conference reports

Our thanks to the summarizers:

2005 USENIX ANNUAL TECHNICAL CONFERENCE

Rik Farrow
Charles Gray
Christian Kreibich
Nikitas Liogkas
Robert Marmorstein
Anthony Nicholson
Peter H. Salus
Andrew Warfield
Charles P. Wright

2ND SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI '05)

Dushyant Bansal
Ashwin Bharambe
Bob Bradley
Rebecca Braynard
Matthew Caesar
Ningning Hu
Ram Keralapura
Sherif Khattab
Robert Picci
Ashwin Sampath
Kevin Walsh
Bernard Wong

6TH IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS (WMCSA 2004)

Fahd Al Bin Ali
Mary Baker
Adrian Friday
Sachin Goyal

2005 USENIX Annual Technical Conference

Anaheim, California
April 10–15, 2005

KEYNOTE ADDRESS

Von Neumann's Universe: Digital Computing at the Institute for Advanced Study, 1945–1958

George Dyson, Historian and Author
Summarized by Rik Farrow

While Dyson's stage presence was not commanding, his story certainly grabbed the attention of his audience within minutes. Dyson told us how von Neumann, working with other mathematicians, developed a key idea behind today's computers, that numbers not only mean something, they also do something.

George Dyson delivers the USENIX '05 Keynote Address.

Dyson's past produced the access to information that made this story possible. Dyson grew up in Princeton, New Jersey, where his father, Freeman Dyson, had an office in Fuld Hall, along with Einstein, Gödel, von Neumann, and other well-known scientists. Dyson was granted access to archival information stored in the basement of Fuld Hall that pertains to the development of one of the earliest computers. Two other computers had been built previously, the Atanasoff-Berry Computer (ABC) at Ames, and ENIAC at the University of Pennsylvania. But neither computer included the concept of using numbers as order code, what we call machine code.

Dyson illustrated his talk with many pictures, diagrams, and logbook entries. The Princeton computer used 40 cathode ray tubes for memory, not display, storing data at pixels on each tube. The entire machine could be hand-cranked as a method of single-stepping through instructions, or could be run at 16 kilocycles max, using 16 kilowatts of power. Programs were small enough to be totally reliable, but the hardware was not. All programs and data were run at least twice, or more if the results differed. As Dyson quipped, this is very different from today, when we have sloppy code and reliable hardware.

Dyson made it clear that one of the ways von Neumann was successful was that he shared all of his research into computing, even with the Russians (this during the Cold War, when a main purpose of building computers was to aid in designing nuclear weapons). During the Q&A, Dyson agreed that this approach was similar to open source. Dyson had also pointed out that von Neumann was against patents, even though he took research money from IBM. A questioner asked whether Dyson was opposed to patents, and Dyson answered that he was not against patents in general, but he was opposed to many patents granted today.

Dyson has published three books, and it appears that the material included in his presentation will someday appear in his fourth.

GENERAL TRACK

DEBUGGING

Summarized by Anthony Nicholson

- ***Debugging Operating Systems with Time-Traveling Virtual Machines***

  Samuel T. King, George W. Dunlap, and Peter M. Chen, University of Michigan

  Awarded Best Paper

  In prior work, Sam King and others developed ReVirt, a method which used a virtual machine monitor to let the execution of a machine be "rewound" to any arbitrary point in the past. In this talk, he described how these ideas can be leveraged to allow better and more efficient debugging of operating system bugs, by allowing "time-traveling" inside the debugger.

  The authors argued that the common method of "cyclic debugging" (repeatedly rerunning a failed process and trying to detect the point at which its execution deviated from the expected path) is both costly in terms of user time and not guaranteed to find non-deterministic "Heisenbugs." By using ReVirt to log the execution of the entire machine, they eliminated the effects of non-determinism by replaying the exact sequence of instructions that caused the bug.

General Track Program Chair Vivek Pai congratulates a Best Paper author.

  For the example of a null pointer dereference, Sam noted that we only care about the *last* time the variable's value was modified before the crash. "Time-traveling" back to the exact point where the variable's value last changed is much quicker than forcing the user to rerun from the beginning, especially for long-running kernel processes and daemons.

  Their system periodically takes checkpoints to allow coarse-grained jumps back in time, and it uses instruction replay to get to a specific point between checkpoints. Sam described how they implemented the concept of a "reverse watchpoint" within gdb, and he gave a live demo of debugging a race condition within the Linux kernel using this reverse watchpoint tool.

- ***Using Valgrind to Detect Undefined Value Errors with Bit-Precision***

  Julian Seward, OpenWorks LLP; Nicholas Nethercote, University of Texas, Austin

  Valgrind is a framework for dynamic analysis of programs that provides a common infrastructure to varied specialized tools. Julian described one such tool, Memcheck.

  Memcheck is designed to catch runtime addressing errors (read/write to freed areas and/or array bound overruns), invalid malloc/free calls and memory leaks, and definedness errors (use of undefined variables). The authors noted that no other tool currently detects uninitialized values as Memcheck does. Memcheck also operates at bit-granularity, letting it catch errors in use of bitfields that tools such as Purify and Third Degree cannot. Since it is a dynamic tool, applications need not be recompiled.

  Memcheck shadows every memory value with both A bits (addressability) and V bits (definedness bits). Registers are shadowed with V bits only. All memory accesses are interposed, to allow updating of these bits, and all signal handlers and system calls are caught for the same reason. To prevent a flood of false positives, Memcheck delays reporting of errors until the access might cause an externally visible fault (such as control flow change). It is currently in wide use and their results show a 20–50 times slowdown with low false positive rates.

  A questioner asked how hard it would be to handle Java as well. Julian responded that Valgrind would need to understand Java's internal memory allocation, but that hooks could perhaps be written to facilitate this. Another questioner asked about using this on UML and/or compiling it into the kernel. Julian responded that this has been tried and abandoned, but that users often pull kernel code into a user-level harness just so they can debug with Valgrind/Memcheck.

- ***Pulse: A Dynamic Deadlock Detection Mechanism Using Speculative Execution***

  Tong Li, Carla S. Ellis, Alvin R. Lebeck, and Daniel J. Sorin, Duke University

  The authors argued that existing deadlock-detection schemes have limitations. Dynamic detection can detect but cannot always point to the cause of deadlock. Wait-for-graphs and static detection methods, such as RacerX, are accurate but useful only for lock-like resources. Tong presented Pulse, whose goal is to handle all resource types, not just locks.

  Pulse tries to look into the future to see the effects that unblocking one process would have on others, by speculatively unblocking each process in the set of long-sleeping processes. It discovers process dependencies by observing what else becomes unblocked as a result, thus generating a resource graph and detecting cycles. This is done by fork()ing a copy of a blocked process, unblocking the wait condition in the forked copy, and letting it run to completion or a specified timeout. Since these speculative execution paths must not change the state of other processes, they are not allowed to write to the file system or network, or send signals.

Tong showed how Pulse can be applied to buggy solutions to the classic "Smokers Problem" and "Dining Philosophers Problem," and also analyzed a version of the Apache Web server known to have a deadlock bug. The Apache bug is not detected by RacerX or wait-for-graphs because it involves pipes but is detected by Pulse. Their performance evaluation shows less than 1% slowdown in modified system calls, and less than three seconds execution time from start of deadlock detection to finish.

Margo Seltzer from Harvard University asked how Pulse handles applications holding a combination of kernel and application-level locks. Tong responded that Pulse is strictly application-level for now.

PLANNING AND MANAGEMENT

Summarized by Charles P. Wright

- ***Surviving Internet Catastrophes***

Flavio Junqueira, Ranjita Bhagwan, Alejandro Hevia, Keith Marzullo, and Geoffrey M. Voelker, University of California, San Diego

The authors define an Internet catastrophe as a worm that infects a significant number of hosts and corrupts or destroys their data. There are an increasing number of Internet worms, and some have corrupted data (e.g., the Witty worm). The traditional approaches to this problem are preventing, treating, and containing infections. The authors propose an orthogonal approach: surviving the catastrophe by preventing data loss.

To survive a catastrophe, each host replicates its data using informed replication. Informed replication assigns an attribute to hosts that have a specific piece of software that could be exploited (e.g., the OS or Web browser). To replicate data, a host computes a core, which is a minimal set of hosts such that for each attribute within the core one host does not have that attribute (e.g., at least one host has a different OS than the others).

The number of cores a given host may participate in is constrained by a load limit, which means that if hosts are too homogeneous it may not be possible to form cores. The authors performed a study of 2963 hosts on the UCSD network. They observed that configurations across OS classes are generally different, and that configurations within an OS class are often different. They concluded that there is enough host diversity to support cores.

Optimally, computing cores is NP-complete, so the authors evaluated several heuristics to compute cores, and found that they could construct cores with fewer than three hosts on average; with only two or three hosts, in fact, 99.9% of cores did not have attributes shared by all hosts (with random selection, cores needed to include at least nine hosts for the same coverage). The authors implemented a prototype using DHTs and evaluated it on PlanetLab, with similar results.

One audience member pointed out that the author's system becomes a common attribute. Junqueira responded that this problem is general to all distributed systems, there is still more protection, and only a single package needs to be perfectly secured. Another member asked if they considered worms that exploit multiple vulnerabilities. Junqueira said that it is covered in the paper.

- ***Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers***

Justin Moore and Jeff Chase, Duke University; Parthasarathy Ranganathan and Ratnesh Sharma, Hewlett-Packard Labs

The size and number of clusters are increasing—the top 500 clusters are four times larger than they were in 1999. This trend is mirrored in Web hosting (e.g., EV1 has 20–30k servers in a data center). For every watt of computing power, one additional watt of cooling is required. For a 10MW data center, the annual cooling bill is up to $8,000,000. The authors propose a software-only IT solution to this problem.

The authors propose two algorithms that schedule batch jobs on clusters, with the goal of allowing the air-conditioning units to cool the room more efficiently. The previous state-of-the-art thermal-aware scheduler was OnePassAnalog (OPA). The goal of OPA is to minimize temperature differences produced across the data center, by "poaching" power from adjacent machines. Unfortunately, OPA is difficult to implement, because each machine has a power budget that is somewhere between off and on. The authors developed a similar algorithm called Zone-Based Discretization (ZBD), in which machines are either powered on or off, and found ZBD to be within 2–3% of OPA.

The second algorithm the authors developed takes into account the thermal properties of each specific machine. Each machine takes cool air into its inlet, and then sends hot air out its outlet. Most of that hot air should go into the AC return duct, but some of it is recirculated into the inlet of other machines, thereby reducing cooling efficiency. If these bad machines are the last ones powered on, then cooling efficiency can be increased. The authors found that 20% of machines account for 60% of hot air recirculation, and that the minimum heat recirculation policy is 30% better than a uniform workload placement policy.

- ***Chameleon: A Self-Evolving, Fully Adaptive Resource Arbitrator for Storage Systems***

Sandeep Uttamchandani, Guillermo A. Alvarez, and John Palmer, IBM Almaden Research Center; Li Yin, University of California, Berkeley; Gul Agha, University of Illinois at Urbana-Champaign

Storage systems are difficult to manage, and human administrators can take three general types of

action when performance requirements are not met: (1) short-term (e.g., throttling or prefetching), (2) long-term (e.g., migration or replication), or (3) permanent (e.g., purchasing new storage). Unfortunately, it takes a long time for the human administrator to react, so a dynamic solution is required.

Chameleon is a resource arbiter for storage systems, with the goal of maximizing utility. The administrator defines bounds on latency and throughput for each workload, and Chameleon in turn manages the storage system by throttling and unthrottling various workloads. Chameleon optimizes a system of equations based on the component (e.g., disk or storage adapter) capabilities, and models of the workloads as inputs, with constraints derived from the SLOs.

The throttle values from the optimizer are sent to an action executor. Because Chameleon uses general models that are imperfect, the throttling may not have the desired effect. Therefore, the action executor learns if the throttling has the desired effect. If the model's accuracy falls below a certain threshold, the action executor falls back on heuristics.

The authors evaluated their system with several traces. One, a high-priority trace, was stopped and started throughout the experiment. Without Chameleon all three workloads violated their SLOs, but with Chameleon the SLOs were all met. When the workload models were changed to be unrealistic, the action executor detected this and switched to heuristics. The authors found that Chameleon reacted within 3–14 minutes, which is a fraction of the time a skilled system administrator would need.

IMPROVING FILE SYSTEMS

Summarized by Anthony Nicholson

- ***A Transactional Flash File System for Microcontrollers***

  Eran Gal and Sivan Toledo, Tel-Aviv University

  Sivan presented a new file system for NOR-based flash memories. It exploits the properties of NOR memories (as opposed to NAND) to allow transactional security for file operations, while using a minimal amount of RAM and leveling wear across all blocks on the flash device. It leverages a critical property of NOR devices: once a word is already programmed, one can go back and change bits in the word that aren't already set to 1.

  This allows them to use a modified version of b-trees to store file data. Their "pruned versioned search trees" are analogous to common file inodes. This trick of going back and modifying uninitialized bits in a word is used to make changes to the root node. Some "spare pointers" allow modification of file metadata without requiring wholesale copying or rewrite of an entire file, which both makes writes fast and reduces wear on the flash.

  Providing transactions allows for concurrency control, such as enforcing single writer/multiple readers of a file concurrently, in an efficient fashion. Their file system code is only 8500 lines of C, with a 24kb compiled footprint. Their results for three simulated workloads (fax machine, cell phone, automotive device) show good wear leveling and good performance up to the point where the file system is very full.

  A questioner asked about the limitations to using this approach with NAND devices. Sivan responded that the main problem is you cannot "flip" bits in NAND like this but must rewrite an entire sector to make a one-bit change. Another questioner asked why NAND is more common than NOR, given all of NOR's great properties. He responded that NAND is cheaper to manufacture, because it does not have access lines to each bit as NOR does.

- ***Analysis and Evolution of Journaling File Systems***

  Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

  Vijayan argued that modern file systems are too complex to comprehend, given current analysis methods. Code eyeballing only works if the code is available, and it can be tedious. Examining disk traces gives no semantic information about the cause of these mysterious disk block reads and writes.

  The authors introduced Semantic Block-level Analysis (SBA), which combines knowledge of file system design with block-level traces. The idea is to model and evaluate different file systems efficiently to find existing bugs and inefficiencies. SBA is implemented as a pseudo-device driver between the Linux generic I/O layer and the file system driver in question. The authors used various workloads to evaluate several file systems' write bandwidth performance, and they found that a flaw in ext3 unnecessarily limits parallelism of writes.

  Vijayan also discussed Semantic Trace Playback (STP), an extension which allows rapid evaluation of proposed modifications to file system design. As an example, he discussed how they used STP to evaluate their proposed solution to the ext3 parallelism bug detected by SBA. Simulation suggested an 18% performance improvement. This number was subsequently verified by actual implementation.

■ *Comparison-Based File Server Verification*

Yuen-Lin Tan, Terrence Wong, John D. Strunk, and Gregory R. Ganger, Carnegie Mellon University

Terrence described "server tee," a method for debugging servers that compares all outputs of a server under test with a reference server that is trusted to be operating correctly. A "tee" sits between the clients and the servers and intercepts all communications. The tee can therefore compare all output produced by the server under test with that of the reference server, allowing full testing without requiring that any servers be brought down.

The authors implemented a server tee for the Network File System (NFS). Terrence described many of the hairy implementation details, such as how to handle concurrent writes. He also described several case studies they ran. One compared a Linux 2.4 server and one running Linux 2.6, and discovered that the 2.4 server took NFS timestamps with second granularity, while the 2.6 server used ms granularity. Comparing the Linux 2.6 NFS server to a FreeBSD 4.7 server uncovered that the FreeBSD implementation has a bug whereby it does not send EOF at the end of a read reply to a FreeBSD client, as required by the NFS protocol specification.

The authors are currently working on an NFSv4 tee. One challenge is that NFSv4 servers must store state, so that state must be mirrored at the tee for all reference and test servers. NFSv4 also supports callbacks, which must be dealt with as well.

Jason Flinn, from the University of Michigan, asked if the tee could be used to evaluate servers via trace replay. Terrence responded that the "clients" in their system could be either actual live clients or trace replay sources.

INVITED TALKS

■ *DDoS Defense in Practice and Theory*

Eddie Kohler, UCLA and Mazu Networks

Summarized by Robert Marmorstein

Kohler presented an optimistic and extremely detailed picture of the state of the art in defending against distributed denial of service attacks. Approaching the problem from an operating system perspective, he presented a very thorough analysis of DoS and described how the industry is responding to changes in the cyber-landscape. Additionally, he described what additional changes are needed to adequately address the problem now and in the future.

Kohler defines DoS broadly as resource exhaustion that can affect either the target resource or legitimate users. The key characteristics that distinguish a DDoS are the high ratio of attackers to victims and the use of zombies or address spoofing. Unlike defacing Web sites or stealing credit card data, the direct gain to the attacker in a DoS is usually intangible, but incentives are changing very rapidly. Instead of rebellious teenagers, we now have to consider crackers from the Russian mafia who use DoS as a threat to extort money from porn and gambling sites.

Kohler also pointed out that a DoS can use either malicious or innocent traffic to achieve its end. Furthermore, a DoS may crash the host or merely slow it down. The former attack he labels "malignant," while the latter is a "pseudo-benign" attack. Malignant attacks use a small number of packets to bring down infrastructure. They often take advantage of the fact that protocol developers don't pay enough attention to error cases. Pseudo-benign attacks rely on the sheer volume of packets targeting the victim to inhibit service. Because a hacker can now purchase millions of zombied hosts to carry out a large-scale attack, pseudo-benign attacks are becoming an increasing concern.

After identifying these characteristics of DoS, Kohler pointed out that, in both cases, what makes DoS practical is that the target performs useless wasted work instead of servicing legitimate users. This suggests that the best solution is to redesign protocols to eliminate unnecessary work, to more carefully prioritize work, and to drop all work as early as possible. In addition to minimizing work on the victim, successful approaches will maximize work for the attacker.

Kohler also discussed ways to identify the source of an attack. While it is theoretically impossible to distinguish DoS from legitimate flash crowds, large ISPs can, in practice, identify suspicious net-flow patterns that identify attack sources.

Kohler also described several particular means of carrying out a DoS and showed how intelligent solutions to them fit into his analysis framework. He concluded by stressing his argument that the right place for solutions is the operating system rather than the network architecture and that an optimistic view of the foreseeable future is reasonable.

Former Board President Andrew Hume enjoys USENIX '05.

- ***Online Gaming***

  Mark Wirt, Butterfly.net

  Summarized by Rik Farrow

  I am not a game player, but the technology behind the current Massive Multiplayer Online Games (MMOGs) intrigues me. Wirt started out with a short history of online games, ranging from text-based role-playing games to today's game environments where participants control avatars in 3-D.

  There are real challenges in supporting such environments, largely because of scale but also because of the requirements of game playing. There are an estimated 10 million game players today, with over five million people playing one game (Legend of Mir). Everquest, at 345,000 players, is much smaller by comparison. But imagine having 300,000 customers making transactions and expecting instantaneous results—online game playing means you cannot have players wait seconds for a result and remain interested in the game.

  Wirt outlined some of the technical challenges. The world of the game must appear consistent and responsive to the players. That means that actions that change the state of the game must be visible to all players as soon as they occur. But the computing requirements for games mean that the games' back ends run on distributed clusters of computers. These clusters face a nondeterministic loading environment. For instance, there is no way to know in advance when the number of active players will double, which would require also doubling the number of game software servers to avoid degrading the quality of the game player's experience.

  The number of persistent objects can be greater than 2^{32}, and changes to these objects and other state require distributed transaction processing.

  There are also people who cheat, known as griefers. Cheating adds requirements for security, including data signing, protection of sensitive data, avoiding DoS (hiding players' source IP addresses), and blind protocols.

  Butterfly.net designs and builds software that provides a foundation for MMOG designers. Rather than having to focus on needs for a secure, scalable, distributed, transaction-oriented environment, the game designer can focus on the game itself.

  MMOGs are already big business, estimated to currently gross $1 billion per year. One questioner asked about profits; Wirt said they are razor-thin at this point. Another person asked about bandwidth limitations, whether, for example, a dial-up player is at a disadvantage compared to a broadband-connected player. Wirt said that game designers avoided using high bandwidth network transfers because that increases the cost of running the game (the game servers require more bandwidth).

  The entire area of MMOGs looks fascinating from a networking and sysadmin perspective, with tremendous challenges. It is also an area that is exciting to hardware vendors. IBM is a backer of Butterfly.net, and Sun Microsystems has developed a Java back end for gaming.

FREENIX TRACK

SOFTWARE TOOLS

Summarized by Robert Marmorstein

- ***ScmBug: Policy-Based Integration of Software Configuration Management with Bug-Tracking***

  Kristis Makris, Arizona State University; Kyung Dong Ryu, IBM T.J. Watson Research Center

  Software content management (SCM) and bug-tracking software are two of the key enabling technologies for large-scale software development projects. In this talk, Kristis presented a tool for coordinating these two technologies. ScmBug is designed as a generic intermediary that can easily connect any bug-tracking front end (including Bugzilla) to a variety of content management back ends such as CVS or subversion. ScmBug is policy-based, which means that it forces user submissions to satisfy constraints such as minimum content length.

  ScmBug makes it easier to understand the context of changes to a software project. Bug fixes, for instance, are directly connected with bug reports, and new features are connected to requests for them. The system is implemented using a two-component architecture. The glue component provides a common interface for all content management systems. The integration daemon handles user interaction with the tool.

  In addition to describing the design of ScmBug, Kristis described the development challenges of creating the software and the benefits of his tool over existing work. He pointed out that most existing tools connect a specific content manager to a specific bug-tracking system and do not scale well to new versions of either component. He also described a real-world deployment of the system and data obtained from that deployment.

  A demo of ScmBug is available at http://bugzilla.mkgnu.net, and the tool may be downloaded from http://freshmeat.net/projects/scmbug/.

- ***Linux Physical Memory Analysis***

  Paul Movall, Ward Nelson, and Shaun Wetzstein, IBM

  During their development of an embedded system, the authors discovered a need for detailed analysis of physical memory in order to prevent out-of-memory conditions. Unfortunately, the level of detail they needed was not provided by any existing kernel mechanism. In response to that need, they developed a kernel module to expose

detailed physical memory information, and an analysis tool suite to express that information in a easily readable way. Using their tools, they were able to discover a missing option flag in their build process that solved their problems with memory usage. While the tool suite is currently designed for Linux 2.4, they also plan on developing a version for kernel 2.6.

Their tools provide several advantages over existing memory analysis tools. Tools like mpatrol and memprof are focused on the memory profile of a single process and don't provide a good view of the overall usage of physical memory by the system. In particular, existing tools provide poor information on shared library use. While the proc file system supplies some information about physical memory, it doesn't show information about the layout of pages and so is of little help.

At the moment, the authors' tool suite does not handle System V shared memory or mmap scenarios, but they plan to explore this in future work. They also plan to replace the kernel module with a netlink interface to allow the analyzer to read on a socket.

EMULATION

Summarized by Charles Gray

- ***Running Virtualized Native Drivers in User Mode Linux***

  V. Guffens and G. Bastin, Université Catholique de Louvain

  Guffens discussed simulating wireless device drivers in virtualized operating systems to test and debug wireless protocols. The talk was divided into three parts.

  The first part provided a brief background to User Mode Linux. User Mode Linux is a paravirtualized (non-transparent) system for running multiple guest instances of the Linux operating system on a host operating system. This allowed the authors to simulate multiple wireless nodes on a single physical machine.

  Next, a virtual wireless driver was added to the guest Linux operating systems, which communicate through the host "tun/tap" interface. A physical layer simulation was added to model wireless networks as nodes moved in space.

  This environment allows easy testing of wireless routing protocols as nodes move around in space and drop in and out of contact. The simulation environment also had a GUI demo that showed nodes moving and how packets were routed in real time.

  In the third part, the authors listed a number of applications of this work, not limited to: testing ad hoc on-demand distance vector routing; running true-to-life simulation, only simpler; and debugging protocols (debugging the asymmetrical link problem was demonstrated) in a realistic environment.

- ***USB/IP—A Peripheral Bus Extension for Device Sharing over IP Network***

  Takahiro Hirofuchi, Eiji Kawai, Kazutoshi Fujikawa, and Hideki Sunahara, Nara Institute of Science and Technology

  Awarded Best FREENIX Paper

  The goal of this work was to provide seamless device sharing between computers. Hirofuchi pointed out that existing device sharing won't share fine-grained operations of devices, and typically exports only high-level interfaces such as file systems via NFS. There is currently no system that can play or eject a DVD in a remote drive.

  The authors identify that there are existing peripheral buses (e.g., USB in operating systems), so it is possible to make a pseudo-bus device for remote devices. This work is done with USB by implementing a virtual host controller device (the very bottom of the USB stack) and transmitting the requests over IP to remote nodes.

  An implementation of this has been done in which you can access all remote devices properly over the network. It is fully functional, network transparent, and interoperable between operating systems. It is also general, supporting many different devices.

  Foreseeable applications for this work are thin clients, PDAs, and central device servers. Issues with the system revolve around the fact that IP is not USB, so bandwidth, packet loss, latency, and jitter can vary greatly. Experiments show, however, that all devices work in an implementation on Linux 2.6.

  Experiments were conducted on both bulk transfers and isochronous devices. Some clever implementation is required to deal with packet loss at the device side for soft real-time devices.

  Future work is to implement USB over UDP instead of TCP, and also to test over wireless networks.

  The authors found that USB/IP provides transparent sharing of devices over a network, all devices tested seem to work, and there is sufficient I/O performance over a LAN.

  Further details can be found at http://usbip.naist.jp/.

FREENIX Track Program Chair Niels Provos congratulates a Best Paper author.

NETWORKING

Summarized by Robert Marmorstein

- ***Trickle: A Userland Bandwith Shaper for UNIX-Like Systems***

  Marius Eriksen, Google

  Motivated by a need for ad hoc bandwidth control on his desktop, Marius implemented an easy-to-use and portable bandwidth shaper for home and small office networks. It can be run without special privileges by ordinary users and allows collaboration between multiple processes with priority in addition to a stand-alone single-process mode.

  Trickle uses features of the dynamic linking system to preload middleware into an existing executable. By overriding socket functions, the tool implements simple rate limiting for TCP connections. If libraries are loaded in the correct order, Trickle does not interfere with other preload libraries such as corkscrew.

  Marius also developed a daemon that can coordinate multiple Trickle processes to provide global shaping per host (or even across a local network). To prevent bursty I/O, the daemon uses a smoothing algorithm that imposes a maximum length parameter across all sockets.

  Marius plans to expand Trickle by allowing more expressive policies and improving the smoothing algorithm. Trickle is available from http://monkey.org/~marius/trickle/.

- ***A Tool for Automated iptables Firewall Analysis***

  Robert Marmorstein and Phil Kearns, College of William and Mary

  Passive firewall analysis is a technique that has been used by commercial closed source software tools to allow system administrators to examine and test Checkpoint and Pix firewalls for configuration errors completely offline. Unfortunately, these tools do not support iptables. Robert presented an open source tool, ITVal, for performing passive analysis on iptables rule sets and gave several examples of how a system administrator can use the tool to catch common configuration mistakes.

  ITVal generates multi-way decision diagrams which represent the rule set of the firewall and a set of queries specified in a plain-English query language. It solves the queries using decision diagram operations, which are very quick and consume little memory.

  Because ITVal is a passive analysis tool, it can examine the entire search space of packets potentially seen by the firewall. Active tools such as SATAN and Nessus, in contrast, can examine only some of the packets, due to bandwidth considerations.

  Robert hopes to expand the tool to provide better support for packet mangling and analysis of multiple connected firewalls. He also hopes to pursue means of using the analysis engine for automatic firewall repair. A downloadable copy is available from http://www.cs.wm.edu/~rmmarm/ITVal/.

- ***Grave Robbers from Outer Space: Using 9P2000 Under Linux***

  Eric Van Hensbergen, IBM Austin Research Lab; Ron Minnich, Los Alamos National Labs

  Plan 9 is a research operating system developed at Bell Labs that provides an elegant distributed file system. Eric Van Hensbergen presented 9P2000, which brings some of the features of the Plan 9 file system to Linux. One motivation of the work is to provide a unified solution for resource sharing and control on commercial cluster systems.

  Unlike the Linux device module, which is heterogeneous, Plan 9 treats every device as a file, which means developers have a single, simple API for resource management. Because device access is performed generically, the file system can be distributed using a wide variety of transport mechanisms, including a serial link, TCP, or shared memory.

  Porting these features to Linux poses significant challenges because the semantics, especially caching semantics, differ widely, but a prototype implementation can be used with the 2.6 kernel. Preliminary benchmarks show comparable performance to NFS.

  More information about 9P2000 is available at http://v9fs.sourceforge.net.

GENERAL TRACK

DEFENDING AGAINST ATTACKS

Summarized by Christian Kreibich

- ***Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks***

  Katerina Argyraki and David R. Cheriton, Stanford University

  Katerina Argyraki presented Active Internet Traffic Filtering (AITF), a mechanism for distributed, collaborative filtering of DDoS attacks. Katerina explained that the basic problem during a DDoS attack is that filtering of bad traffic needs to occur before the victim's uplink is congested, i.e., at the very least, at the victim's main Internet gateway. However, filtering out the malicious flows at a single router faces at least two fundamental hurdles: First, source address spoofing renders the source address of a flow useless, and second, while a DDoS attack can consist of millions of flows, filters are a scarce resource in routers. Typically implemented in expensive TCAM (ternary content addressable memory), current capacities are only on the order of 256K entries. Thus, a single router is insufficient for blocking the offending flows.

  To present her solution to this problem, Katerina started from the basic premise of a route recording mechanism built into AITF-

enabled routers that allows the victim gateway to reliably identify common paths in the traffic by recording the addresses of AS border routers. Once a policy component not further addressed in this work identifies the attackers, the victim gateway sends a filtering request to the router closest to the attacker, while temporarily filtering out the attack traffic itself. Katerina proceeded by extending this basic mechanism to make it secure and resilient to non-cooperative nodes.

First, the message exchange is secured using a hashed nonce similar to TCP SYN cookies. Second, liars are caught using shadow filtering tables on both the victim's and the attacker's gateways, tracking the request and release of filters on a longer timescale than the actual filtering duration. Non-cooperative traffic sources are dealt with by disconnecting them, and non-cooperative gateways are worked around using escalation, i.e., moving the filter handling to a gateway closer to the victim. To prevent path spoofing (where nodes could transmit fake path information in packets at high rates in order to get a different, innocent source to be cut off), the attack gateways first check whether they actually transmitted the traffic they are requested to block, using a hashed nonce mechanism similar to the previous one.

Simulation results based on RouteViews data and DaSSF showed a high filtering gain: The victim's gateway manages to filter two orders of magnitude more flows than it uses filters. Katerina finished by concluding that AITF is scalable, incrementally deployable, and can filter millions of flows using only thousands of filters per router.

After the talk, an attendee pointed out that more subtle attack patterns might make it hard for the mechanism to kick in once the victim uplink is congested. Katerina argued that the assumption made is that an attack can be detected before it is too late to propagate filtering requests. Another attendee remarked that in past years, lots of path-marking schemes were proposed but none were deployed, and asked why AITF would be any different. Katerina pointed out that the only thing required to enable deployment is a viable business model. As an example she noted that a victim's gateway will typically be within an ISP, so this ISP could offer this mechanism as a service.

- ***Building a Reactive Immune System for Software Services***

  Stelios Sidiroglou, Michael E. Locasto, Stephen W. Boyd, and Angelos D. Keromytis, Columbia University

  Stelios Sidiroglou presented the design and implementation of a system that enables software applications to automatically and gracefully recover from failures. The work focuses on server-type applications, focusing on high availability and employing a transactional model. The types of failures addressed cover illegal memory dereferences, division-by-zero exceptions, and buffer overflows. Availability of the application's source code is currently assumed.

  Stelios next presented the three main components of the system. First, a set of sensors monitor an application's execution for faults. Once a fault is detected, the sensory information is used to identify the region in the code where the flaw occurred. Second, Selective Transactional EMulation (STEM) safely emulates the code in those regions. Occurrence of a fault is detected in emulation and handled by undoing all memory changes made by the current function and forcing it to return a default value compatible with the function's return type. Third, a test environment is used to evaluate the hypotheses of the effects of possible fixes to check whether the "cure" works against the input known to cause the fault to occur.

  Stelios pointed out that the approach works well in practice: Investigation of Apache, sshd, and BIND sources shows that in each case just under 90% of forced returns allowed the application to continue. While the overhead of the system can be large, selective use of emulation allows the slowdown to stay in the 1.3–2x range. Stelios explained that the downtime incurred by the scheme can be amortized over time, because it only occurs once per fault. Future work includes doing away with the need for source code availability by using debugger-style hooks, improving emulator performance by adding instruction caches, enriching the set of detectable flows, and more.

  One attendee asked whether one couldn't simply fix the code once the location of a flaw is determined. Stelios pointed out that the point of the system is to automate the healing process and do away with the need for human intervention. Another attendee doubted whether the rollback of the memory to the state existing when a function's execution started does indeed cover all possible side effects. Stelios explained that no memory changes are committed to the actual processor environment until the end of the emulation of a flaw's environment. When asked whether the willful modification of an application's semantics couldn't mess up an application's persistent state and cause trouble later on, Stelios admitted that this could cause problems.

- ***Attrition Defenses for a Peer-to-Peer Digital Preservation System***

  T.J. Giuli and David S.H. Rosenthal, Stanford University; Petros Maniatis, Intel Research; Mary Baker, Hewlett-Packard Labs; Mema Roussopoulos, Harvard University

  T.J. Giuli presented an improved communication protocol for the

LOCKSS digital preservation system, aiming to provide long-term protection of content in the system from sustained network-layer and application-layer flooding attacks, which he termed "attrition" in this context. Starting with a simple example of the LOCKSS audit protocol, TJ showed how a random subset of the peer population can ask for participation in an opinion poll about the correctness of a document. Some of those nodes may be busy, while the remainder will vote for the quality of a document by returning a hash value of their local copies. If the requester sees agreement in the received values, the local copy is deemed intact; otherwise it needs to be repaired. TJ then illustrated the spectrum of attrition attacks that can break this process: Access link flooding can prevent a node from performing meaningful communication, while flooding a node with frivolous poll requests can prevent a node from doing meaningful work.

To make the system resilient to these attacks, TJ proposed three strategies: admission control, desynchronization, and redundancy. Admission control employs reciprocity and effort-balancing filters, allowing a node to control the rate at which it considers poll requests from others. As a result, nodes that request at a similar rate as the node itself are favored. Desynchronization makes peers solicit votes individually rather than synchronously, avoiding inadvertent synchronization that can prevent the system from delivering services. Finally, substantial redundancy requires attackers to simultaneously suppress communication between the targeted peers for a substantial period of time.

TJ concluded by presenting simulation results obtained by the Narses simulator, using 100 peers with between 50 and 600 documents suffering a sustained attack pattern lasting over two years. Using access failure probabilities, friction coefficients, and the cost ratio incurred for the adversary as metrics, he showed that over-provisioning the system by a constant factor defends it against application-level attrition of unlimited power.

Someone asked whether it wouldn't make sense to employ cryptographic mechanisms to use authentication as well. TJ explained that while they are looking for a fully global consortium of libraries, it is very challenging to establish a global body that can agree on who an (un)authorized user is while at the same time keeping that knowledge around for many years. Peter Honeyman of CITI, University of Michigan, asked whether there could be a more systematic way to characterize this threat environment. TJ confirmed that this would be useful and pointed out that a significant percentage of peers are indeed considered risky by the system. The reciprocity filter is an important component of the system, and other mechanisms surely could be found that would help formalization.

IMPROVING DATA MOVEMENT

Summarized by Charles P. Wright

- ***Peer-to-Peer Communication Across Network Address Translators***

  Bryan Ford, MIT; Pyda Srisuresh, Caymas Systems, Inc.; Dan Kegel

  Many compelling applications such as teleconferencing, VoIP, and games need peer-to-peer communication. Peer-to-peer communication works well across the public Internet as the hosts can communicate directly. Unfortunately, when both of the hosts are behind network address translators (NATs), this is no longer true. This problem is growing, as more home users and businesses are using firewalls with NAT; and even some ISPs are beginning to deploy NAT (especially those in developing countries).

  The authors described two methods of allowing two NATed hosts to communicate. The first is UDP-hole punching, in which each host binds to a UDP port and then sends a packet along with a piece of identifying information to a rendezvous server (the IP address is not enough because many hosts can originate from the same public NAT address). One of the hosts then asks the rendezvous server to help them reach the other host. The rendezvous server sends back the endpoint that the host already established and the hosts can directly communicate. The second, more novel method is TCP hole punching, which works similarly to UDP hole punching but relies on TCP's simultaneous open behavior.

  There are several ways that hole punching can go wrong, and the authors performed a survey of Internet users to determine how many of them had appropriate hardware. They found that out of 380 hosts surveyed, 82% support UDP hole punching. Out of 286 hosts surveyed, 64% of them supported TCP hole punching. The authors concluded that current compatibility is good for UDP and tolerable for TCP. Finally, as NAT vendors become more aware of these techniques, compatibility will increase.

- ***Maintaining High Bandwidth Under Dynamic Network Conditions***

  Dejan Kostic, Ryan Braud, Charles Killian, Erik Vandekieft, James W. Anderson, Alex C. Snoeren, and Amin Vahdat, University of California, San Diego

  Content distribution is a fundamental service for a wide range of apps (e.g., software updates, virus signature distribution, multimedia distribution, etc.). The authors developed a system, Bullet' (Bullet Prime), with the goal of delivering large content from a single source to a large set of interested clients as quickly as possible. The authors make the assumption that the receiving nodes are willing to coop-

erate, so they are not concerned with freeloaders.

Bullet' splits the content into data objects (e.g., file blocks). No global state or global communication is used, because it doesn't scale well for a variety of reasons: (1) the network topology is unknown, (2) the network conditions are changing, and (3) data retrieval is fast so the information would be constantly out of data. Bullet' constructs an overlay mesh to leverage "perpendicular" bandwidth, and the source sends disjoint data to each of its peers to ensure diversity.

Peers in Bullet' use RanSub to discover a random subset of peers to connect to. Bullet' uses an adaptive peering strategy (with additive increase and decrease) so that the incoming link is filled but there is no excess contention.

The authors compare Bullet' to several systems and found that for a 75MB file, Bullet' is near optimal for both ample and constrained bandwidth. In this case, Bullet' has 20% better median performance than BitTorrent. In another experiment, core bandwidth was cut between nodes. Bullet''s performance was twice as good as BitTorrent's.

- ***Server Network Scalability and TCP Offload***

  Doug Freimuth, Elbert Hu, Jason LaVoie, Ronald Mraz, Erich Nahum, Prashant Pradhan, and John Trace, IBM T.J. Watson Research Center

  Network server performance is not scaling with CPU speed and network bandwidth. This is mainly due to memory speeds not scaling with CPU speeds, but other factors include the heavy costs of interrupts, device accesses, and DMA. The authors state that scalability is different from performance. For example, when you increase the CPU speed you only get between 43–60% of the performance benefits you should. TCP offload can help scalability in several ways. First, more efficient use of the I/O bus is possible because large transfers that are not constrained by the maximum segment size can amortize costs. Interaction with the network adapter can be reduced, and caches can become more effective.

  The authors wanted to avoid benchmarking a particular instance of a TCP offload engine, because such analysis often finds implementation problems, not fundamental constraints. They developed a software-based prototype in user-space using the TCP/IP stack from Arsenic, and they measured DMA transfers, bus cycles consumed, and the number of bytes transferred. Their prototype had two modes, one with a simple API, and another with a batch-oriented API that deferred sending requests to the simulated offload engine until 10 requests were ready or a 10ms timeout expired.

  The authors had significant gains for each metric. DMA was reduced by up to 88%, bus cycles were reduced by up to 23%, and up to 18% fewer bytes were transferred. In the future, the authors plan to use the Mambo full-system simulator to look at more metrics and experiment with arbitrary hardware, explore trade-offs in batching, and explore partial TCP offload. More information can be found at www.research.ibm.com/people/n/nahum.

SHORT PAPERS I

Summarized by Anthony Nicholson

- ***A Hierarchical Semantic Overlay Approach to P2P Similarity Search***

  Duc A. Tran, University of Dayton

  Peer-to-peer networks are extremely dynamic, since nodes come and go at random times. There is also a strong incentive to have minimal use of centralized servers. Duc described a proposed solution, EZSearch, which supports exact, k-NN, and range queries in an accurate and efficient fashion. Each peer in his system has small control and storage overhead, but no centralized servers are required. Duc described how he leveraged his prior work on the ZigZig hierarchy to form node clusters in such a way that the maximum routing distance is O(log n), with constant overhead on failure for reconnection. The key concept is that nodes can reuse this ZigZig hierarchy incrementally as new clusters form, rather than having to regenerate it. As a result, each time a new cluster forms, only a small number of connection changes result.

- ***A Parts-of-File File System***

  Yoann Padioleau and Olivier Ridoux, Campus Universitaire de Beaulieu

  Yoann argued that file hierarchies can be a problem, because files have natural semantic parts. Users would like to be able to switch from one organization of a file to another quickly and easily. His parts-of-file file system lets users apply different views to a file, to see, for example, a C source file without any comments, or without any of the #define statements. He described how this file system uses transducers to assign multiple properties to different parts of files. These transducers are easy to write for many common files, consisting of approximately 70 lines of Perl code.

- ***BINDER: An Extrusion-Based Break-In Detector for Personal Computers***

  Weidong Cui and Randy H. Katz, University of California, Berkeley; Wai-tian Tan, Hewlett-Packard Laboratories

  Current malware detection techniques rely on a priori signature knowledge. Weidong argued that this is undesirable, since these signatures must be generated by some centralized authority that we must trust, and the generation takes time, during which we are vulnerable to attack. BINDER detects new, unknown malware without requiring signatures. It observes user behavior and tries to determine the user intent behind every network

connection. It uses a whitelist to prevent false positives on acceptable network daemon processes. BINDER determines user intent by tracing back the event chain to see whether the user initiated the network operation (e.g., clicking on the Firefox icon ultimately results in a network fetch operation of http://www.google.com/). In summary, Weidong said that BINDER reliably detects the large class of malware that runs as a background process, does not receive user input, and generates outbound user network connections. He described several possible workarounds—for instance, the malware could fake user input via system-call APIs so that BINDER would think its network operation had been initiated by the user.

- ***Proper: Privileged Operations in a Virtualized System Environment***

  Steve Muir, Larry Peterson, and Marc Fiuczynski, Princeton University; Justin Cappos and John Hartman, University of Arizona

  The authors argue that interaction among multiple virtual machines hosted by a single virtual machine monitor can be valuable. This interaction could also be between a VM and the VMM itself, to allow processes running inside a virtual machine to issue privileged operations on the host OS. The authors describe their system, Proper, which was designed for use in the PlanetLab environment. They assume these privileged operations will not be in the critical path of execution for the issuing processes, so implementing as an RPC is OK. As an example, they describe Stork, a shared package management service that allows users to install a package once on a PlanetLab node, and then share its files among k slices without requiring k copies. Another example is an authentication service, whereby a single VM running sshd listens for connections and then uses a Proper hook in the VMM to fork a new VM to handle each incoming connection. Their API is transparent and easy for users to utilize, requiring a minimum of code changes to use Proper.

  A questioner asked how this was better than just using a distributed file system and ssh to facilitate communication between VMs, since poking a hole in the VMM presumably makes things a bit less secure. Steve answered that you could do that but performance would suffer. He argued that Proper's performance for that scenario is far superior, and that this is clearly a case of trading decreased security for increased performance, which is always a trade-off.

- ***AMP: Program Context-Specific Buffer Caching***

  Feng Zhou, Rob von Behren, and Eric Brewer, University of California, Berkeley

  The authors' goal was to create a buffer-caching scheme that performs better than LRU, since LRU can perform badly for pathologically large looping cases. Feng argued that databases and information retrieval tasks are susceptible to this looping condition. He described the design of AMP, a detection-based method for buffer cache management. On every system call or page fault, AMP calculates the current program context (program counter plus all return addresses on the call stack). Based on what happened in the past when the system was at this PC, AMP decides which cache management policy to employ. AMP divides the cache between different program contexts, and uses Randomized Cache Partition Management to adaptively control how much of the cache is devoted to each. Comparison to current solutions DEAR and PCC shows AMP performs better and can reduce the miss rate by 50% compared to LRU or ARC.

  A questioner asked if it wouldn t be simpler to define the PC as the program counter + the stackpointer, since that would presumably be different for each call. Feng agreed that would probably be much simpler than their implementation.

- ***Automatic Synthesis of Filters to Discard Buffer Overflow Attacks: A Step Towards Realizing Self-Healing Systems***

  Zhenkai Liang, R. Sekar, and Daniel C. DuVarney, Stony Brook University

  Zhenkai argued that buffer overflow attacks are the most common propagation method currently used by worms. Self-healing systems observe an attack and adapt so that future attacks will not succeed. He described their self-healing buffer overflow defense. The goal is to avoid server crash and acquire immunity to future attacks. The idea is to drop attack requests without denying benign requests. Zhenkai described how they put a defensive layer between the server process and the external network. They rely on third-party intrusion detectors to detect the first attack and trigger their logger. Based on the attack signature from the logger, their system learns so it can detect the attack signature next time, and let benign traffic pass through its filters. Their evaluation examined a known buffer overflow vulnerability in RedHat Linux, and their system generated effective filters for seven of the eight known attacks against it, without generating any false positives. This technique is effective against unknown attacks, with low overhead, and the filters generated can be shared among trusted friends.

SHORT PAPERS II

Summarized by Charles Gray

- ***Facilitating the Development of Soft Devices***

  Andrew Warfield, Steven Hand, Keir Fraser, and Tim Deegan, University of Cambridge Computer Laboratory

  Andrew Warfield presented the idea of using virtual machine monitors to prototype device extensions and experiment with new "in-

hardware" functionality without the need to modify hardware.

This work was done on the Xen VMM using multiple Linux instances with segregated devices. Devices are provided to multiple clients (Linux instances). A "tap," or filter module, can be interposed to modify the behavior of the device in software for experimental purposes.

Using a VMM, the authors claim, allows easier, safer, and more portable experimentation. The implementation demonstrates good disk performance, but there is high overhead on network performance.

The future work is to simulate cluster storage, provide virtual machine replay, and allow pervasive debugging on devices.

The source code to this is available in the Xen-unstable branch of revision control.

- ***Implementing Transparent Shared Memory on Clusters Using Virtual Machines***

  Matthew Chapman and Gernot Heiser, University of New South Wales and National ICT Australia

  Matthew Chapman points out that some tasks take more than one CPU to run. For example, weather forecasting is complex, and useless if the information is not timely.

  There are two existing solutions to this problem: SMP/NUMA (non-uniform memory access) machines, which are big, expensive, and provide a nice single-system image, and workstation clusters, which are well priced but require dealing with the network and multiple system images.

  This paper concerns getting shared memory on commodity clusters with commodity operating systems. The subject of this work, vNUMA, virtualizes an operating system across multiple nodes of a network. The operating system sees a large SMP machine with one large physical memory. This is provided by classical Distributed Shared Memory (DSM) algorithms.

  Benchmarks show that performance is very good for DSM-friendly workloads. Performance could be better for other workloads.

  Future work involves simulating simultaneous multi-threading while threads are stalled waiting for the network. It is also a goal to take advantage of IA64 memory ordering annotations for improved DSM performance.

- ***Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor***

  Ludmila Cherkasova and Rob Gardner, Hewlett-Packard Laboratories

  Lucy Cherkasova points out that VMMs are a software tool for building shared hardware infrastructure. Managing the shared resources is a critical task for a VMM. This work relates specifically to the Xen x86 VMM and the sharing of I/O resources.

  Xen used to have device drivers but now utilizes the device drivers of a privileged Linux instance—*Domain0*. Ultimately, it is desirable to have devices in their own domain due to bugs.

  This paper tackles the problem of how to charge resources to the I/O requester when all guest operating systems use the same driver. Before trying this, however, it is important to decide whether resource usage is large enough to care. The goal is to quantify CPU usage of I/O intensive workloads.

  To do this, the authors instrumented a machine monitor and tested Web and disk workloads. The authors found that increased I/O workload increases the *Domain0* CPU usage greatly. Interrupt processing is a dominating cost. The authors used an aspect of the Xen implementation (page flipping counters) to attribute work to clients.

  The authors concluded that I/O CPU usage is important and should be taken into account.

- ***Fast Transparent Migration for Virtual Machines***

  Michael Nelson, Beng-Hong Lim, and Greg Hutchins, VMware, Inc.

  The topic of this work was the VMWare system of moving a running VMM from one physical machine to another. The requirements of this system were:

  - The OS being migrated must be unmodified.
  - It should be transparent; that is, TCP connections, etc., should survive.
  - There should be no dependencies on the original machine once the migration is complete.

  There are a number of reasons to want to do this, including load balancing, hardware maintenance, and upgrading of the VMM software on the machine without interrupting the guest operating systems.

  One of the challenges of this work is to make sure the destination machine is compatible—for example, that it's the same CPU model (to avoid model-specific instructions), has its destination on the same network(s), and involves the same devices.

  The implementation uses a distributed SAN-based file system for disk transfer. To ensure decent performance the system needs to pre-allocate resources on the remote machine for the data transfer.

  Two important factors are robustness—virtual machines should not randomly crash—and minimal downtime. The latter is achieved by pre-copying as much memory as possible.

  The current implementation is the first to demonstrate transparent migration of VMMS with unmodified operating systems. Downtime for a migration is less than one second. A side effect of this work is

that VMM startup time became part of the critical path.

- ***Performance of Multithreaded Chip Multiprocessors and Implications for Operating System Design***

  Alexandra Fedorova, Harvard University and Sun Microsystems; Margo Seltzer, Harvard University; Christopher Small and Daniel Nussbaum, Sun Microsystems

  Sasha Fedorova started this presentation by asking two questions: What is multi-threaded chip multiprocessing (CMT)? Why does CMT need a new scheduler?

  Out-of-order-execution CPUs do not achieve great performance with database and similar workloads. There is a low cache hit rate, and most of the time (up to 80%) is spent blocked, waiting for the cache requests to be fulfilled.

  Using simultaneous multi-threading (SMT) you can fill these bubbles with instructions from a different thread. This is effective, with 5% more silicon yielding a 20–60% performance improvement.

  CMT is SMT with multiple CPUs on the same die. CPUs share the same L2 cache. Processor performance is very sensitive to L2 cache miss rate, even for CMT, but not so for L1 cache miss rate. This was demonstrated with graphs of benchmark results.

  The goal is to design a scheduling algorithm for L2-friendly scheduling of threads.

  This algorithm starts by classifying threads as L2 "greedy" or "frugal," and then tries to schedule many frugal threads with each greedy thread on a CPU. The challenge is in working out which threads are frugal and which are greedy, online, at runtime, with low overhead.

  A cache model and algorithm based on memory access is used to classify threads. This has been implemented and tested and demonstrates a throughput improvement of 27–45% by avoiding "thrashing" of the L2 cache.

- ***Hyper-Threading Aware Process Scheduling Heuristics***

  James R. Bulpin and Ian A. Pratt, University of Cambridge Computer Laboratory

  Simultaneous multi-threading (SMT) is fine-grained hardware multi-threading. Intel HT in Pentium processors provides heavyweight threads with some shared resources.

  The goal of this work is to optimize scheduling. So how do you measure performance to optimize for it?

  One measure is IPC (Instruction Per Cycle); however, using this metric biases highly parallel programs. A new metric is to compare HT vs. non-HT performance. The system performance is the sum of running ratios. This can be done online using Pentium 4 performance counters.

  Co-efficients can be learned with a training set of spec benchmarks. With some regression, runtime performance information can be used to correlate threads with training data. This correlation works OK in practice, but is not great. Future work will aim at improving this.

  The next problem is how to put this information into an existing scheduler. The aim of this work is not to create a new scheduler but to approximate gang scheduling. Other threads may run together due to a thread blocking, and this allows a runtime chance to see if there are better pairings available. This is implemented in Linux by modifying the "goodness" function that selects a thread's CPU affinity. Overall, some noticeable speed-ups were achieved.

  The authors concluded they can estimate a thread's behavior using performance monitoring, and they can improve performance with HT-aware scheduling.

INVITED TALKS

- ***NFSv4***

  Spencer Shepler, Sun Microsystems
  Summarized by Charles P. Wright

  In 1984, NFSv2 was presented at USENIX, and 10 years later NFSv3 was presented. Today, after another 10 years, NFSv4 was presented by Spencer Shepler, the document editor for the NFSv4 RFC and the leading engineer for Sun's NFSv4 group. Shepler presented a whirlwind tour of NFSv4.

  Shepler began by describing the history of NFS's protocol development. In 1985, the first version (NFSv2) was released, and it provided support for exporting basic POSIX 32-bit file systems. NFSv3, published in 1994, was limited in scope and solved two major problems well: (1) fields were extended to 64 bits for larger file systems, and (2) write performance was greatly improved through caching. NFSv3 is the most widely used distributed file system on UNIX/Linux LANs today.

  NFSv4 had a larger scope. To develop NFSv4, Sun went through the IETF to provide an openly defined standard. Other distributed file system protocols are either closed or not openly defined (many are only implemented). Shepler believes that being openly defined is one of NFSv4's greatest strengths.

  NFSv4 combines many existing protocols (e.g., portmap, mount, NFSv2 or v3, locking, and more) into a single NFSv4 protocol. In NFSv2 and v3 "traditional" RPC messages were used (e.g., getattr), but in NFSv4 there are only two RPCs: null and compound. The compound RPC in turn includes several RPCs, and evaluation stops after the first error. Compound RPCs reduce latency, because there are fewer round trips, and give the clients greater flexibility.

  A major addition to NFSv4 is state. Clients establish connections with

the server over a network protocol that supports congestion control (e.g., TCP), but does not necessarily guarantee in-order or reliable delivery. Each NFSv4 connection between a client and server may use multiple TCP connections (or tear all of them down when idle). Persistent state is managed with leases, which the client must renew. If a client's lease expires, then all of its state is discarded. This makes the locking protocol significantly more robust, as the server automatically frees locks for crashed clients (which required manual intervention in v2/v3).

Delegations allow the client to take full control of a given file. For example, if the server delegates a file to a client, then the client can open, read, write, and close the file an arbitrary number of times without contacting the server. Delegations are intended for environments with minimal sharing (e.g., home directories are often used by only one client), and can greatly improve performance (60% fewer messages are required for a Solaris build). When shared access is required, the server revokes delegations using callbacks.

There are a plethora of other new features described by Shepler. The IETF RFC makes it mandatory to implement Kerberos and SKPM/LIPKEY security, thereby bringing security to all NFS implementations. NFSv4 supports two types of file handles: (1) traditional persistent file handles, and (2) volatile file handles that are useful for file systems like FAT, which do not have unique object identifiers, or for user-level NFS servers. NFSv4 supports three classes of attributes: (1) eight mandatory attributes that are required for minimal NFS service, (2) 46 recommended attributes with defined semantics (e.g., standard POSIX attributes), and (3) arbitrary named attributes for use by applications.

Currently, there is no standard for NFSv4 performance measurement. Shepler thinks that it is unlikely the industry standard, SPEC-SFS, will be extended for NFSv4. Sun is currently developing a performance framework called FileBench that will support benchmarking standard POSIX system calls, NFSv4, and possibly CIFS using simple workload descriptions. FileBench currently has two sets of benchmark descriptions, "File Macro" and "File Micro," which describe several large benchmarks (e.g., database and Web server workloads), and micro-benchmarks (e.g., sequential read and write). FileBench is open source, and Sun will coordinate the community's development using Source Forge.

Shepler concluded with the state of implementations on Solaris, AIX, Netapp, Hummingbird, Linux, and BSD. In the future, the IETF working group will continue to interpret the current NFSv4 RFC (particularly ACLs) and will work on several new features. One particularly interesting project is pNFS, which will allow RAID-like striping across several filers in a portable and open way.

See http://blogs.sun.com/shepler for more information on NFSv4, the IETF working group, Solaris's implementation, and the slides from the presentation.

10-20x Faster Software Builds

John Osterhout, Electric Cloud, Inc.

Summarized by Robert Marmorstein

Motivated by frustration at waiting for large builds, John Osterhout has implemented a complete system for distributing the make process. Electric Cloud can be used on a large variety of platforms, including Windows, and uses virtualization to abstract away environment details. It can be used as a plug-in replacement for make with very little modification to the build system required. His system provides several advantages over distcc, but is commercial software. Using Electric Cloud, clients have seen speedups of as much as 20x for their build systems.

For projects with a large number of engineers and a significant code base, building can be painfully slow. Osterhout noted that, in one case, a client's from-scratch build process took about 70 hours to complete and that build times of more than 20 hours are not uncommon. Slow build times translate directly to loss of productivity. Osterhout estimates that, on average, slow builds mean a productivity loss of between 5% and 15% and a 5% to 10% delay in time to market. Slow builds also impact quality, because frustrated engineers are less able and willing to rebuild the code for testing.

Electric Cloud speeds up builds by distributing work across a cluster of machines with a central make server and a pool of build nodes. The utility uses virtual machines to provide a generic environment on the build nodes in the cluster, thereby greatly mitigating a significant challenge to the approach. Unfortunately, the other challenge, handling dependencies, is far more complicated to resolve.

Osterhout's approach is to "deduce dependencies on the fly" and recompile components when it becomes clear that they are in an inconsistent state. The tool does this by computing the sequence of file accesses that would occur in a sequential build and using a specially designed file system to detect out-of-sequence file accesses in the parallel build process.

The solution is cost-effective and scalable, but does require some overhead for network communication. The system now uses P2P and just-in-time compression, which greatly boosts effective bandwidth. In order to properly handle recursive makes, some minor edits to makefiles may be required to achieve full performance. Unlike distcc, Electric Cloud preserves the

original log output of the build system.

The system is managed through a Web interface and provides a priority system with classes of users and classes of builds. It is robust and has demonstrated build system speedups of 10x to 16x on a set of benchmarks that include building MySQL and the GTK libraries.

For more information, see http://www.electric-cloud.com or email info@electric-cloud.com.

- ***Enhancing Network Security Through Competitive Cyber Exercises***

Colonel Daniel Ragsdale, United States Military Academy

Summarized by Christian Kreibich

Very excited to be with the USENIX crowd, Dan Ragsdale presented the past and present of the annual Cyber Defense Exercise (CDX), a competition among the US Service Academies which aims to increase the technical knowledge required to defend computer infrastructures and to raise awareness of the importance of handling digital information cautiously; at the same time, CDX improves the participants' soft skills. Dan explained the military's enthusiasm for CDX by presenting the armed forces as a highly information-driven but therefore also information-dependent organization. According to Dan, nobody is going to threaten the US military in direct armed conflict in the near future, but asymmetric warfare will likely take place in restricted niches, which include the digital networks. Indeed, Dan pointed out that the military now considers cybersecurity a "Force Protection" issue, since failure to protect classified information in cyberspace would immediately put at risk the individuals serving in the Army.

To underline this point, Dan noted that around 80% of military digital communications cross U.S. borders, and while dedicated networks do exist, much of that traffic is encrypted and tunneled through the public Internet. He pointed out the need for increasing awareness of the significance of digital information, mentioning accidental leakages—for example, the fact that several security-relevant details about Baghdad's International Zone (the diplomatic/government district) can be found on the Internet.

Dan then presented the CDX in more detail. The participants are divided into three groups: the Red forces (the penetration team, led by security experts from the Assurance Directorate of the NSA), the Blue forces (the various Service Academies teams which design, implement, manage, and defend their networks), and the White cell (the referees, staffed by individuals from CMU's CERT and co-located with all teams). The scoring is based on maintaining required functionality, successfully defending against intrusions, and reporting suspicious activity. Successful intrusions result in penalties.

The first CDX was held in April 2001, around four years after the idea was first discussed. Over time, the rules of the game changed to reflect more realistic attack scenarios. Examples include permission for limited denial of service attacks, increasingly refined anomalies that present selective equipment failures, social engineering, and limited attacks on the competitors.

In the four years the CDX has been held, education, leadership development, and research opportunities have proven to be the key benefits of the program. Other lessons learned include the knowledge of how to fund, organize, and operate a cybersecurity exercise. A key element for cost savings is virtualization: Virtual topologies save on hardware costs while providing flexible and reproducible training environments for the students. Since the establishment of the exercise, multiple spin-offs have emerged, such as the IA Defense Exercise and Collegiate Cyber Defense Competition (CCDC).

Questions touched on a wide range of topics. One attendee asked how ethical considerations are incorporated into the program. Dan explained that students receive a detailed legal briefing during preparation for the exercise. Another attendee suggested that more proactive training, including attacking skills, would be preferable to focusing on the defense. Dan explained that the students need to understand the attacks anyway in order to be able to defend against them. When asked how far the rules are going to be loosened in the future, Dan explained that there will be increasing leeway; however, some fraction of the deployed software will continue to be required to be made by Microsoft, since that is what the students are going to be working with in the field. When asked how good the Red team is, Dan explained that the team is staffed by experts from the NSA who perform just as one would expect to see in the wild.

Further information about the CDX is available at http://www.itoc.usma.edu/cdx.

- ***Under the Hood: Open Source Business Models in Context***

Stephen R. Walli

Summarized by Peter H. Salus

Stephe Walli, sometime USENIX standards rep, founder of Softway, etc., etc., gave a wide-ranging and well-organized talk on the nature of open source business models.

Walli pointed out that businesses have to offer a value proposition to customers (he sounded a lot like "Doc" Searls at times) and that, despite the gossip, OSS isn't "new" or "special." It is a particular sort of business, growing up around a specific community. But it is just this that makes our community subject to the FUD of larger enterprises.

Nonetheless, "economics works," and economic exchanges are

human social behaviors in the marketplace (Searls again, Eric Raymond, and Bob Young all came to mind).

Participation, Walli said, is asymmetric: You get back more than you give—think of 1,000 folks each contributing to Apache or GNOME and what the bundle they receive for that contribution is.

In fact, it's not really about altruism—people value their skill sets in different ways in different contexts.

Walli hopes we'll see that these phenomena work for companies, not merely individuals (here, the current activities of IBM come to mind—if IBM is successful where Linux is concerned, many other companies will emulate their activities, to the benefit of both the corporations and the users).

Walli's slides are downloadable from http://www.usenix.org/events/usenix05/tech/slides/walli.pdf.

- ***MacOS X Tiger: What's New for UNIX Users?***

Dave Zarzycki, BSD Technology Group, Apple Computer

Summarized by Rik Farrow

Zarzycki provided lots of interesting new details about Tiger, the most recent release of MacOS X, while managing to enrage a good portion of his audience. More than anything, the difference in perspective between Apple OS developers and the UNIX community became glaringly obvious.

Zarzycki started out on safe ground. He described some user-level features that can be employed by programmers writing specifically for MacOS: searching from within any program (Spotlight), new scripting interface, voiceover support (vision-disabled users can have any text read aloud), and a new Dashboard interface for some cool applications. There are new codecs for iChat and Quicktime, and 64-bit support is now included in Lib system (similar to libc in other UNIXen).

The operating system has better support for SMP, with finer-grained locks. The earlier versions of MacOS have just two big locks, one for network operations and one for everything else.

The Apple file system has also been improved, with built-in defragmentation for HFS. Zarzycki said that extended file attributes have been added to HFS which support ACLs similar to those found in Windows NT and later—a POSIX superset. Support has also been added for extensions that are like the old Mac resource forks. It was at this point that Zarzycki started getting into trouble. If you want to copy files AND their extensions, you must use special options to UNIX commands, including cp, mv, tar, and rsync. In other words, if you don't know about the new options or remember to use them, you lose the extensions.

Zarzycki also described improvements to networking, including IPsec support for certificate-based authentication, IPv6 firewall, ipfw2 logging, Wide Area Bonjour (formerly Rendezvous) support, and Ethernet bonding/failover. Userland includes updated versions of Perl, Python, and Ruby. The Apple System Logger (ASL) stores log messages in a database that can be searched and pruned, and is much more powerful than syslog.

Finally, Zarzycki announced launchd, a new superdaemon. Launchd would take the place not only of inetd and cron but also (apparently) of init itself, and of mach_init, running as process ID 1. Launchd can run background processes on behalf of users, including starting network services. Launchd makes it much easier to start and manage daemons, helps reduce system load, allows parallelization during the boot process, supports messaging, and uses an XML configuration file (similar to the plist syntax already in MacOS X). There is both a command line interface (launchdctl) and a GUI interface.

I personally found the notion of having launchd making it easier for my users to launch network services terrifying, and so did some others. Locking down network services is a first step in securing a system, and this appears to unlock it. Someone asked the reason for choosing XML for the configuration language, and the answer was that it is industry-standard and human-readable, a response which evoked some laughter.

Another person asked what happened to dump/restore? You have the only workstation that cannot be backed up out of the box? The Apple answer was to use Apple System Restore (ASR), but the questioner pointed out that ASR's syntax differs from dump's, and that ASR only does level zero (complete) system dumps and cannot do incremental backups. Zarcycki really stepped in it by saying, "If we had to deal with muscle memory, we would be stuck back at point one." This arrogant response really got some of the audience riled up.

I came away from this talk having learned several things: There were some really cool new features in Tiger, and Apple really isn't interested in creating an OS that can be treated like a UNIX system.

You can find some info about MacOS X Tiger here: http://www.apple.com/macosx/features/unix/.

WORK-IN-PROGRESS REPORTS (WIPS)

Summarized by Robert Marmorstein

- ***Collecting Long-Term Storage Failures***

Mary Baker, LOCKSS Team, HP Labs

Because RAID and tape backups degrade over time, for a variety of reasons, long-term data preservation requires new techniques. In order to build a model of long-term storage failures, Baker proposes gathering information about disk faults using quantitative studies, case studies, and even anecdotal evidence. Some of the information

gathered may be confidential, but much will be released to the public.

- ***Power-Aware RAID***

  Charles Weddle, Florida State University

  Although the disk is a major source of energy consumption, RAID implementations are not friendly to power-saving techniques. Weddle is investigating means to take advantage of the cyclic behavior of loads to increase power efficiency while maintaining performance and reliability. PARAID uses a "skewed striping pattern to save power without degrading performance" via a bi-modal distribution of busy and idle drives. A prototype saves 15% power with a 1% performance hit when compared with RAID-0, but modeling higher levels of RAID is the next milestone. For details, go to http://www.cs.fsu.edu/~weddle/paraid/.

- ***SchedMark: Evaluating Scheduler Performance***

  Eitan Frachtenberg, Los Alamos National Laboratory

  There are a plethora of job schedulers, each of which has different characteristics and advantages in different scenarios. Eitan has begun work on a benchmark suite that will fairly evaluate different schedulers. Instead of assigning them a single number, the benchmark will provide a set of metrics that represent different workload types. For more information, see http://www.cs.huji.ac.il/~etcs/pubs/talks/frachtenberg05:schedmark-wip.html.

- ***Linux in Hollywood/ CinePaint***

  Robin Rowe

  Rowe presented a double header. First he described the current status of Linux advocacy in Hollywood. Then he described the status and history of CinePaint. Linux has replaced Irix as the industry standard OS for special effects, prompted largely by its successful use in *Titanic*. Because it is far more scalable than other solutions and provides better graphics driver performance, it is a perfect match for the movie industry. More information on Linux in the movies can be found at http://linuxmovies.org.

  After Adobe discontinued Photoshop on SGI, Hollywood sponsored development of Gimp, but the work fizzled. The tool later reemerged to become CinePaint, which Rowe maintains on SourceForge. The development team is exploring a new architecture based on a shared-memory image buffer that will allow multiple applications to access the same image. The Web site for CinePaint is http://cinepaint.org.

- ***Stream-Oriented Scalable Cluster Architecture***

  Tassos S. Argyros and David R. Cheriton, Stanford University

  Tassos presented ideas for scaling clusters to large numbers using a stream model instead of a model based on requests. The basic premise is that streams can avoid hotspots caused by lots of nodes requesting data from a single critical node. Instead of pulling the data with requests, he advocates a framework for pushing the data with streams. For details see http://www.stanford.edu/~argyros/argyrosStreams.pdf.

- ***Grisp***

  Peer Stritzinger

  Grisp is a new programming language that is almost ready for implementation in a real compiler. It will have two flavors: a static, compiled flavor and a dynamic, interpreted flavor. The language will be feature-rich and C-like enough to appeal to most programmers, but will provide a cleaner, easier way to "get at the bits of the machine." It will also have built-in parallelism. Details can be found at http://www.grisp.org.

- ***Testbeds for Exploring Wireless Networking***

  Kirk Webb, David Johnson, Daniel Flickinger, Tim Stack, Leigh Stoller, Robert Ricci, Mark Minor, and Jay Lepreau, University of Utah

  The Emulab group has produced three new testbeds for exploring wireless networking: a building-scale testbed, a fixed sensor net, and a mobile WiFi + sensor net. All three testbeds can be accessed remotely as a platform for testing wireless applications. The building-scale testbed supports the OS of your choice and consists of 27 PCs located throughout the building. The fixed sensor network consists of 25 Mica2 motes using TinyOS. The mobile testbed carries XScale or Mica2 motes mounted with vision tracking and an odometer. Motion of the nodes is scriptable or interactive in real time. More information can be found at http://www.emulab.net/.

- ***Pre-Virtualization and Hypervisor Diversity***

  Ben Leslie, Josh LeVassuer, and Volkmar Uhlig, Karlsruhe University

  Because para-virtualization requires invasive changes to the guest OS, Ben has implemented a system that inserts calls to the virtualization system at the assembly level. This provides virtualization on an unmodified operating system while still allowing hypervisor diversity.

- ***Digital Preservation***

  David Rosenthal, LOCKSS

  Because file formats may become obsolete, data preservation efforts need some way to convert old formats to modern formats. Storing converted formats is costly, so a mechanism for triggering conversion on the fly is necessary. Fortunately, content negotiation in HTTP can solve this problem by requiring the server to reject requests for old formats and converting images to new formats. For more information, see http://

www.dlib.org/dlib/january05/rosenthal/01rosenthal.html.

- ***Clusters in Suitcases***

  Mitch Williams, Sandia National Labs

  Mitch showed lots of pictures both of miniclusters and larger clusters. The miniclusters use LinuxBIOS and BProc, which gives them very rapid boot times. The larger clusters use Infiniband. Current work focuses on accelerating the development of an open source Infiniband software stack for high-performance computing. For details see http://eri.ca.sandia.gov.

GENERAL TRACK

SPEEDING UP THINGS

- ***A Portable Kernel Abstraction for Low-Overhead Ephemeral Mapping Management***

  Ben Leslie, Josh LeVassuer, Volkmar Uhlig, Karlsruhe University

  Khaled Elmeleegy opened the presentation by explaining what an ephemeral mapping is and where the ephemeral mappings are used in OS. He elaborated two examples of ephemeral mapping: writing to a pipe, and reading from a pipe. These applications involve multiple virtual memory pages and use ephemeral mappings to eliminate the copy operation by the writer. Next, Elmeleegy presented the motivation of the work—the increasing cost of modification to the virtual-to-physical mapping—and showed how expensive the TLB invalidation instructions are from 486 to Pentium 4.

  To solve the problem the authors proposed an sf_buf interface, which consists of four functions. In the five implementations for different architectures, Elmeleegy introduced their AMD64 implementation, optimization for TLB invalidation and shared private mapping. Later, he evaluated the performance on i386 platforms using several applications. On a Xeon box with a 2.4GHz processor, using FreeBSD5.3 and Apache 2.0.50 with 30 clients, Elmeleegy showed the reduction of local and remote TLB invalidations, which leads to about 7% throughput improvement. Using the Postmark benchmark, the throughput improvement is about 4–12%. Using disk dump, the private mapping optimization was effective and achieved up to 30% bandwidth improvement. Finally, Elmeleegy concluded that combining virtual address allocation and mapping creation is beneficial for both low overhead and portability, that AMD64 implementation is virtually free, and that i386 implementation is effective.

  In the Q&A, Vivek Pai from Princeton University pointed out that the interface was originally implemented by his student and the effectiveness of mapping reduction was discussed in their USENIX '04 paper as well. Elmeleegy answered that they did cite that work.

- ***Adaptive Main Memory Compression***

  Irina Chihaia Tuduce and Thomas Gross, ETH Zürich

  Irina Chihaia began her presentation by demonstrating the increasing CPU-memory gap among DRAM, disk, and CPU speed; applications require more and more memory, and those that exceed the DRAM size run at disk speed. The authors' proposed solution to this problem is to store memory pages in compressed form to avoid disk accesses. Next, Chihaia showed the common and compressed memory hierarchy and explained compression tradeoffs.

  Chihaia discussed prior work on compression and design issues, in which the size of the compressed area is the key. The authors' design divides the compressed area into independent zones. Chihaia illustrated how to store compressed data and swap to disk and introduced their implementation for the Linux kernel module. In evaluating their approach, Chihaia showed the results of a Symbolic Model Verifier(SMV), NS2 simulator, and qsim, a car simulator. The SMV results exhibit lower slowdown with compression, and the simulators have a speedup of 1.04–55.76 with compression.

  When asked about faster strategies to get reference information, which is required by some adaptive schemes, Chihaia answered that looking up metadata may still have high overhead and that approaches may be application-dependent. When asked about the effects on multi-threading platforms, she commented that it may have different impacts on applications using compression and that one needs to model specifically to find out the results.

- ***Drive-Thru: Fast, Accurate Evaluation of Storage Power Management***

  Daniel Peek and Jason Flinn, University of Michigan

  Evaluations of storage power management need to be fast, accurate, and portable. Daniel Peek compared three approaches to evaluating power management—trace reply, trace reply without delay, and reply with simulator—before presenting Drive-Thru, which uses a hybrid methodology. Drive-Thru's innovation is to separate time-dependent and time-independent activities, allowing reply time-independent behavior without idle time and simulating time-dependent behavior. He illustrated their design of merging replay and simulation, and the Drive-Thru operations.

  Peek validated their design with the ext2 file system and a network file system (the Blue file system) with 802.11b power management. Drive-Thru has an average prediction error within 0.21%–3% on the ext2, and 7% on the network file system. In the case study, Drive-Thru can complete evaluation over 40,000x faster than trace reply on local storage, and over 13,000x

faster on network storage. Peek also showed their improvement of system energy cost for the BlueFS by flushing on spindown and reducing writeback delay. After presenting related work, Peek concluded that Drive-Thru is fast, accurate, and portable.

When asked about the effects of other policies, such as delaying the flush further, Peek commented that they could evaluate this. To a question about accuracy when there's not enough data, Peek answered that some situations are common in file-system trace and not particular in their study. Another questioner wondered about consistency over time and the cause of the noise. Peek said they didn't know and may need to study it further.

LARGE SYSTEMS

Summarized by Andrew Warfield

- ***Itanium — A System Implementor's Tale***

  Charles Gray, University of New South Wales; Matthew Chapman, Peter Chubb, and Gernot Heiser, University of New South Wales and National ICT Australia; David Mosberger-Tang, Hewlett-Packard Labs

  Awarded Best Student Paper

  The presentation described the authors' experiences in building OS support for Intel's Itanium processor. The Itanium presents several considerable architectural changes relative to the x86, and the presentation described a set of clever tricks that were used to achieve good performance.

  The presentation began with a discussion of the Itanium's virtual memory support and of issues regarding the explicitly parallel instruction-set computing (EPIC). EPIC allows instruction parallelism to be provided by the compiler (or assembly programmer) by placing multiple instructions within a very large instruction word (VLIW). They discussed the implications of both VLIW and the huge number of processor registers (128 Integer and 128 float) on performance, with examples of instruction barriers and performance improvements for OS signal delivery.

One of the Best Student Paper authors accepts their award from General Track Program Chair Vivek Pai.

Much of the remainder of the talk was spent discussing the consequences of the escalate privilege (EPW) instruction, which allows fast entry for kernel operations. They explained how EPW has allowed them to reduce IPC time in the L4 micro-kernel from over 500 cycles down to about 30. They theorized that they should be able to get it down to nine cycles. The presentation included several war stories about attempts to reduce IPC time, and the difficulties of using the Itanium's performance counters to figure out where cycles were being gained and lost.

Jonathan Shapiro asked how likely it is that compilers would be able to realize the speedups that the authors described automatically. The presenter replied that many of their optimizations could be codified for a compiler. Another person asked how the optimizations worked for more complex system calls, given that the examples had been for getpid(). The speaker said that they still applied, as optimizations were for entering and exiting privileged execution and not keyed to any specific call.

- ***Providing Dynamic Update in an Operating System***

  Andrew Baumann and Gernot Heiser, University of New South Wales and National ICT Australia; Jonathan Appavoo, Dilma Da Silva, Orran Krieger, and Robert W. Wisniewski, IBM T.J. Watson Research Center; Jeremy Kerr, IBM Linux Technology Center

  Andrew Baumann presented this paper about patching a running kernel without downtime. The implementation he described was on IBM's K42 kernel, which is an object-oriented operating system written in C++.

  Andrew described a scheme in which subsections of the kernel could be replaced at runtime. The approach involved replacing the class within the kernel with a new version and then translating all existing instances to it. The talk described K42's hot-swap support, as well as the extra required mechanisms, including using the factory pattern to track outstanding state and transfer functions to update state to a new version. He additionally described read copy update (RCU) techniques used to ensure a safe point of upgrade, a translation table to redirect invocations to the new object version, and a versioning scheme to track versions in the system.

  Three sample upgrades were described: a new kernel API for a partitioned memory region, a fix for a memory allocator race condition, and a file cache manager optimization for the unmap page operation.

  Someone asked how complex the changes in the examples were. Andrew explained that in each of the examples only a single class was upgraded, but that they could potentially provide support for more complicated upgrades. A second questioner asked how they handled tasks that were sleeping on code that was being upgraded. Andrew pointed out that K42 is completely event-driven, using task

callbacks, and so the sleep situation is not a concern.

- ***SARC: Sequential Prefetching in an Adaptive Replacement Cache***

  Binny S. Gill and Dharmendra S. Modha, IBM Almaden Research Center

  This animated presentation described extensions to the ARC cache management work to integrate sequential prefetching with cache replacement. The presenter began with a long discussion of the importance of caching given the increasing rift between processor and disk performance, likening the cache to a refrigerator, a disk to a grocery store, and a remote procedure call to himself.

  He explained that large commercial systems make almost exclusive use of sequential prefetching, and then described asynchronous prefetching, where prefetches are issued in overlaid windows to improve performance. SARC works by adaptively trading off space at the bottom of the random and sequential cache lists based on the relative marginal utility. He showed a simulation to illustrate the point.

  Someone asked how aggressive the prefetching window needed to be. The presenter explained that the use of adaptive, asynchronous prefetching meant that they did not need to scale the prefetch window size, but that the approach would scale to any workload.

IMPROVING OS COMPONENTS

Summarized by Charles Gray

- ***SLINKY: Static Linking Reloaded***

  Christian Collberg, John H. Hartman, Sridivya Babu, and Sharath K. Udupa, University of Arizona

  The goal of this work is to replace dynamic linking with static linking. Why would you want to do this, since dynamic linking saves space by sharing pages?

  SLINKY, consisting of tools plus kernel modifications, allows static linking of binaries while providing the space advantages of dynamic linking. It has fewer than 2000 lines of code.

  One of the problems of dynamic linking is unmet dependencies at runtime that may result from an erroneous installer or possibly a library upgrade that was not fully compatible.

  Dynamic linking has benefits but can lead to "DLL Hell," causing programs to just stop working. Moreover, this problem is not confined to Windows. Static linking solves it by resolving everything at compile time. The distributed binary is exactly what the developer built.

  There is no free lunch, however; you still need to relink for a library upgrade, and there is no disk sharing. Statically linking 300MB of binaries can blow out to 3GB!

  Dynamic library updates can also cause bugs resulting from developers deliberately or accidentally relying on outdated bug fixes or workarounds in libraries that cease to exist after an upgrade. The authors assert that it should be the programmer's job to test new libraries, not the user's.

  The SLINKY system was constructed to solve the problems with static linking. It allows the system to share common executable file content (the multiply linked libraries) by hashing pages of binaries. This requires that all binaries are linked with position independent code (PIC) to ensure that addresses in the binaries are not a problem for sharing.

  The slink command creates a static binary from a dynamic executable and its libraries. A digest program sorts out shared pages, and an in-kernel module is used to share pages at fault time. The system also provides client-server distribution of pages of files over the network.

  The result is page sharing with no significant performance impact. The current "chunking" program to share pages is a generic algorithm and has a 20% space overhead over dynamic executables. An ELF-aware algorithm should be able to eliminate this overhead.

  Another feature of SLINKY is that some pages can be "blacklisted" so that they will not execute. This can prevent buggy or insecure pages from ever executing, since complete relinking is required for an upgrade, and this shouldn't be done by users.

- ***CLOCK-Pro: An Effective Improvement of the CLOCK Replacement***

  Song Jiang, Los Alamos National Laboratory; Feng Chen and Xiaodong Zhang, College of William and Mary

  This work presents a new VM replacement algorithm. The authors feel that replacement algorithms can be improved to deal better with a broader range of configurations and workloads.

  As background, the clock algorithm is an efficient approximation of LRU. In a replacement algorithm, recency of access is important. Reuse distance should also be considered, however. That is, how close are two consecutive accesses to a page?

  The computation cost of any replacement algorithm must be proportional to page faults, not to individual memory accesses. This is why LRU is too expensive and is simulated with clock.

  Some workloads are LRU-friendly (reuse distance < memory size), and some are unfriendly (reuse distance > memory size).

  There has been some work in this area. Linux uses a two-queue-like clock. It maintains active and inactive lists, and protects some pages from eviction. IBM has a "CAR" algorithm which maintains "hot" and "cold" queues.

  CLOCK-Pro is based on the authors' previous work, LIRS. LIRS took advantage of reuse distance knowledge but required per-memory-access knowledge.

CLOCK-Pro aims to improve this.

CLOCK-Pro essentially makes a clock with three pointers: "hot," "cold," and "test." The authors demonstrated how a state machine is used to move pages between various states and to govern when pages are evicted.

Performance was tested with a simulation and a prototype. Overall, CLOCK-Pro gives good results in the simulation on a broad spectrum of workloads. The authors claim this is also shown in the prototype.

CLOCK-Pro is a low-cost eviction algorithm that has advantages over the clock algorithm. It supports a broad spectrum of workloads, and a prototype implementation has been tested with gnuplot, gzip, and "gap" workloads.

- ***Group Ratio Round-Robin: O(1) Proportional Share Scheduling for Uniprocessor and Multiprocessor Systems***

  Bogdan Caprita, Wong Chun Chan, Jason Nieh, Clifford Stein, and Haoqiang Zheng, Columbia University

  The goal of this work is to provide O(1) proportional fair-share scheduling for UP and MP systems. Proportional scheduling is useful and predictable. You can characterize the fairness of an algorithm by measuring the actual time received by all threads and comparing that to their allocated proportions.

  There have been various round-robin schemes proposed; these have been fast but unfair. Fairer attempts based on "virtual time" have also been tried and have been fair but slow. Both these sets were uniprocessor only, but multiprocessor is more common now.

  For MP scheduling there is Surplus Fair Scheduling, but this provides no guarantees and is slow.

  This work proposes GR3—the first proportional fair scheduler with fixed overhead and fixed error bounds for UP and MP systems.

  Put simply, the algorithm groups clients based on their weights, to help simplify the scheduler. Groups are based on exponentially increasing intervals of weights. This provides a constant small number of groups and ensures that all clients in a group have weights within a factor of two. Within a group a round-robin-like scheduler is used that adjusts for the variations in weights.

  Extensive simulations on GR3 were run and the results were good for both single- and multi-processors.

INVITED TALKS

- ***Linux and JPL's Mars Exploration Rover Project: Earth-Based Planning, Simulation, and Really Remote Scheduling***

  Scott Maxwell and Frank Hartman, NASA JPL

  Summarized by Nikitas Liogkas

  NASA/JPL's Mars Exploration Rover project is the first time Linux systems are being used for critical mission operations. In this invited talk, Scott Maxwell and Frank Hartman, both of them rover drivers, discussed the software they developed and their experiences driving the two rovers, Spirit and Opportunity, across the Martian terrain.

  Scott Maxwell gave an overview of the Mars Exploration Rover mission. The two rovers are twelve light minutes away from Earth, which means that it takes a while for a command to reach Mars. Consequently, much planning and simulation of the Martian environment has to take place before any actual actions are performed. Scott then described the rovers' Mobility subsystem. They can achieve a top speed of around two meters per minute, and their software runs on a 20MHz CPU. They also carry a device called the IMU, which tracks their altitude but is not a full-fledged compass. Writing software for moving the rovers around involves developing a program that they will follow throughout the Sol (the Martian day). There are three types of commands: high-level, for which the rover itself does all the "hard work" of calculating the exact motor commands to be issued; mid-level, where arcs and point turns are specified; and low-level, which move the wheels directly. Low-level commands are rarely used except for straightening the wheels at the end of each day (so that if the wheels freeze during the night, they won't lose the ability to move the rover altogether!). Scott then briefly talked about the hazard avoidance mechanisms in place, which include an on-board navigation map. He noted, however, that these are quite expensive in terms of computing power and so are not used much.

  He described a typical drive, which consists of two major parts: first, the rover is instructed to blindly follow a certain path, during which it can reach its top speed; in this way it can traverse up to 100 meters per Sol. From the point of view of the rover, this is like traversing a dark room with a flash bulb! Eventually, it reaches its goal position, where it slows down and starts taking small steps in order to take interesting pictures or otherwise perform data collecting and scientific experiments.

  Scott gave an overview of the rovers' Imaging subsystem. There are nine cameras, eight of which are used for driving, in four stereo pairs. All the cameras have a one-megapixel resolution. A wavelet-based, lossy compression scheme is used to reduce the amount of data needed to be sent back to Earth. The IDD (Instrument Deployment Device) subsystem consists of the robotic arms the rovers are equipped with. Each of these arms is loosely modeled on the human arm, with five joints, including a double-jointed elbow that can move up and down. Four tools are mounted, arguably the most interesting of which is the

RAT (Rock Abrasion Tool). All tools must be stowed when driving to avoid damage to the rover. Scott told a nice story involving a rock named Humphrey, and how they created the first RAT mosaic on Mars, consisting of three tightly spaced circles, which they call "Mickey Mars."

Frank Hartman then provided an overview of the Rover Sequencing and Visualization Program (RSVP). A full Sol's worth of activities are planned, involving a number of different software tools, collectively called RSVP. ROSE, a Java-based command editor, specifies which commands are going to be executed when. HyperDrive, a C++ visualization component, helps in the task of visualizing the environment the rover is moving in. Other tools include OpenGL Performer and GTK+/Libglade on Linux. In order to build the terrain meshes, they use a stereo correlation process, and then they generate a synthetic 3-D terrain mesh from the output of that process. They are also able to generate a Digital Elevation Model (a 2.5-D model), which can be done much faster than the 3-D one. Frank talked about additional features, such as on-board collision detection, terrain analysis (highlighting dangerous areas), shadow generation during simulation, and image sequencing. He concluded by showing us some awesome views of the surface of Mars and of the rovers performing scientific experiments.

During the Q&A, there was great interest in finding out more details about the project. Replying to several of the questions, the two scientists described the testing room they are using to simulate the Martian surface, which they are utilizing as a realistic testbed. They described how they indicate sites of interest by gluing coins to the fake rocks in the testing room. They talked about the role of open source software at JPL, and noted that the first use of Linux in a deep space mission has been surprisingly successful. They indicated that their greatest source of frustration so far has been a closed-source enabling component of the system. All in all, the talk was very inspiring.

More information about the mission can be found at http://marsrovers.jpl.nasa.gov/overview. The talk is available online in mp3 format at http://www.usenix.org/events/usenix05/tech/mp3/maxwell.mp3.

Possible Futures for Software

Vernor Vinge

Summarized by Nikitas Liogkas

No one knows what software technology will be in the future. In this invited talk, Vernor Vinge, an award-winning science fiction author, presented four different scenarios for the future of software, which are based on different values for enabling drivers, mainly hardware improvements.

He first made sure to indicate that this was not a survey of future programming languages but, rather, a guess about what software will look like in the future. He started off by describing a new style of futurology that is becoming increasingly popular. This style dictates that we should not be looking at a single vision for the future but, rather, explore different possibilities based on important parameters; he called this technique scenario-based planning. In this way, we can mark out the extremes of the probability space, and we can then watch for symptoms indicating that one of our scenarios is coming true.

Three out of four scenarios he presented are based on what will happen with Moore's Law. Although Moore himself has admitted that his law cannot hold forever (see http://www.techworld.com/opsys/news/index.cfm?NewsID=3477), Vernor thinks that everything is possible. According to his first scenario (The Machines Stop), the hardware's current exponential growth can end abruptly with a catastrophic collapse. This may be caused by a fundamental problem at the physical layer, where semiconductor technology constitutes a single failure point. We are using the same microcontrollers to solve more and more problems, and at some point a disaster may ensue with catastrophic results. He vividly described the psychological blind spot people sometimes get, where they are willing to go much further into danger when the risk of a highly rewarding behavior is not precisely quantified. He named some science fiction novels related to this scenario, and even proposed a solution: building some machines that do not depend on semiconductor technology.

The second scenario (Legacy Software Forever) assumes that the exponential growth saturates at some point, and we end up with a situation where some types of hardware continue forward as others level off. He described a future where we have loads of legacy software, and, when faced with a new programming project, there is always a difficult choice between software archeology and reinventing the wheel.

According to the third scenario (Technological Singularity), exponential growth continues without saturation, and we eventually reach a point of "singularity," a situation where the rules change profoundly. In such a situation, AI has succeeded, and the physical world is awake. He described how pre- and post-singularity software might look, mentioning biological models for problem-solving and the future of software engineering and software verification. He also pointed us to a page on Accenture's site that eloquently describes this scenario from a company's point of view (see http://accenture.co.uk/xd/xd.asp?it=enweb&xd=services%5Ctechnology%5Cvision%5Ctech_vision.xml). He noted that some new problems may be revealed when systems scale up to millions of pro-

cessors, similar to the new problems that showed up in mathematics when computers first appeared.

The last scenario (Ubiquitous Law Enforcement) is the most Orwellian, in that it conjectures that a certain area on every chip may contain government-owned/law-enforcing logic, and that all interactions with the outside world will be filtered through that logic. In that way the police could monitor all electronic transactions, and thus there would be no real privacy.

During Q&A, Andrew Tanenbaum noted that past science fiction was not able to predict the advance of computers, cell phones, or even the Internet, and that he sees no reason why science fiction should have improved in the meantime. Vernor surprisingly indicated that the development of computers and the Internet were indeed predicted back in 1946 by Murray Leinster in his book called *Logic Named Joe*.

His slides are available at http://www-rohan.sdsu.edu/faculty/vinge/misc/u05; the talk itself is available in mp3 format, at http://www.usenix.org/events/usenix05/tech/mp3/vinge.mp3.

- ***Flying Linux***

Dan Klein, USENIX

Summarized by Nikitas Liogkas

In this invited talk, Dan Klein attempted to evaluate the feasibility of Linux for running "fly-by-wire" software. He noted that Linux may be "better" than Windows, but when your life is at stake, your attitudes change considerably: is it better enough? He talked about what it takes to make software truly mission critical, and whether Linux meets those criteria.

Dan started the talk by looking back at the earliest fly-by-wire systems, such as Mercury, Gemini (which utilized computer-based control), and Apollo (computer-guided, with a much better GUI). He explained that, whereas for old aircrafts, there is a one-to-one linkage between pilot actions and aircraft responses, in a DFBW (digital fly-by-wire) system a pilot action expresses intent, and the system has the responsibility of translating this intent into an appropriate response. He noted that unstable aircrafts (such as the F117A, which is unstable in all three axes) cannot be flown without such computer aid.

He then went on to explain how contemporary DFBW systems are built. Most of them are triple-redundant, utilizing voting systems. Also, all of them are purpose-built systems, designed to do one thing and do it well. Surprisingly enough, cars are becoming "drive-by-wire" as well, utilizing such advanced systems as ABS and traction control. He mentioned the new experimental Mercedes Benz F200, which has no steering wheel, just a yoke for navigation!

He also noted that proprioception, the unconscious perception of movement and spatial orientation arising from stimuli within the human body itself, gives rise to short-loop pilot reflexes. It is exactly these reflexes that fly-by-wire systems try to accommodate.

He then embarked on a long discussion of various flight accidents and the reasons behind them. Computer applications crash because not every possible case can be tested. Testing is usually limited, performed for normal operating conditions and for some extreme cases. He illustrated how various bugs exist in flight software, most of which, fortunately, are found during testing. He also mentioned some interesting problems with flight software that people do not normally think about: time-zone handling, leap years, and the change to the Gregorian calendar. He also told us the "Gimli glider" story, where an Air Canada aircraft ran out of fuel, due to a bad fuel sensor, and had to divert to Winnipeg, and then to Gimli, where it landed.

Klein argued that there is no room for error in these kinds of systems. Error-prone code or untrusted loadable modules that might crash the system are strictly forbidden. He looked at diverse issues, such as bugs, boundary conditions, and compartmentalization. He concluded by saying that Linux is not yet ready for these kind of platforms, and that it still has a long way to go to meet the criteria of mission-critical systems. (Note that, according to Maxwell and Hartman's Mars Rover talk, Linux is already able to handle all the planning tasks that are executed on the ground.)

The talk is available online in mp3 format, at http://www.usenix.org/events/usenix05/tech/mp3/klein.mp3.

FREENIX TRACK

MULTIMEDIA

Summarized by Christian Kreibich

- ***OpenCSG: A Library for Image-Based CSG Rendering***

Florian Kirsch and Jürgen Döllner, University of Potsdam

Florian Kirsch started his talk with an introduction to graphical modeling using Constructive Solid Geometry (CSG). Florian explained that CSG consists of performing union, intersection, and subtraction operations on closed geometric primitives in order to obtain more complex shapes. While CSG is well known, it is typically used only in offline rendering systems such as RenderMan or POVray. Florian then presented approaches suitable for online CSG processing, singling out image-based rendering, where the final image is composed in the frame buffer. Florian presented the two basic algorithms used in this approach, the Goldfeather and SCS algorithms. The former can handle any kind of geometric primitive

and counts layers of object surfaces to determine visibility. The latter is faster, but only works for complex primitives. It successively removes components in order to produce the final result.

In the remainder of the talk, Florian presented OpenCSG, the first open source library for image-based CSG rendering. It is implemented in C++, has OpenGL as its only external dependency, and features a simple API allowing the user to define arbitrary primitives.

Florian concluded his talk with a live demo and gave a number of potential applications of OpenCSG—for example, in gaming or terrain modeling. Also, the RenderMan modeler Ayam is already using OpenCSG. More details about OpenCSG can be found at http://www.opencsg.org.

- ***FreeVGA: Architecture-Independent Video Graphics Initialization for LinuxBIOS***

  Li-Ta Lo, Gregory R. Watson, and Ronald G. Minnich, Los Alamos National Laboratory

  Li-Ta Lo presented FreeVGA, a graphics initialization module for LinuxBIOS using x86 emulation in order to run the VGA BIOS. Li-Ta started with a summary of LinuxBIOS's essential design goals and features, and pointed out that VGA initialization was the key missing piece: The traditional VGA initialization process cannot be used by LinuxBIOS, because it does not provide the 16-bit callback interface the VGA BIOS uses to interact with the system BIOS. Li-Ta next presented the solutions adopted by SVGAlib, ADLO, the VIA/EPIA port of LinuxBIOS, and XFree86, and concluded that none of these graphics projects provided a satisfying solution.

  FreeVGA solves the problem by integrating x86emu in LinuxBIOS, keeping the code free of x86 specifics, making it easier to debug and test, and providing graphics initialization as early as possible in the boot process. Li-Ta explained in detail the changes required to LinuxBIOS to facilitate this approach. FreeVGA increases the ROM image size by 16KB and 40KB during run time, has no significant impact on boot time because the VGA BIOS spends most of the time polling hardware, and works well on modern AGP cards. Li-Ta mentioned chipset dependencies and support for other architectures as the main items for future work, and concluded his talk by stating that FreeVGA is a successful demonstration of the feasibility of using an emulator for VGA initialization. Furthermore, work is underway to integrate the approach in the Linux kernel tree.

  After the talk, an attendee wondered whether the emulation approach isn't too complicated. Li-Ta explained that FreeVGA development does not actually require in-depth understanding of the emulator, allowing the developers to focus on the important parts of the code. Another attendee asked whether the approach also works on older graphics cards. Li-Ta pointed out that a number of older cards had also been tried and mostly worked. The next question addressed the possibility of using the seemingly generic technique of hardware initialization for other classes of devices as well. Li-Ta stated that things like SCSI initialization are feasible as well; basically, any kind of PCI card support could be made to work.

- ***The Ethernet Speaker System***

  David Michael Turner and Vassilis Prevelakis, Drexel University

  David Turner started his talk with a summary of the current state of the art for conventional distributed audio systems. These systems typically are analog and wired, expensive, and proprietary. The goal of the Ethernet Speaker (ES) System was to provide an open-source and low-cost solution to the problem, only requiring an embedded PC, stereo speakers, and stereo audio and networking cards, available for a total of around US$50.

  David explained that in the ES System, speakers are installed like normal Ethernet network clients and the network itself is used to distribute the audio signal. Since in the ES System, speakers are "smart" compared to typical "dumb" conventional speakers, the ES nodes can perform more intelligent processing in order to replay the audio signal. Ethernet speakers play audio transmitted by a remote audio server, using a local rebroadcaster that communicates with the speakers directly. The rebroadcaster is the central instance converting the audio signal to a single transmission format that the Ethernet speakers understand. This transmission protocol is built on top of UDP multicast, is connectionless, is completely one-way, and consists of periodic audio stream control packets among audio data packets. David next explained how the audio driver for the ES System was realized in the OpenBSD audio architecture. Actual hardware is replaced with a virtual card that returns the raw audio data back into user space for distribution to the speakers.

  David concluded his talk by mentioning a few of the current challenges of the system. These include rate limiting to provide the correct replay speeds, data compression, and synchronization issues between multiple speakers.

  After the talk, an attendee asked on what scale the Ethernet Speaker System is meant to be deployed. David explained that the initial motivation was a campuswide announcement speaker scenario. Another attendee asked whether relative positioning of the speakers could be exploitable for solving the synchronization problem. David answered that this would likely be tricky, but certainly an interesting way to attack the problem.