conference reports

THANKS TO OUR SUMMARIZERS

2010 USENIX FEDERATED CONFERENCES WEEK

2010 USENIX Annual Technical Conference 62

Italo Dacosta
Paul Marinescu
Aleatha Parker-Wood
Dan Schatzberg
Xiao Zhang
Rik Farrow
John McCullough
Joshua Reich
Marc Staveley

USENIX Conference on Web Application Development (WebApps '10) 77

Pradeep Chalise
Rik Farrow
Pradeep Teregowda
Aiman Erbad
Thomas Moyer

3rd Workshop on Online Social Networks (WOSN 2010) 84

Saptarshi Ghosh
Christo Wilson

2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10). 91

Alva L. Couch
Joshua Reich
Rik Farrow
Malte Schwarzkopf

2nd Workshop on Hot Topics in Storage and File Systems (HotStorage '10) 100

Rik Farrow
Aleatha Parker-Wood

Configuration Management Summit . . 104

Aleksey Tsalolikhin

2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10) 106

Romain Cledat
Chris Gregg
Rik Farrow
James C. Jenista

2010 USENIX FEDERATED CONFERENCES WEEK

June 22–25, 2010
Boston, MA

This year, USENIX combined established conferences and new workshops into a week full of research, trends, and community interaction, with a completely customizable program. For more information about the events and the format, see http://www.usenix.org/events/confweek10/.

2010 USENIX Annual Technical Conference

June 23–25, 2010
Boston, MA

WELCOME, AWARDS, AND KEYNOTE ADDRESS: JOINT SESSION OF 2010 USENIX ANNUAL TECHNICAL CONFERENCE AND USENIX CONFERENCE ON WEB APPLICATION DEVELOPMENT

Summarized by Rik Farrow (rik@usenix.org)

Timothy Roscoe, program co-chair with Paul Barham of Annual Tech, said that 147 papers were submitted, slightly fewer than the previous year, due to competition from other conferences; after a thorough review process, 24 papers were accepted. Roscoe presented awards and checks for the two Best Papers: "LiteGreen: Saving Energy in Networked Desktops Using Virtualization," with Pradeep Padala of DOCOMO USA Labs accepting the award, and "ZooKeeper: Wait-free Coordination for Internet-scale Systems," with Benjamin Reed of Yahoo! Research accepting.

John Ousterhout, chair of WebApps, took over the podium. Ousterhout said that the size of the conference was a good beginning, with 80 attendees, 26 papers submitted, and 14 accepted. Ousterhout said that there have been three phases of the Web: the first, distributing documents; the second, as a platform for delivering apps; and phase three, the current one, which will see a complete turnover in the application development food chain. Ousterhout announced the Best Paper award, "Separating Web Applications from User Data Storage with BSTORE," by Ramesh Chandra, Priya Gupta, and Nickolai Zeldovich.

Clem Cole, President of the USENIX Board, took the stage to hand out two more awards. The USENIX Lifetime Achievement Award, a.k.a. "The Flame," went to Ward Cunningham, the inventor of the Wiki. The STUG award, which recognizes significant contributions to the community that reflect the spirit and character demonstrated by those who came together in the Software Tools User Group, went to the group who created MediaWiki, whose work includes a tool many of us use every day—Wikipedia. The award money was donated to the Wikimedia Foundation.

KEYNOTE ADDRESS

- ***Lessons of Scale at Facebook***
  Keynote by Bobby Johnson, Director of Engineering, Facebook, Inc.

  Summarized by Xiao Zhang (xiao@cs.rochester.edu)

Bobby Johnson explained how they address the technical challenges as the number of Facebook users grows explosively. In particular, he elaborated on three key perspectives: moving fast, server scaling, and client performance.

To be able to move fast, Facebook has a culture of making frequent small changes. Johnson commented that it was really easy to figure out what went wrong in production if you only changed one thing at a time and watched it closely over time.

The server infrastructure of Facebook is divided into Web Server, Memcache, and Database. Most of the scalability work falls into the Memcache layer, because it has to serve hundreds og millions of objects in a second. Johnson gave an example on how to dynamically scale the number of Memcache machines communicating with the switch to avoid packet dropping due to overload. He also pointed out most machine failures were due to software, and many failed machines run the same piece of buggy code. He told an anecdote of twenty machines leaking memory at the same rate.

Johnson introduced two projects to improve client performance. The first is called Big Pipe, which splits objects in a page and runs them in pipelines. By doing so, it allows priority content to be shown quickly and also benefits from parallelism. The second is a small JavaScript library core called PRIMER, which does bare-minimum things to make a page feel interactive during loading. Big Pipe and PRIMER share the property of dividing things into a fast path and a slow path.

In closing, Johnson talked about engineering culture at Facebook. One particular principle is that control and responsibility have to go together.

Marvin Theimer asked if all data had to stay in memory. Johnson replied that the social graph data is entirely indexed in memory, while pictures and videos are stored in disk. He also mentioned increasing interest in flash storage. Bill LeFebvre inquired about the problem of constantly increasing storage demand. Johnson said that Facebook does not plan to delete data and that the current solution is to buy lots of cheap hard drives. John Ousterhout asked whether PHP is the right language for Facebook. Johnson agreed that PHP is not a great language for running Web applications, although it is a fantastic language for writing them. And that's partially why Facebook has a compiler project to transfer PHP to C++ code. Johnson also emphasized that an interpreter language is critical for Facebook building things quickly. Ben Johnson asked about data consistency, and Johnson replied that Facebook cares about consistency and puts a lot of work there.

JUNE 23, 10:30 A.M.–NOON

Summarized by Joshua Reich (reich@cs.columbia.edu)

- ***DEFCon: High-Performance Event Processing with Information Security***
  Matteo Migliavacca and Ioannis Papagiannis, Imperial College London; David M. Eyers, University of Cambridge; Brian Shand, CBCU, Eastern Cancer Registry, National Health Service UK; Jean Bacon, Computer Laboratory, University of Cambridge; Peter Pietzuch, Imperial College London

Matteo Migliavacca presented the problem: event-stream processing needs strong security—this is of particular application in financial contexts. If flows are incorrect, this can lead to security violations (e.g., companies may see each other's trade data). Consequently, the authors propose tracking and controlling data flows. Their primary contribution is a decentralized event flow control, DEFCon, implemented in Java, where all data is tagged. For data tagged with an access security tag, one either needs to have access granted or the data needs to be declassified in order to be read.

Preventing nodes from peeking at data is actually rather tricky in practice, as there are many opportunities for information leakage (e.g., returning "access denied" provides information, and failure to respond may also do so). The DEFCon approach assumes that all units communicate through labeled events. This could be done using VM or OS-level mechanisms, but they would prove too heavy for low-latency environments. Instead, the authors use threads that share data in a single address space. They wrote an implementation using Java threads, but need to have them share immutable data objects, and thus some engineering design is called for. The authors show that with the right set of techniques the overhead can be made reasonably small.

Why use only one VM? For performance. Why not use features Java already has to divide flows? Currently existing features don't focus on label checking performance. Additionally, these approaches are generic, and they want to be as efficient as possible for their domain.

- ***Wide-Area Route Control for Distributed Services***
  Vytautas Valancius and Nick Feamster, Georgia Institute of Technology; Jennifer Rexford, Princeton University; Akihiro Nakao, The University of Tokyo

Currently, all traffic from a given data center uses only one path to the user, Vytautas (Valas) Valancius began. Yet different cloud apps have different requirements. Interactive applications need low latency and low jitter, while bulk-data applications need high throughput at low cost. Amazon EC2 has 58+ routing peers but picks only one route per user!

Today, if one does want to route flexibly one needs to obtain dedicated connectivity and numbered Internet resources, which are both difficult and expensive to set up. The authors proposed essentially building a BGP-level NAT Transit Portal. Each service has a virtual router through which all traffic flows. This virtual router essentially uncovers the

Transit Portal's info, allowing the virtual router to decide which path it would like to use for a given traffic flow (at least first hop).

In this setup each service has its own router (virtual or physical). Each router has a link to the Transit Portal, which emulates a connection to an upstream ISP (e.g., three links to a Transit Portal for three peered ISPs). This exposes a standard BGP router control interface. The authors have found it takes about 30 seconds to converge when a service router changes a path. Their system is currently deployed in academic settings, built on top of a regular router running custom software at three active sites.

Active experiments include BGP poisoning, IP anycast, and advanced networking class—students can use BGP.

The authors are also exploring advanced Transit Portal applications such as fast DNS and service migration (currently only available to large operators that have their own global network backbones).

The final challenge addressed by Valas was scaling. Here the Transit Portal needs to scale to dozens of sessions to ISPs and hundreds of hosted sessions, but standard BGP only chooses one peer to send to. Consequently, the authors have implemented separate routing tables for each peered ISP. They use virtual routing tables to shrink from 90 to 60 MB per ISP and schedule/send routing updates in bundles to reduce CPU usage.

Future work includes more deployment sites, making it accessible to testbeds (e.g., GENI), faster forwarding NetFPGA, OpenFlow, and a user-friendly interface for route control (running BGP is heavyweight right now).

Someone wondered whether this could have applications beyond the cloud. Valas responded that it is indeed more general. Have they considered abuse, security risks? Good question. These things have been seen in the wild (e.g., the YouTube Pakistan problem). They currently advocate that administrators regulate which paths users should be allowed to announce. How do they manage to negotiate between users and the ISP? By making the market more competitive and letting economic incentives prevail.

- ***LiteGreen: Saving Energy in Networked Desktops Using Virtualization***
  Tathagata Das, Microsoft Research India; Pradeep Padala, DOCOMO USA Labs; Venkat Padmanabhan and Ram Ramjee, Microsoft Research India; Kang G. Shin, The University of Michigan

  Won Best Paper Award!

Pradeep Padala began by saying that PCs waste much energy while idling but users do not like disruption. Also, manual methods for waking machines for remote access are cumbersome and, thus, automated energy saving methods are needed. Padala noted that much energy waste occurs during idle periods of less than three hours and this is the energy they focus on saving (their approach does save power for longer idle periods as well). The LiteGreen architecture calls for users to always run their OS inside a VM. This VM runs inside a hypervisor/VMM either locally (when the user is physically present or significant computation needs be done) or remotely (when the machine would have been idling). LiteGreen maintains instant availability and masks migration effects by using a combination of indirection (even when the VM is local users, log in through Remote Desktop) and live migration of VMs between the local machine and the remote LiteGreen server.

This setup requires that the user's PC, the LiteGreen server, and network storage server (data is no longer stored on local hard-drives) all be attached to a gigabit switch (the network storage could run on a separate backbone, of course). There is a several-second delay while live migration occurs and the Remote Desktop session transfers from remote to local VM instances (or vice versa).

The authors explore how idle should be defined/when VMs should be migrated, coming up with heuristics involving user activity and resource usage (both on the local machine and on the LiteGreen server). Finding good heuristics for this problem is still very much an open question. With their current methods, the authors found that on some machines very little energy could be saved, but for machines that slept soundly overnight, savings were quite significant. This does prove a bit problematic vis-à-vis the authors' goal of saving power on <3-hour idle periods.

Their prototype was built on top of Hyper-V and Xen. They found they could shrink VMs 8x by using just the working set. Moreover, they may get even larger consolidation ratios if the overlap between VM working sets is significant. For now they claim that 80 or so VMs could be supported by a single LiteGreen server.

How did they support the large amount of storage needed for all of their VMs? They only need to store the main OS image on the server and can use snapshots to reduce VM images even further. But what about user data? They use shared storage—e.g., NAS, SAN. It seems as if they've taken VDI and made it a harder problem—why not go for thin client, since they are running RDP anyway? This is different from thin clients. You need to have lots of servers for peak usage, but here they only keep idle VMs. What about scalability? They can support 100 users per machine. But don't idle Windows VMs use a lot of resource consumption compared to the VMs they've implemented on? With work they can get similar numbers. Do power savings also include server consumption? Yes. The server takes 250W, more or less static. Current servers aren't energy proportional.

JUNE 23, 1:00 P.M.–2:00 P.M.: INVITED TALK

- ***Visualizing Data***
  Ben Fry, Author and Consultant

  Summarized by Marc Staveley (marc@staveley.com)

Ben Fry talked about his work in providing ways to understand through various methods of visualization the mountains of data that are being produced today. In his words, "Given a pile of data, how can we very quickly visualize it and mine through it to ask interesting questions?"

Fry is a cross-discipline practitioner combining graphic arts and computer science. He is the author (along with Casey Reas) of Processing, which is an open source programming language and environment for images, animation, and interactions. With Processing it is possible to quickly and easily generate an interactive image of "information that dances before your eyes."

Fry showed a number of examples to illustrate the power of Processing. One was a graphic of Fortune 500 companies over time that allowed the user to see the rise and fall of different companies and market segments by just mousing over their names. Another was a DNA browser that allowed the user to "look at the forest and the trees at the same time" by providing the user a way to expand segments on a DNA strand while still seeing the full chromosome for context. Fry also showed work where data was used to just provide a pretty picture that could, for example, be used as a magazine cover or illustration. One example was DNA strands spelled out on many planes.

A vibrant community has built up around Processing, with a user base that has grown to over 25,000 active members. Fry, of course, had a graphic that showed the activity of the user base over time. Processing, which is written in Java (so it works on Windows, Mac, and Linux), is an interpreted interactive visualization language that hides the complexity of graphic generation, while still providing a powerful set of primitives.

The community has contributed a large number of different libraries to the project to extend the power of Processing. There is even a port to JavaScript (processing.js) which allows Processing datasets to be visualized entirely in a Web browser.

To learn more about Processing, you can pick one of the available books, including Fry's *Getting Started with Processing,* which just came out for the nontechnical market. Or go to processing.org to read the wiki and download the environment.

An audience member noted that Processing and Apple Quartz Composer are similar. Fry replied that Quartz is all GUI programming–based (i.e., drag and drop boxes), while Processing is text programming–based, which he believes is more powerful for doing things the original designer didn't think of.

JUNE 23, 2:00 P.M.–3:00 P.M.

Summarized by Marc Staveley (marc@staveley.com)

- ***Stout: An Adaptive Interface to Scalable Cloud Storage***
  John C. McCullough, University of California, San Diego; John Dunagan and Alec Wolman, Microsoft Research, Redmond; Alex C. Snoeren, University of California, San Diego

John McCullough observed that there is a need to improve the performance of application server access to the storage tier in multi-tier Web architectures, especially when those applications are hosted in cloud environments where access to the storage tier may have competition from other users of the cloud.

When the storage tier is under high load, it is possible to achieve this improvement by batching storage requests from middle tier applications, thereby amortizing overhead costs over a number of storage requests. But there is a throughput vs. latency tension; when load is low on the storage tier, you want latency to dominate by batching only a small number of requests (or not batching at all), but when load is high, batching aggressively will increase aggregate throughput.

Stout is a storage interposition library that uses an adaptive algorithm to choose the batch size based on the current load on the storage tier. The algorithm runs independently in each middle tier application but adapts to give each application a fair share of the storage bandwidth. It does this by using the latency history of recent storage requests to adjust the batch size (similar to recent work in TCP congestion control).

Would using Stout on some of the middle-tier servers but not others still achieve fair sharing? The clients not using Stout would not achieve fair share, while those that do would still be able to improve their overall performance.

- ***IsoStack—Highly Efficient Network Processing on Dedicated Cores***
  Leah Shalev, Julian Satran, Eran Borovik, and Muli Ben-Yehuda, IBM Research—Haifa

Leah Shalev observed that TCP/IP is a major consumer of CPU cycles but wastes lots of those cycles on multi-processor machines with cross-calls and cache line misses (stalls). She claims that TCP/IP uses tens of thousands of CPU cycles for just hundreds of "useful" instructions per packet.

The problem with running the TCP/IP stack on a multiprocessor (including multicore) system is that using a single lock produces high contention, while using finer-grained locking has higher overhead and causes many cross-calls and cache line misses. She noted that using CPU affinity to keep the application on the same CPU as the TCP/IP stack for that application doesn't work in practice with multithreaded applications using multiple cores.

IsoStack runs as a single kernel thread isolating the network stack to a single dedicated CPU with a lightweight interconnect API between the rest of the kernel and the network

stack on the single CPU. They produced the interconnect by splitting the socket layer with the front end in the application and the back end in IsoStack. They have achieved near line speed (10GiB/s) with a 25% CPU utilization on an IBM Power6 with eight cores.

An audience member asked whether this is a scalable solution as networks get faster but single cores do not. Is there a future bottleneck looming for IsoStack? Shalev replied that Receive-Side Scaling (explained in the paper) can be used to scale IsoStack to use multiple cores without the overhead of introducing any locks.

JUNE 23, 3:30 P.M.–5:30 P.M.

Summarized by Aleatha Parker-Wood (aleatha@soe.ucsc.edu)

- ***A Realistic Evaluation of Memory Hardware Errors and Software System Susceptibility***
  Xin Li, Michael C. Huang, and Kai Shen, University of Rochester; Lingkun Chu, Ask.com

Xin Li presented a survey of memory hardware errors, focusing on non-transient errors. The data was collected from 212 servers at Ask.com, with over 800GB of memory, monitored for nine months. In addition, they looked at data from PlanetLab machines, and 20 desktops from the University of Rochester. These results have been previously reported in USENIX '07.

One purpose of this work is to evaluate the efficacy of countermeasures such as ECC and Chipkill. These countermeasures are often expensive to add to a chip, and so the authors wanted to examine how often these problems occurred, as well as whether countermeasures were effective when applied. Since memory errors are rare, the authors used Monte Carlo simulation in order to step up error rates and evaluate the impact on software, with and without each of the countermeasures applied.

The authors were particularly interested in the effect on software running on faulty memory, since not all errors are accessed. In order to evaluate the effect, they injected faults into a virtual machine. To track memory accesses, they used a novel memory-tracking approach which relies on page access controls for coarse-grained tracking and then uses hardware watch points for faults within the page. They concluded that without error correction, 50% of non-transient errors cause errors in software, in the form of wrong output, software crash, or a kernel crash. When ECC is applied, the frequency is reduced, but some errors still creep through and are just as severe.

Mohit Saxena from the University of Wisconsin—Madison asked how the approach compared to 2-bit ECC and cache errors. Li said he was unfamiliar with the approach, but believed it was a weaker model than the Chipkill ECC. If it was widely available, he would look into its effect.

- ***The Utility Coprocessor: Massively Parallel Computation from the Coffee Shop***
  John R. Douceur, Jeremy Elson, Jon Howell, and Jacob R. Lorch, Microsoft Research

Jeremy Elson presented a utility-computing framework specifically designed for desktop applications working in high latency, low bandwidth applications for limited periods of time. A framework like this would allow highly parallelizable applications, such as software development, video editing, 3-D modeling, and strategy games to take full advantage of the computational power of the cloud. However, users and application designers are unlikely to want to install a new operating system or write highly specialized code to take advantage of this computing power. And users have highly heterogeneous systems, with different software and libraries, which the system should take advantage of.

To achieve a system with a low barrier to entry, they rejected manual replication of code, and software as a service. Instead, they suggest a remote file system, which requests files as needed from the client file system. To keep this from being prohibitively slow, they use a variety of techniques. First, they carefully relax consistency semantics, using task-open to task-close rather than file-open to file-close. This reduces the amount of data transferred. Second, they use a content-addressable storage (CAS) model to ensure not only that data can be re-used between runs of the software, but that users using the same libraries or software can leverage data from one another. On the first run, the parameters are sent to a distributor, and from there to worker processes. Workers request the files they need, such as libraries and binaries. Writers write to a temporary area and, on completion, the results are returned to the client. Subsequently, remote file hashes are checked against the local files to ensure that files are up to date, and differential compression is used to send changes.

Since all of the libraries and software are pulled from the client, there are no major OS compatibility issues or any need to manually update libraries on the cluster. One cluster can be shared across a variety of users and applications. The authors note that the only downside is a lack of shared memory. All IPC must be done through the file system.

Someone asked about what was required to persist between invocations, whether a file system was needed or whether computer time would need to be rented. Elson replied that all that was needed was a file system. Further, since the system used content-addressable storage, the file might already be cached from a different user. What about licensing issues, since the net result might be thousands of copies of Photoshop running on the cluster? A good point, but not one that Elson felt qualified to address. What about privacy issues? Cache sharing was not a vulnerability, since if you can name a file, you must already have a copy of it. Another audience member noted that the current model for cloud computing is to pre-allocate virtual machines, which are then billed by start-up cost. Did Elson think the charge

model would change? Elson said that for the time being, it was best to assume that one client would be the only one using it. This would still be economical for many tasks, especially extended ones. However, he predicted that the cost model might change if there were enough people to amortize the cost of these clusters. Finally, an audience member noted that the target applications might benefit from using GPUs, or computing resources on a remote desktop. Elson replied that making things faster locally was always superior, but that it wasn't always practical to take a spare GPU to a coffee shop.

- ***Apiary: Easy-to-Use Desktop Application Fault Containment on Commodity Operating Systems***
  Shaya Potter and Jason Nieh, Columbia University

Shaya Potter presented Apiary, a framework for fault containment. Desktop applications are a common vector for exploitation. However, many of these applications have no reason to persist data or to interact with one another. One possibility would be to use an isolated VM for each application or instance of an application, in order to keep the impact of exploited applications to a minimum. But this represents a significant amount of overhead, both for the system and for the user. Instead, the authors propose a slightly smaller virtual system, known as a container. Containers can contain one or more applications and can either persist data between invocations, in an isolated file system, or be ephemeral. They retain the look and feel of the desktop. They are low overhead and quick to instantiate. They offer a lower degree of isolation than a full hardware VM, but are sufficient for most applications. If applications need to invoke external applications, such as a browser invoking a PDF viewer, an ephemeral container can be invoked for the duration of that session.

The system uses unioning file system concepts to manage packages. They introduced a new file system, known as the Virtual Layered File System (VLFS). VLFS turns packages into read-only shared layers. This allows different applications to depend on different versions of packages. Since layers are shared, a file system image for an ephemeral application can be created instantly by dynamically composing layers. Any file system changes are updated to the private layer, which isolates changes and makes malicious file system changes visible. The authors presented their system in a variety of case studies, and concluded that it introduces approximately a 10% overhead for 25 parallel instances running a suite of applications.

Catherine Zhang from IBM asked what would need to be changed to migrate to this system. Potter replied that you'd need to replace all of the packages with layers. The authors have a tool which converts packages into layers, but it's not very robust yet. John McCullough from UC San Diego asked how important it was to have the different layers for applications, and whether that was just to support conflicting versions. Potter replied that it also supports granularity. For instance, if a security hole is found in a library such as libc, it is better to be able to simply upgrade a single layer. Someone asked what happens when you don't want ephemeral behaviors, such as when a document is downloaded from the Web. Potter replied that files that are changed in an ephemeral process are persisted to the file system, but the container itself is deleted after use.

- ***Tolerating Malicious Device Drivers in Linux***
  Silas Boyd-Wickizer and Nickolai Zeldovich, MIT CSAIL

Silas Boyd-Wickizer presented SUD, a confinement system for Linux device drivers. SUD (not an acronym) is designed to convert existing kernel-space device drivers into drivers that can be run in user space. One of the major obstacles to this goal is the lack of modularity in the current driver interfaces. The kernel runtime cannot currently be used for drivers in a different protection domain.

To achieve their goal of user-space drivers with a minimum of rewriting, they emulate the kernel environment in user space using SUD User-Mode Linux (UML), which can be used to shadow necessary variables. In addition, they add proxy drivers to the Linux kernel, which allows reuse of the existing driver APIs. Proxy drivers and SUD-UML converts the existing Linux driver APIs into RPCs. The proxy driver is responsible for synchronizing shadowed variables before and after RPCs. Non-preemptable functions are implemented in the proxy driver to prevent the user-space driver from being preempted. SUD adds a hardware access module to the kernel to prevent drivers from doing real physical accesses which could be used to attack the system directly via hardware. By using I/O virtualization, the driver can be given direct device access while preventing attacks. This is implemented using the IOMMU capability of modern systems.

Wenji Wu from FermiLab asked how many times SUD copied from user space to the kernel, for instance, in the given example of packet transmission. Boyd-Wickizer replied that shared buffers in the user-kernel shared memory remove any actual copy operations in that example. How does the driver write to the actual registers for the hardware from user space? The memory is mapped using mmap. How is control passed from the proxy driver to the user-space driver, and does that need to be privileged? Silas replied that it did not need to be privileged. Xin Li from the University of Rochester asked how often device drivers were actually malicious versus simply a source of bugs. In general, drivers were not written to be malicious, but due to exploitation could become malicious over time. Li followed up, saying that this implied that the interface between the user level and the kernel level is fragile and that pushing the device driver outside the kernel wouldn't improve the situation. Boyd-Wickizer replied that this sort of isolation made it easier to restart the driver and keep it from crashing the kernel. An audience member noted that because the user-space drivers had a flag set to keep them from being swapped out, this would result in partitions in physical memory, which might make it hard to allocate large

contiguous buffers. Boyd-Wickizer replied that since most of the DMA buffers were a few megabytes or smaller, this wasn't a major concern.

JUNE 24, 9:00 A.M.–10:00 A.M.: KEYNOTE ADDRESS

- ***Some Thoughts About Concurrency***
  Ivan Sutherland, Visiting Scientist at Portland State University

  Summarized by Dan Schatzberg (schatzberg.dan@gmail.com)

Ivan Sutherland opened the second day of the conference by discussing his design for an asynchronous computer. He has created the Asynchronous Research Center at Portland State University to work on this design, which he believes is achievable if we change two paradigms.

The first paradigm is that of sequential computing. When Maurice Wilkes ran the first program on EDSAC on May 6, 1949, the nature of the computing was a sequential order of operations. The cost of logic operations was much greater than the costs of communication between the operators. So it made sense to focus on the sequence of logic operations. But now the majority of the cost in the system is in communication. We currently don't have a vocabulary to configure communication. The details are hidden from the software. Sutherland then described his design, called Fleet. Fleet is a system designed to have configurable communication between the functional units. Programming is done by describing where data is sent to or from a functional unit. Sutherland claims that because the default is concurrent execution, programming the machine for concurrency is simpler.

The other paradigm is the use of a clock. It is not a necessary for a machine to have a clock. At one time it was useful for dealing with electrical noise, but now it creates power supply spikes. Because Fleet is designed so that functional units run when they have input to do so, there is no need for a clock tick for each execution. Everything runs concurrently (not just across "cores" but across all functional units).

Sutherland concluded his talk by saying that the system was still in its infancy. There is still much to do to make such a system really useful.

JUNE 24, 10:30 A.M.–NOON

Summarized by Dan Schatzberg (schatzberg.dan@gmail.com)

- ***Proxychain: Developing a Robust and Efficient Authentication Infrastructure for Carrier-Scale VoIP Networks***
  Italo Dacosta and Patrick Traynor, Converging Infrastructure Security (CISEC) Laboratory, Georgia Tech Information Security Center (GTISC), Georgia Institute of Technology

Italo Dacosta presented work done with Patrick Traynor on efficient large-scale authentication. He began by talking about the trade-offs among performance, scalability, and security. Some robust but computationally expensive security mechanisms are difficult to deploy in production environments, while others are more efficient but weaker and can be broken or abused. Session Initiation Protocol (SIP) is used for establishing, managing, and terminating sessions between at least two clients. It is generally associated with VoIP. Typically, only Digest authentication is used, because it is more efficient even though it is weak.

SIP Digest Authentication is a challenge-response protocol that uses cryptographic hash operations. The authentication works as follows: A user sends an invite request to a nearby proxy server. The proxy server asks the user for a hash of some secret stored on the database. The user responds and the proxy queries the database for the hash value to confirm that it matches, then sends the invite. The issue is that each time a user sends an invite, the database server must process a request and send it to the proxy. In testing, with no authentication their scenario could handle 24,000 calls per second. With authentication, they were brought down to just 1,160 calls per second.

The proposed solution is to cache temporary credentials created from hash chains to reduce the number of requests to the database. A hash chain is a sequence of one-time authentication tokens created by applying a hash function to a secret value multiple times. The server can cache the *n*th value in the chain. Then when the user sends an invite, the server can authenticate it by asking for the (*n*-1)th value in the chain, hashing it, and confirming it matches the original value. Then, on the next invite from the user, the server can ask for the (*n*-2)th value and so on. This only requires one database request initially and then none afterwards. The modifications required to implement this were relatively small and the cached credentials are only 134 bytes each. With Proxychain they were able to achieve 19,700 calls per second. Italo Dacosta can be reached at idacosta@gatech.edu.

- ***ZooKeeper: Wait-free Coordination for Internet-scale Systems***
  Patrick Hunt and Mahadev Konar, Yahoo! Grid; Flavio P. Junqueira and Benjamin Reed, Yahoo! Research

  Won Best Paper Award!

Benjamin Reed presented his work on a system for Yahoo! applications. The challenges involve lots of servers, users, and data. Requiring fault tolerance in such a system makes designing applications difficult. Reed discussed various distributed system architectures, some involving a master-slave relationship and others being fully distributed with a coordination service. Their system had a few requirements, including wait-free (slow processes cannot slow down fast ones), linearizable writes, serializable reads, client FIFO ordering, and client notification of a change before the change takes effect.

They designed a system with a very simple API that included only about 10 primitive instructions. A hierarchi-

cal namespace is designed where each node has data and children. Workers can get configuration when brought up and set a flag to be notified of a change. Administrators can change the configuration and then the workers receive the updated settings. Benjamin Reed showed how the API can be used to do leader election as well as locking.

The ZooKeeper Service is designed to have many servers with a copy of the state in memory. A leader is elected, the followers serve the client, and updates go through the leader. $2f+1$ machines tolerate f failures. The service is open sourced at http://hadoop.apache.org/zookeeper.

- ***Testing Closed-Source Binary Device Drivers with DDT***
  Volodymyr Kuznetsov, Vitaly Chipounov, and George Candea, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Vitaly Chipounov presented work on debugging device drivers. Testing device drivers is difficult for many reasons. Using sample input, it is difficult to cover corner cases. Exhaustive path exploration is also inadequate, because drivers run in an environment, so symbolic analysis alone is not effective. Modeling the environment completely is difficult, too. Dynamically testing requires HW and cannot do multipath execution. Static testing requires the source code of the driver and a modified environment.

Chipounov proposes DDT as a solution. It executes the driver symbolically within a virtualized machine. The machine outputs symbolic values for each hardware request. The driver is then symbolically executed, with the system state forked on conditionals. If bugs are found, constraints are solved. Interrupts are also symbolic. It's not possible to call the kernel symbolically, so each call is returned with a random value that satisfies the constraints. With kernel API annotations, coverage can be increased.

With this exception, an OS-level checker can be run on multiple paths, and a VM level checker can run outside the machine. Detailed reports are output about bugs. Chipounov concluded with a demo for a reproducible blue screen on Windows XP SP 2 based on a bug in a Microsoft-certified closed-source driver.

JUNE 24, 1:00 P.M.–3:00 P.M.

Summarized by Marc Staveley (marc@staveley.com)

- ***A Transparently-Scalable Metadata Service for the Ursa Minor Storage System***
  Shafeeq Sinnamohideen and Raja R. Sambasivan, Carnegie Mellon University; James Hendricks, Carnegie Mellon University and Google; Likun Liu, Tsinghua University; Gregory R. Ganger, Carnegie Mellon University

Shafeeq Sinnamohideen gave a brief description of the Ursa Minor Storage System, a storage system designed to scale to thousands of storage nodes. Ursa Minor is split into data storage nodes (storing bulk file data) and metadata storage nodes (storing file attributes including the location of bulk file data on data storage nodes).

Ursa Minor needed a scalable metadata store that is consistent across all metadata servers. Since some operations can affect two objects (e.g., object rename and object create) whose metadata may be on two different servers, a mechanism was needed to maintain consistency across metadata server boundaries.

Sinnamohideen's team decided not to use a distributed transaction protocol (like Farsite) or a shared state with distributed locking protocol (like GPFS), since these seemed to be overly complex systems to handle an infrequent event. Instead, they decided to migrate all of the metadata objects needed for the operation to a single metadata server before applying the metadata change.

The authors noted that multi-object operations usually operate on objects that are close in the file-system hierarchy. So they decided to organize the store so that objects that are close in the filesystem hierarchy are handled by the same metadata server and therefore do not require object migration to have operations applied to them. Sinnamohideen showed that the latency added by this model does not adversely affect the overall system, since multi-object operations are so very rare. The team then measured the performance of the store with a modified version of SPECsfs97 (with multi-sever OPS at 100 times the observed usage) and showed that the system scales linearly with added metadata servers.

It was noted during the question period that this only works if metadata migration can happen quickly. In Ursa Minor the metadata is actually stored on the data storage nodes, so migrating metadata doesn't require actually moving the data, but just changing which metadata server is responsible for it.

- ***FlashVM: Virtual Memory Management on Flash***
  Mohit Saxena and Michael M. Swift, University of Wisconsin—Madison

Mohit Saxena noted that application memory footprints are ever-increasing but we don't always have the ability to just add more DRAM (e.g., power and DIMM slots limitations). Saxena presented a virtual memory subsystem for the Linux kernel which uses Flash memory as the backing store. He showed how they had modified the current VM subsystem to remove the disk optimizations, which are not needed for a Flash backing store. Saxena then went on to show how they handled the characteristics of Flash—for example, the need to erase pages before writing to them (it was noted that the SSD discard command is very slow, so FlashVM coalesces discards to amortize the cost of the command).

Their performance evaluation showed up to a 94% performance increase when there is pressure on the virtual memory subsystem. But Saxena believes that there is more work to be done in avoiding expensive discard operations.

Does their architecture interfere with the wear leveling that is being done by the SSD? Saxena did not believe so, since they are not doing any leveling themselves, but they do ag-

gressively reduce the number of writes to the device so as to extend its life.

- ***Dyson: An Architecture for Extensible Wireless LANs***
  Rohan Murty, Harvard University; Jitendra Padhye and Alec Wolman, Microsoft Research; Matt Welsh, Harvard University

Matt Welsh believes that 802.11 (WLAN) is not suitable for the new applications and classes of traffic that it is currently being asked to handle. The inherent problems can be mitigated if the access points and clients are all cooperating to maximize aggregate throughput, unlike 802.11, where all decisions are made by the clients with no coordination between clients or the access points. Welsh also noted that changing 802.11 is a very lengthy process (802.11e took over six years to complete).

Dyson is an extensible WLAN system that uses a central controller to gather traffic data from the access points and clients and allows the IT administrators to set policies. The policies are short Python scripts that can, for example, cause all clients to associate the access point with the lowest load factor or separate VoIP traffic from bulk TCP traffic (which greatly reduces jitter). This combination of data gathering across all participants and policy implementation gives an elegant solution to current WLAN problems.

An audience member made the observation that this could all be done without the need for a central controller, since all the APs and clients are communicating with each other. Welsh agreed but thinks it would be much more difficult to get decentralized decisions working. Someone also asked what the overhead of the data gathering packets was, to which Welsh responded that it is very, very low, since most of the information can be piggybacked on standard 802.11 control messages.

- ***ChunkStash: Speeding Up Inline Storage Deduplication Using Flash Memory***
  Biplob Debnath, University of Minnesota, Twin Cities; Sudipta Sengupta and Jin Li, Microsoft Research, Redmond

Sudipta Sengupta noted that using deduplication to decrease the amount of data stored in enterprise backup systems can save a significant amount of storage. It can also save network bandwidth if the target is not on the local machine (which can be very important if the target is across a WAN).

But in order to run deduplication at line speeds it is necessary to have a scheme for quickly looking up the chunk fingerprint (in their case a 20-byte SHA-1 hash) in the database of previously seen chunks. The problem is that with current data stores this database is too large to keep in memory. Previous systems have used disk-based database schemes with heavy caching, but there are still performance challenges.

Sengupta's team devised a scheme to use Flash memory to hold the database. Their system uses Flash-aware data structures and algorithms and strives for low RAM usage to allow for large Flash databases. Chunk metadata (chunk length and location) is organized on Flash in a log-structured manner, with a cuckoo hash table of the chunks in RAM. They also have a metadata cache in RAM.

Sengupta compared ChunkStash with using Berkeley DB to store the database on hard disk and SSD, showing that they get a 25x (HDD) and 3x (SSD) improvement over using Berkeley DB.

An audience member asked if deduplication can be done offline. That is, copy all the data to secondary storage and then dedup the secondary storage in batch mode before moving off to tape. Sengupta replied that it could be done, but you lose one of the benefits of local deduplication, which is decreasing network traffic if the backup system is remote.

JUNE 24, 3:30 P.M.–4:30 P.M.: INVITED TALK

- ***Google Books: Making All the World's Books Universally Accessible and Useful***
  Jon Orwant, Engineering Manager, Google

  Summarized by Italo Dacosta (idacosta@gatech.edu)

The Google Books project is an example of Google's philosophy of organizing the world's information. The main goal of this project is to digitize the content of all the books in the world, organize it, and allow everyone to search it. Jon Orwant, the leader of the Google Books project, presented the motivation behind Google Books, the challenges faced by this project, and the benefits and possible uses provided by this service.

Orwant said that Google Books can be divided into two parts: the publishers half and the libraries half. Today Google works with approximately 30,000 publishers worldwide. While publishers want their books to appear in Google Books, they demand that only 20% of the books' content be displayed as text snippets. Surprisingly, only 10% of the books received by publishers are in digital format. As a result, Google has to digitize most of the books provided by publishers.

According to Google Books' weekly count, there are approximately 174 million books worldwide. From this total, 20% are in the public domain (out of copyright), 10–15% are in print (copyrighted), and the rest are books that are presumably copyrighted but out of print. The problem with the books in the last category is that they are only available in libraries. Therefore, to make these books more accessible to the public, Google began to scan books from libraries in 2005. Since then, Google has been working with more than 40 public and private libraries and has scanned around 12 million books. Orwant estimates that all the books in the world will be scanned in 10 to 15 years. In addition, Orwant mentioned that libraries benefit from this project, because they can obtain digital copies of the books for backup purposes for free because no money is exchanged during the process. However, Google has been the subject of several lawsuits from groups such as the American Associa-

tion of Publishers and the Authors Guild regarding the fair use of the books' content. Orwant expects that a soon-to-be-approved settlement will allow Google Books to continue scanning books while providing additional benefits to libraries, publishers, and copyright holders.

The process followed by Google to digitize each book is conducted in seven steps. First, the book is obtained from the publisher or library and is scanned. Second, the book's scanned pages are enhanced using several image processing techniques (i.e., cropping, cleaning, de-warping). Third, optical character recognition (OCR) techniques are applied to obtain the text that will allow people to search the book's content. Fourth, the scanned book is analyzed to understand its structure (i.e., text flow, headers, footers, etc.). Fifth, the book is identified based on the metadata available from different sources. Sixth, the book is classified and indexed. In the seventh and final step, the digitized book is served in Google Books.

The process of adding books to Google Books faces several challenges such as: careful handling of library books; books in many different languages; multiple inaccurate, inadequate, and ill-formatted metadata sources; non-monograph books (e.g., boxed sets, series, and multi-volume works); the lack of unique book identifiers (e.g., ISBN); determining a book's contributors; and figuring out a book's structure (e.g., page numbers, publication year). To overcome these challenges, Google relies on different engineering and computer science techniques, as well as the creativity of Google's engineers (on their 20% time projects).

Finally, Orwant described how all the information gathered by Google Books represents a "corpus of human knowledge" and presented some examples of how to take advantage of this knowledge. He commented on the use of Google Books by researchers doing linguistic analysis (i.e., predicting the regularization of verbs and determining popular words in a particular decade). Also, Orwant described how Google Books could be used to test the "Great Man" hypothesis by determining if great ideas and discoveries could have been reported earlier in history by people in different cultures and places. These types of applications are possible because Google Books allows searching not only for phrases but also for concepts. In conclusion, Google Books exposes information that before was only available on library shelves, allowing everyone to ask questions that were not possible to be answered before.

During the Q&A, someone asked to what books the Google settlement applies. Orwant answered that the settlement applies to books scanned until May 5, 2009. The settlement also gives partial benefits to books scanned after that date and to future books. Orwant added that most of the settlement benefits only apply inside the US. Can the books covered by the settlement be scanned and sold by Google without the authors' permission? Copyright holders can decide if they want their books in Google Books or not. If a book is out of print and the author does not come forward, Google can sell the book and put the money in escrow until the author reclaims it. Orwant added that other companies as well as Google can sell the books. A short discussion on whether this was a fair practice followed. Another attendee asked if Google is planning to do the same with other forms of media. Orwant answered that it is a good idea but there are several technical and legal challenges associated with gathering information from other types of media.

JUNE 24, 4:30 P.M.–6:00 P.M.: WORK-IN-PROGRESS REPORTS (WIPS)

First three WiPs summarized by Aleatha Parker-Wood (aleatha@soe.ucsc.edu)

- ***Live Gang Migration of Virtual Machines***
  Umesh Deshpande, Xiaoshuang Wang, and Kartik Gopalan, State University of New York, Binghamton

Umesh Deshpande presented Live Gang Migration. Co-located virtual machines are often migrated for load balancing. Since VMs often share a lot of pages, this can result in many duplicate pages being sent across the network. Live Gang Migration identifies these identical pages and transfers only a single instance. Further instances are migrated by transferring a page ID to the remote machine. In their experiments, this resulted in 40% reduction in total migration time and 60% reduction in network traffic. Kai Shen noted that VM monitors already have a feature identifying identical pages, and Deshpande responded that there is a feature for sharing pages on the same host but that Live Gang Migration is for reducing duplicate pages during migration, a case which is not currently addressed.

- ***Remote Shadow I/O: A Framework to Achieve High Performance Remote I/O Using the Shadow Device State***
  Sejin Park and Chanik Park, POSTECH, Pohang, South Korea

Sejin Park presented Remote Shadow I/O. This work focuses on unmodified guest OSes running within a virtual machine (VM). Remote I/O is a significant amount of overhead for virtual machines. Currently, performing remote I/O to a hard drive requires the guest OS to go through VMExit Handler. However, in their analysis, 80% of disk I/O doesn't actually modify the disk, just reads or sets state in the file system. Only 20% of requests actually access disk. They propose to take advantage of this by maintaining a shadow device state in the hypervisor during the 80% of get/set disk I/O operations. When the 20% of real I/O occurs, updates are piggybacked into the write, in order to synchronize the state with the real disk state. Scheduling overhead is high, so the expectation is that 8.8% of performance can be improved for their test trace.

John McCullough from UCSD asked how this compared to paravirtualized devices such as for Windows. Park replied that they don't consider this to be a paravirtualized device. Dan Peek from Facebook asked whether there was extra latency that's added to a real request because of the additional

changes, since the system has to replay the shadow device to the real device. Park replied that the real device has the same latency.

- ***Designing a Snapshot File System for Storage Class Memory***
  Eunji Lee, Seung-hoon Yoo, and Kern Koh, Seoul National University, South Korea; Hyokyung Bahn, Ewha University, South Korea

Eunji Lee presented a new snapshot filesystem concept. Storage Class Memory (SCM) is non-volatile and byte addressable. It is expected to be widely deployed by 2020. It will likely replace hard disk drives, due to high performance and low power consumption. The authors wanted to build a snapshot file system which exploits the properties of Storage Class Memory. Storage Class Memory has no seek time but has a limited capacity. Current algorithms optimize seek time by using extra capacity, using copy on write, for instance. For Storage Class Memory, the authors suggest that systems should reduce space usage rather than seek time, using a "write in place" snapshot policy. Rather than creating a new root and new data in a new location, they copy the old data into a new location and overwrite the existing location with the new data. Rather than mounting a new root, the system needs to do more work to recompute an old version, but this is rare. To access it, the system restores using copy on write, updating the new data back to the old data.

Someone asked if this system was optimized for rollback versus time travel. Lee replied that it was. Peter Desnoyers asked how the system was maintaining the copies it made and whether they were linked off the old block. Lee replied that the old data blocks are contained in a list, with a pointer in the old inode to the list.

Last three WiPs summarized by: John McCullough (jmccullo@cs.ucsd.edu)

- ***Multi-Client Proxy Server on a Low-Power Wireless Router***
  Amal Fahad and Kai Shen, University of Rochester

Mobile devices are often limited by their connection quality and battery life. Wireless gateways can potentially improve the experience for mobile devices by leveraging their improved network connectivity and dedicated power source. Potential activities include caching/prefetching, media transcoding, Web site customization, offloaded computation, and security functions. The main challenge is supporting such high-demand services on a low-power device. So far they have studied the Squid caching proxy and found that it has a modest latency increase over a desktop implementation for cache hits, but for cache misses the writes have higher latency because of the shortcomings of the compact-flash storage media.

- ***SSDAlloc: Hybrid SSD/RAM Memory Allocation Made Easy***
  Anirudh Badam and Vivek S. Pai, Princeton University

Flash storage provides cheaper and more power-efficient storage than DRAM. While most Flash does not support byte-level access, it is still useful for increasing working-set capacity. Current techniques either involve custom coding to SSDs, which is labor intensive, or using SSDs as a swap backing store, which is not very well suited to the medium. This work provides a calloc style interface to a runtime that keeps objects in memory when in use, maintains unused objects in a packed form in RAM for caching, and manages log-structured page storage on the SSD. This approach provides transparent access with a 2–6x performance gain over SSD-backed swap. Information about the project can be found at http://www.cs.princeton.edu/~abadam/ssdalloc.html.

- ***Jboa Minicluster (Just a Bunch of Atoms): New Techniques for HPC***
  Mitch Williams, Sandia National Lab

Sandia has a long history of building portable mini-clusters for HPC demonstrations and small scale simulations. Historically, mini-clusters have been constructed from Pentium 2, Pentium 3, Geode lx800, Core2Duos, Via C7, and, most recently, Atom processors. Most of the work focuses on virtual machines and software. The goal is to hit 50M VMs. The current 16-node cluster gets 3K VMs on lguest using oneSIS, Clustermatic, and VMatic. The current goal is to study botnet-spreading behavior on a simulated Internet. Currently, they hope to look at other platforms, potentially including cell-phone style platforms, because Atom is slow as a cluster node.

JUNE 25, 9:00 A.M.–10:00 A.M.: INVITED TALK

- ***Reconstructing Ancient Rome: 700 Years of IT and Knowledge Management***
  Maximilian Schich, DFG Visiting Research Scientist at BarabásiLab, Center for Complex Network Research at Northeastern University

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

Documentation provides a fascinating view of our world. Today we have research projects that can construct 3D models of places like the Coliseum based purely on photos from Flickr. Beyond that we have Google Street View, which gives us views into even more obscure locations. These views give us a strong sense of what the world looks like today, but can we get a sense of what they looked like long ago?

Historical evidence provides only limited evidence. The best maps of ancient Rome include only one-third of the city, and it can be very hard to reconstruct what is missing. During the Renaissance there were many who documented ancient Rome, providing insight into ground plans and architecture—or at least part of it, as the documentation is heavy-tailed with concentration on the most popular monuments and little focus on anything else. The documentation that does exist has been through a remixing process.

The documentation process iterates through five steps: (1) study existing fragments or potential source documentation and surveys; (2) integrate the fragments, creating sketchy ground plans; (3) make a full reconstruction, which may have missing pieces; (4) re-fragment the reconstruction when publishing, losing the uninteresting parts due to the high cost of paper; (5) recombine the fragments, taking artistic liberties when putting them back together. This process repeats, losing more information, and introducing architectural pieces from one document to fill in gaps in another, or even making things up completely. Historically, we can observe the process of the "inductive surveyor" who adds documentation for monuments lacking any kind of source documentation.

This leads to a paradox of progress in modern archeology, as it tends to cite modern work and the ancient sources with little mention of the middle period. This falls in line with the practice of citing the original source rather than the place it appeared, but it is hard to know what the intermediate source may have introduced. Encyclopedias have collected the various historical documents. In the mid-20th century, researchers started putting together card indices to locate monuments and sculptures. The problem of citing the original, rather than the source used, persisted. In more modern forms, the card-indices were put in a database that gives you a UI to browse for documents associated with a monument, or monuments associated with a document, but provides little information on how they relate. Schich has used link-clustering to show cross-correlation between the Roman baths with maps and provide higher-level information than simple document queries and counts. Having full access to the data can be highly beneficial, because others may have a better idea for hot to interpret data than the simple structure a query UI can provide. The datasets are complex networks of complex networks, which are themselves part of larger networks. In many ways we are approaching high-throughput humanities: research databases have been on the order of thousands or tens of thousands, but now Google Books has scanned millions of books. Perhaps we can make a huge atlas of the humanities. For more information see http://schich.info.

An audience member, observing that we're drowning in data and that a lone person is inadequate, asked whether it would be more appropriate for a doctorate to be completed by teams. Schich responded that this question has come up before, as someone's life work might be reduced to two points on a line. There is enough complex overlap that we can't carve up the world into pieces for individual study. Someone else observed that there is a lot of aggregate data and asked whether there are ways to tag it with how valid it is and arrive at a probability of correctness. The trouble is that each person entering data has a different idea of the standard of correctness and you are back to the original problem. Ideally, we want to look at correlations and implicit citations and be able to toss out the junk.

JUNE 25, 10:20 A.M.–NOON

Summarized by Joshua Reich

- ***Sleepless in Seattle No Longer***
  Joshua Reich, Columbia University; Michel Goraczko, Aman Kansal, and Jitendra Padhye, Microsoft Research

Joshua Reich pointed out that idling PCs in corporate/enterprise networks waste significant amounts of power by idling. These machines generally have their OS settings disabling sleep because users and administrators want continuous and seamless access to these machines. Sleep-proxying systems were suggested as a solution to this problem over a decade ago. Yet they have not yet been deployed commercially. Reich argued that the key issue that need be considered here is the economic feasibility of the sleep-proxying system. The authors chose a sleep-proxying design for easy and economical deployment and maintenance. Their sleep-proxies reaction policy extends the best recommendations of previous work with their own customized improvements.

The reaction policy proposed by the authors is straightforward. Right before the client machine sleeps, it broadcasts a quick notification—informing the sleep proxy of the ports on which it is actively listening. The sleep proxy (which can be a lower-power, low-cost box—potentially even a client peer) then takes over, redirecting all traffic for the client to itself. It responds to IP resolution traffic, wakes the client only for incoming TCP connection attempts to the set of ports on which it had been listening, and ignores all other traffic.

Reich next shed light on the factors that impede the practical performance of sleep-proxying systems in real networks—identifying the twin problems of "crying babies" and "application-based insomnia." The first of these accounts for ~10% of lost sleep and is caused by other networked machines that attempt to connect to sleeping clients too often. The second accounts for ~90% of lost sleep and is caused by applications running on the host that prevent the host from sleeping in the first place. In both of these cases, it appears that IT servers and applications are the main troublemakers. The good news is that relatively low-cost approaches can likely be leveraged to schedule these applications in a coordinated fashion that will leave much more potential sleep time.

How much of the sleep savings you show came from your system as opposed to the default Windows sleep behavior? Reich said that in their environment it was 100%, as Windows sleep was disabled on all of their machines before their system was rolled out. In other environments, these savings would be reduced by one-third to 5%—depending on what proportion of the machines would have been sleeping. How does their system differ from the Apple sleep proxy system? Their system is geared to the home consumer and only works in their own closed ecosystem. In terms of reaction policy, they are quite similar (they support WiFi). However, the focus of the authors' work is on economic

deployment and learning the lessons of such and their main finding is that the IT setup is really what you have to worry about.

How much of the sleep achieved was due to the particular setup at Microsoft? Wouldn't machines wake more elsewhere? Reich answered yes, that's one of the main reasons why they chose an extensible software-based approach instead of a hardware NIC-based approach—so they could do blacklisting, whitelisting, etc. However, these additional wake-ups would really come from scanning machines and they are focused on the corporate network, which tends to be firewalled pretty heavily, not on more open academic networks where this would be more of an issue. You could implement pretty much any reaction policy you'd like (although LiteGreen-style virtualization wouldn't work) using their framework.

- ***Wide-area Network Acceleration for the Developing World***
  Sunghwan Ihm, Princeton University; KyoungSoo Park, University of Pittsburgh and KAIST; Vivek S. Pai, Princeton University

Sunghwan Ihm pointed out that Internet access in developing regions is a scarce and expensive commodity. Web proxy caching has been proposed as a solution to this problem in the developed world. However, this solution isn't adequate for the developing world, where there is significantly greater diversity of demanded content (and thus much less cache-able content). So the authors propose a combination of Web proxy caching and WAN acceleration. In this scheme WAN accelerators sit in both the developed and developing world, with data being chunked together, compressed, and sent using much less bandwidth. Chunk metadata is stored in accelerator memory, while data is kept on disk.

There is a significant challenge here—small chunking has a high compression rate (less extraneous data is put in a given chunk) and puts little pressure on the disk (fewer cache misses) but puts much more pressure on the memory (since many more chunk IDs need to be stored). Large chunking has the opposite trade-off: better for memory, but it puts pressure on the network and disk. Consequently, the authors proposed multi-resolution chunking (MRC), which uses large chunks to ameliorate memory pressure and disk seeking and small chunks to achieve high compression rates. They generate these chunks efficiently by detecting the smallest chunks first and then making their way up (data contained in small chunks may also sometimes be encoded in larger chunks).

The authors also took advantage of the assumption that there will be many meshed machines in such a developing-world network. If this is the case, they can trade off between network (grabbing content from peer memory caches) and disk (grabbing it off one's own disk) to maximize efficiency. The authors evaluated their work with simulation experiments and a small testbed implementation.

Why not apply these techniques for the developing world to the developed world? Sunghwan agreed.

- ***An Evaluation of Per-Chip Nonuniform Frequency Scaling on Multicores (Short Paper)***
  Xiao Zhang, Kai Shen, Sandhya Dwarkadas, and Rongrong Zhong, University of Rochester

Xiao Zhang described the problem as applying DVFS to all cores on the same chip. However, not all cores are doing (or need be doing) the same amount of work. The authors proposed smart scheduling to facilitate per-chip frequency scaling, thereby saving power on the cores eligible to be run at lower frequencies. To do so, they group applications with a similar cache-miss ratio on the same chip. This way, applications with high cache-miss rates can be run at lower frequency (since the processor will most often be blocking for I/O anyway), while applications with low cache-miss rates can be run at higher frequencies. This also removes pressure on the cache (as applications with a high rate of cache misses are not continually knocking the cache lines of applications with lower rates of cache misses out of the cache). Likewise, it reduces pressure on the memory bus.

They evaluated their techniques on a 2-chip Intel 3GHz WoodCrest processor (two cores per chip, sharing a 4MB L2 cache) SMP running Linux 2.6.18 by running 12 SPEC-CPU 2000 benchmark applications. They found that their techniques performed reasonably well. Moreover, it appears that the power savings they experienced can be reasonably approximated using a relatively straightforward model. They then applied this model to develop frequency scaling policies that provided reasonable power savings.

Someone asked why the similarity grouping without using frequency reduction raises temperature. A CPU working at full blast will generate more heat than two CPUs sharing load. Someone else pointed out that their performance prediction model assumes that the behavior of other cores doesn't affect performance of the core that they are modeling. Doesn't that seem odd, given that lots of other resources are shared? They are looking at stand-alone applications. Having several applications running on other cores will affect things, but they think it is a second-order effect. This holds on an SMP-based machine, not on a NUMA-based machine.

- ***A DNS Reflection Method for Global Traffic Management (Short Paper)***
  Cheng Huang, Microsoft Research; Nick Holt, Microsoft Corporation; Y. Angela Wang, Polytechnic Institute of NYU; Albert Greenberg and Jin Li, Microsoft Research; Keith W. Ross, Polytechnic Institute of NYU

Jin Li raised the question of how to best select one of many remote locations from which to serve a Web-based content request. In order for a provider to direct users to the server it desires, it will use DNS redirection/reflection based on the

IP address of the client and insert that into the client's local DNS. This can be done using a geo-location database or using an anycast solution.

Yet, how do we pick the best remote site for a given client? Passive DNS-based measurement can be used, but this has many drawbacks, particularly that the performance of some of the clients is significantly degraded. So most CDNs have used active probing techniques. However, many clients (~45%) cannot be probed actively. So instead they use DNS traffic (DNS reflection) to trigger DNS queries from any LDNS server. Essentially, when an LDNS server that cannot be actively probed makes a query to a top-level DNS server, that server reflects the query to a collector node. Then the time between queries is measured and the network path performance inferred.

Using 17 DNS servers and 274 Planetlab nodes, the authors show that DNS reflection tracks within 6ms of ping.

Someone asked if they had thought about applying this technique to similar passive measurement problems. Li said that they have some other work in this area (e.g., measurements of clients to CDN providers).

JUNE 25, 1:00 P.M.–2:00 P.M.: INVITED TALK

Summarized by Rik Farrow (rik@usenix.org)

- ***RoboBees: An Autonomous Colony of Robotic Pollinators***
  Matt Welsh, Associate Professor of Computer Science at Harvard University

The idea started as hallway talk. Bee colonies around the country have been dying off, yet bees are essential pollinators of crops. The idea turned into a short paper, then a team was recruited. Brainstorming was followed by creation of an outline, division of labor, and a funding request for $10 million, which the NSF actually granted.

But that's not where Matt Welsh started his talk. Creating a colony of robotic bees is not just a CS project, as there are many problems to solve. The researchers broke the problem into three main areas: the brain, the body, and the colony, with different teams working on each area.

The body shares some aspects with actual bees: a pair of veined wings and small size. The veins and associated wing corrugations are important for flight. For muscles, piezoelectric actuators that require 200 volts but just tens of milliwatts of power are planned, with a flapping frequency of 230 hertz—very similar to bees. First flight has been achieved, but only when tethered to a power supply.

Power is a critical issue. Batteries will not work, because of size limitations, so Welsh said they plan on creating fuel cells tiny enough to fit on chips. There are existing micro-fuel cells, but they run at 200–500° C and require hydrogen for fuel.

The brain must interpret sensors, control flight, and follow instructions. Welsh explained how optical flow can be used with a simple 64x64 pixel sensor from Centeye: if you want to pass through an opening between obstacles, you want the optical flow to be equalized on either side of the opening. If the optical flow is getting uniformly larger, you are about to run into a wall.

They plan on using an ARM processor and accelerometers that can be turned on or off as needed. The program will model neural control, keeping things as simple as possible.

For the colony, they need a non-centralized organization but robustness as well. Welsh described using a high-level language to create a program that would be downloaded to robobees to get them to search, for example. For now, they are experimenting with Blade mCX micro-helicopters, with a goal of having 50 helicopters under radio control. Welsh showed a video of a computer controlling a micro-helicopter via radio, flying briefly then crashing. There is obviously a lot of work to do here.

Dan Peek of Facebook pointed out that plants and pollinators co-evolved, and that he just wanted to pass along that idea. Dan Klein commented on a news clip that Welsh showed toward the end of his talk. Fox News had called the program "a good example of wasting government funds as only 1.66 people were hired," and Klein wondered who the .66 person was. Welsh explained that the grant was funding 1.66 post-docs. Jitendra Padhye wondered why use flapping wings, and Welsh said that one of the other researchers believes that this is the most power-efficient design. Maximillian Schich worried about birds eating robobees, and Welsh agreed that it was important that they be able to find lost hardware.

JUNE 25, 2:00 P.M.–3:00 P.M.

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

- ***An Analysis of Power Consumption in a Smartphone***
  Aaron Carroll, NICTA and University of New South Wales; Gernot Heiser, NICTA, University of New South Wales, and Open Kernel Labs

Aaron Carroll pointed out that smartphones have poor power consumption, as evidenced by how often we have to charge them. The situation is only getting worse as we add functionality without any fundamental increases in battery capacity. Current technology uses dynamic voltage and frequency scaling for computation, but in real systems the CPU doesn't use that much power. If we ask what does use power, the answer is often that nobody knows or vendors won't tell us.

To address this question, the authors instrumented an OpenMoko Freerunner to cover 98% of the components and measured the phone with a variety of benchmarks. In

the suspended state the phone draws 69mW and at idle it draws 269mW. Half of the idle power is in the GSM chip, and the GPU, LCD, and backlight draw significant power. At full power, the backlight alone draws 400mW. RAM and CPU are actually fairly power-proportional. When browsing email, the GSM draw is half of 610mW. When playing back locally stored video, the CPU and RAM are dominant. When playing audio, display power is high because SD access goes through the GPU.

The authors looked at the more modern HTC Dream (G1) and Google Nexus One (N1) for validation, assuming the power breakdown is similar. The G1 and N1 have better idling power because of improvements in 3G over the older chipsets, and they found that the radios draw similar power even with the large differences in data transfer throughput. Computationally, DVFS provides energy benefits for the Freerunner and N1, but the G1 works better completing at full power and then sleeping. This generally shows that DVFS can still be effective, even though it has been eschewed lately. In general, the biggest consumers of power are the display, the cell radio, and, in some cases, the CPU. Power is not going to RAM, Audio, Bluetooth, or storage.

One of the audience members asked about variance in the LCD power based on displays. The author responded that there is variation from 14mW for white to 70mW with black for some displays, but that it varies by display technology and in some cases you get the opposite. Therefore you have to be sure to match a power-saving designed theme with your phone. Was the platform measurable because of its construction and could you measure other phones if you had schematics? The OpenMoko is measurable because of construction, but other platforms are likely to be hard due to routing through multi-layer circuit boards. While you can do some inference from coarse measurements, the authors wanted better accuracy.

- ***SleepServer: A Software-Only Approach for Reducing the Energy Consumption of PCs within Enterprise Environments***
  Yuvraj Agarwal, Stefan Savage, and Rajesh Gupta, University of California, San Diego

Yuvraj Agarwal said that buildings represent a large fraction of total power consumption. While the lighting, heating, and cooling are all duty-cycled well, the IT loads are typically not. This is particularly worrisome as the amount of power dedicated to IT is expected to continue increasing. The authors instrumented the UCSD CSE building and found that IT loads constitute 50–80% of the total building power even when most of the machines are idle.

Most modern PCs support sleep states that reduce power consumption to 1–2% of idle. This represents a huge potential for power savings, yet most people don't put their computers to sleep. The problem is that users or IT departments want to access the computers remotely or the users want to keep downloads running and maintain IM or VoIP presence. Unfortunately, sleeping computers can't provide that directly. There is wake-on-LAN, but it requires the magic "wake-up" packet to be sent from the local network segment and is typically a usability non-starter. You can use a sleep proxy that solves the usability problem of wake-on-LAN and can provide high-level filters, but it cannot handle stateful applications and users leave their computers running at full power for simple downloads or to update emails while they're out to lunch. The other end of the spectrum is full desktop VM-migration that allows the computer to run all of its applications on a server while the desktop sleeps. But that requires heavy technological buy-in, and the degree of power savings is tightly coupled to the scalability of hosting heavyweight VMs on servers. The authors offer an alternative called SleepServer that has most of the functionality of VM migration with the same cost of the sleep proxy.

The goal, then, is to be able to maintain presence transparently, match proxying demands for each sleeping PC, be highly scalable, address enterprise management, and be multi-platform. SleepServer addresses ARPs, ICMP, and DHCP directly while providing the ability to wake up on user-defined filters for traffic like incoming ssh or remote desktop requests. Stateful applications such as background Web downloads need application-specific "stubs" that receive current state when the machine goes to sleep and transfer new state back to the associated application when the machine wakes up. SleepServer is implemented using lightweight virtual machines that maintain the IP and MAC addresses of the machine with all associated VLAN encapsulation. The VMs are provisioned with 64MB of RAM and 1GB of storage, which has been shown to scale up to 250 virtual machines on a single 300W test server.

The authors have a deployment with 40 users, many of whom would not otherwise put their computers to sleep, due to remote access needs or needing at least one stateful application. In early tests they found that automatic sleep policies are much more effective than manual activation, due primarily to forgetfulness. When using the automatic sleep policy, overall power savings averaged to 70%. Widespread deployment in the department could be supported by two servers and potentially result in a cost savings of $60,000 per year. For more information and measurements, see http://energy.ucsd.edu.

Can SleepServer handle 802.1X authentication? They haven't looked into it. Another audience member inquired about the complexity of customizing applications and stubs for each image. Agarwal responded that most users are typically covered by a few stateful applications. What would be lost if stubs were removed? A number of the users would not participate in SleepServer without some of the features, regardless of whether they are used. How can you translate to stubs? You need to modify the applications, though in general there are only a few stateful applications that the users are concerned about. Could SleepServer be implemented in a lightweight manner, something closer to a honeypot? The necessary functionality could be implemented in software,

but most of this functionality already exists in VM technology and there is an implementation trade-off.

JUNE 25, 3:30 P.M.–4:30 P.M.

Summarized by Paul Marinescu (pauldan.marinescu@epfl.ch)

- ***An Extensible Technique for High-Precision Testing of Recovery Code***
  Paul D. Marinescu, Radu Banabic, and George Candea, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Paul Marinescu started his presentation by arguing that current general-purpose software testing lacks the tools for testing error recovery code, as coverage information from various systems indicates. He then introduced a tool, LFI, that uses library-level fault injection to test error recovery code without making changes to the system under test.

Marinescu said that the real problem when doing fault-injection testing is finding good strategies to inject faults. He then focused on answering the when, where, and what to inject questions. He first introduced the notion of injection triggers, a mechanism that allows testers to specify with an arbitrary degree of precision when to inject. Then he showed a static analysis tool that can automatically find where to inject faults by choosing only the places where the return codes are not checked. Finally, he presented a different static analysis tool that can automatically infer possible error codes that an arbitrary library function can return.

The evaluation showed 11 new bugs LFI found in BIND, MySQL, Git, and PBFT, as well as the ability to improve line coverage of error recovery code from less than 5% to 35% for Git and 60% for BIND, entirely automatically, without writing new tests. LFI is open source software, available at http://lfi.epfl.ch.

How can LFI work without needing source code since some of its components were explicitly using source code information? Marinescu replied that source code or domain knowledge is not needed by LFI but can improve the results if available. How fast are the static analysis tools presented? The tools can analyze large systems (e.g., MySQL, libxml2) in a couple of minutes.

- ***Mining Invariants from Console Logs for System Problem Detection***
  Jian-Guang Lou and Qiang Fu, Microsoft Research Asia; Shenqi Yang, Beijing University of Posts and Telecom; Ye Xu, Nanjing University, P.R. China; Jiang Li, Microsoft Research Asia

Jian-Guang Lou argued that console logs are widely used by programs because (1) they are easy to use and (2) the free text format is very expressive. However, console logs are usually too big to manually parse in search of abnormal program behavior. The speaker proposed an automatic solution for interpreting log files. At its core, the solution relies on linear invariants based on the execution count of logging instructions. The linear invariants can be used to model control flow such as sequential execution, branching, or joining. Violations of these invariants indicate anomalies and also point to the place where the anomaly happened.

The problem is that automatically inferring the invariants for an arbitrary log file is NP-hard. Lou proposed a three-step solution for reducing the computational cost of the analysis: (1) the free text is transformed into structured text; (2) log entries are grouped according to the system variables they refer to; and (3) a hypothesis and testing algorithm is used to find the invariants. Several strategies including divide and conquer, early termination, and skipping are proposed to reduce the search space of invariant mining.

The evaluation consisted of searching for anomalies in Hadoop, CloudDB, and SharePoint log files. The approach was able to find anomalies in all the log files, out of which about 75% were caused by bugs.

Timothy Roscoe was interested in whether domain knowledge could be incorporated in the proposed algorithm. Lou said that is certainly feasible and could improve the accuracy of the analysis.