strong AI exists. Brin believes that the singularity is approaching within the current generation, due to the acceleration in technological and social advances that started in the 15th century with the development of printing presses and glass lenses. Printing presses democratized knowledge, while glass lenses made it possible to study the solar system—incidentally uncovering the fact that Earth is not the center of the universe.

The 18th century brought with it mass literacy, printed illustrations, and science, or Brin's memory, vision, and attention. The 19th-century version of these three themes were mass education and public libraries, photography and cinema, and global communication. In the 20th century, we got computers and databases, television and mass media, and abstraction and immersion. By sometime in the 21st, we will have a knowledge mesh, omniveillance (stick-on cameras with IPv6 and one-year batteries) and super immersion. The acceleration of technology, including Moore's Law, will bring about the merger and/or replacement of humans with post-humans and/or AI.

Brin told us that Internet millionaires, like his distant cousin Sergey Brin (Google), believe in positive sum games. The world of the future should not rely on scarcity for worth but be a world where everyone gains.

Brin spoke on many other topics, one of the strongest being a plea for CERTs: Community Emergency Response Teams. Brin pointed out that the many of the most effective responders during 911 were members of the local community, and that we need to support training for CERT members as well as develop P2P communication that will stand up during emergencies such as Katrina.

Eventually, Brin slowed down and opened the floor to questions. Matt Blaze strode to the mike and picked out just one of the many controversial points Brin had made, that no online argument has ever been settled. Matt said that he can count "zillions of times I've been personally informed by an online discussion that I never participated in that prevented me from spreading wrong information." Hey, me too, Matt. Brin feinted by suggesting that we should turn portions of the Internet into arenas for ideas with rankings by reputation for the posters. Blaze countered by suggesting that the Internet may have evolved a generation with better bullshit detectors. Brin agreed, saying that he still wanted better tools for discourse.

Stephan Neuhaus disagreed with Brin's point that graduate school has forced many people into very narrow and focused interests and that this was actually harmful. Neuhaus contended that poor countries really needed to build a professional class. Brin said that he thinks the Third World will quickly pass through their own over-professionalization curve.

You can learn more about David Brin and his thoughts on his Web site: http://davidbrin.com/.