conference reports

THANKS TO OUR SUMMARIZERS

2008 USENIX Annual Technical Conference

Boston, MA
June 22–27, 2008

VIRTUALIZATION

Summarized by John Krautheim (kraut1@umbc.edu)

■ ***Decoupling Dynamic Program Analysis from Execution in Virtual Environments***
Jim Chow, Tal Garfinkel, and Peter M. Chen, VMware

Awarded Best Paper!

Jim Chow described a novel method for software testing and debugging using a virtual machine (VM) as recording and replay device. The concept is not new, but the technique presented provides a new tool for the arsenal of software developers and testers.

Jim points out that one of the main reasons for developing such a tool is the lack of automated methods in software development. The team at VMware wanted to make Dynamic Program Analysis (DPA) more accessible. DPA is ability to take a running computer program, stop it, and inspect its state. This technique is very useful for the programmer; however, existing tools for DPA have a very high overhead from context swapping, instrumentation, and analysis, which results in a slowdown on the order of one hundred times. Therefore, the VMware team looked for a way to improve this analysis technique without the slowdown from overhead. The solution the team came up with was to decouple the analysis and execution by parallelizing the problem with virtual machines. This allows the target system to run freely in one VM while the analysis system records and regenerates events on a separate VM. The hypervisor is used to record all inputs to the VM under analysis and can start the analysis machine from the same state and replay instructions. The analysis system regenerates all the data needed, removing the overhead from the target system. Since the overhead of recording is very efficient with virtual machines, the target system can run at roughly native speed.

To demonstrate the technique, the team developed the Aftersight system. Aftersight is built on the VMware virtual machine monitor and thus it inherits many properties of VMs that can be leveraged to solve the problem. The Aftersight system provides isolation of the target and analysis system so that the analysis is self-contained and communication bottlenecks are eliminated. Through parallelism, the analysis and the target can run separately, on multiple cores if available. This allows analysis to go faster, and multiple analyses may be performed at the same time. An added side benefit of this parallel playback and recording is that ex-post-facto analysis can be performed on behavior not known at the time of recording, providing the ability to examine events not foreseen at execution time. The team has used Aftersight to debug VMware's own ESX Server, the Linux kernel, and the Putty secure shell client, finding previously undiscovered bugs in all three.

The Aftersight system relies on the concepts of heterogeneous replay and parallel analysis. Heterogeneous replay is the ability to record and replay events at the same time, thus increasing the speed and timeliness of analysis. Parallel analysis allows analysis and system execution simultaneously, further increasing the timeliness of the results. Implementing heterogeneous replay and parallel analysis presents several technical challenges. First, keeping target and analysis systems in sync with each other without slowing the target system down is difficult. The analysis is typically slower than the target system, and there are times when the target system must be blocked because of resource allocation issues. This limitation can be overcome by additional buffering in the target system and further refinement and tuning of the analysis system to speed it up. However, there are situations where the analysis system just cannot keep up, so additional techniques such as forward caching and buffering can be applied. Also, the addition of more processing cores in the system can help offload the analysis task through further parallelization.

This talk gave several compelling reasons for using dynamic program analysis and showed how decoupling the execution and analysis environments can significantly improve productivity and effectiveness. The Aftersight system appears to have many useful applications in the development, test, and security worlds. The audience was greatly intrigued and several questions arose on the difference between the decoupled approach and existing parallel environments. The difference is at what level the recording and playing occur. Jim stated that recording at the OS level incurs more overhead than recording at the hypervisor level.

- ***Protection Strategies for Direct Access to Virtualized I/O Devices***
  Paul Willmann, Scott Rixner, and Alan L. Cox, Rice University

Paul Willmann, now with VMware, presented performance and safety measures of several strategies for access control to I/O devices from within virtualized environments. Direct access to I/O devices is required in many datacenter applications where high throughput performance is needed; however, access to these devices needs to be controlled to protect from an untrusted virtual machine (VM) tampering with or using devices it does not have permission or privilege to use.

Wallmann presented implementations of protection strategies in both hardware and software, with surprising results. Hardware implementations utilize an Input Output Memory Map Unit (IOMMU) to implement single-use mappings, shared mappings, persistent mappings, and direct mapping strategies. The software strategy is an implementation of the single-use mapping that requires the guest OS's drivers to register with the virtual machine monitor (VMM) before access is granted to the device. Both hardware and software implementations have advantages and disadvantages that are evaluated in the paper.

The different strategies were evaluated based on performance and protection capability in inter-guest and intra-guest protection categories. Three types of invalid accesses were evaluated for each strategy and for each category: bad address, invalid use, and bad device. The results showed that hardware implementations worked very well and efficiently for intra-guest protections, but it did not perform well for inter-guest protection. The software implementation performed well in all inter-guest protections except for the bad device case. Additionally, the software method provides additional protection in the intra-guest invalid use case.

With all the strategies showing very good overall protection, the biggest differentiator among the various strategies becomes performance-related. Several benchmarks were run against the strategies, including a TCP stream, a VoIP server, and a Web server. The benchmark also tested against various levels of mapping reuse. The results showed that the single-use strategy had the highest inter-guest overhead, at 6%–26% of CPU workload; however, significant mapping reuse can greatly reduce that overhead. Persistent mappings showed the highest performance, at only 2%–13% overhead with nearly 100% reuse. The software implementation showed better performance than two of the hardware strategies (single-use and shared), with 3%–15% CPU overhead. The direct-mapped hardware strategy was the best performer, although it had limited intra-guest protection capability.

The surprising result is that the software protection strategies utilized in this paper provide performance comparable to or better than the hardware IOMMU results while still maintaining strict inter-guest and intra-guest protection.

- ***Bridging the Gap between Software and Hardware Techniques for I/O Virtualization***
  Jose Renato Santos, Yoshio Turner, and G. (John) Janakiraman, HP Labs; Ian Pratt, University of Cambridge

Jose Renato Santos's talk was on improving I/O performance in virtual machines through combining hardware and software techniques. In a virtualized environment, physical devices need to be multiplexed so that each guest virtual machine (VM) can use the device. This multiplexing can be handled in software and hardware, each with its advantages and disadvantages. Software incurs a significant overhead in managing the device; however, the driver is simplified by providing a transparent interface as I/O access is handled by the host OS using a device-specific driver and the guest can use a standard virtual device driver independent of the hardware. The hardware approach is more complicated since the transparency is reduced, requiring each guest VM to have a device-specific driver; however, the performance is usually much better.

The HP Labs research team wanted to reduce the performance gap between driver domain model and direct I/O while maintaining transparency. To do so, they analyzed the Xen device driver model, focusing on the networking receive path, and compared the same workload with a direct I/O approach. They focused on several areas to improve the performance of the Xen driver. First, they reduced the data copy cost by keeping all copies between guest and driver domains on the same CPU to increase cache hits. Next, they avoided extra data copies by using dedicated NIC receive queues. Finally, they reduced the cost of the grant mechanisms, the second highest cost in Xen, by maintaining grants and mappings across multiple accesses. The team was able to reduce the receive path execution costs for a conventional NIC by 56%. For devices with multiple hardware receive costs, they were able to achieve performance near direct hardware I/O while maintaining the benefits of the Xen driver model. This is a significant improvement in performance over the original driver domain model in Xen.

By keeping the new multi-queue completely hidden from the guest and encapsulated in the driver domain, migration to the new driver is completely transparent to the guest. The team has stated that the new mechanisms will be updated in the Xen Netchannel2 in approximately 2–3 months. The next improvement they plan to make will be to look at high-bandwidth (i.e., 10 GigE and multiple guests) improvement in the Xen driver.

INVITED TALK

■ ***Free and Open Source as Viewed by a Processor Developer***
Peter Kronowitt, Intel

Summarized by Ward Vandewege (ward@gnu.org)

Peter Kronowitt's talk grew from an internal Intel presentation. He works in the Software Solutions Group, which optimizes software—all sorts of software, ranging from embedded to server. The purpose of the optimization is to ensure that when the product reaches the marketplace, there is a complete hardware and software solution.

The traditional software-enabling model at Intel goes something like this. Intel works with over 12,000 software companies. Most of these are proprietary, so Intel has to sign nondisclosure agreements (NDAs). Then engineers are assigned; they need time to get the work done, and then Intel has to wait for the market to generate demand in order to get to a mutually beneficial state for Intel and its partners.

Open source development is very different. Intel feeds software into the kernel. That software then gets picked up by community distributions such as Debian, Fedora, and OpenSuse, and those in turn feed into the products of Linux companies such as Canonical, RedHat, and Novell. This is a much more efficient model.

Intel has learned to work more effectively with kernel developers: In 2001, Alan Cox, a core kernel developer, gave direct feedback that Intel required many NDAs and was secretive about its hardware, making it very difficult to work with. Fast forward to 2007 when Alan Cox said that Intel is one of the most cooperative hardware vendors, providing good docs, errata, and software such as graphics drivers. In those six years, Intel has learned and relearned a lot of stuff.

Linux is estimated to be one-third of the market based on server shipments today. But tracking open source software (OSS) is very difficult. This is a problem—if Intel can't tell what software customers are using, it cannot put its resources in the right place to make sure the hardware works perfectly. Intel needs to know what software customers are using and deploying in order to be able to offer a "complete solution." Also, OSS is growing three times as fast as proprietary software.

Intel has been growing its open source involvement over the years, starting in 1990 when Linus Torvalds booted Linux on Intel Architecture for the first time. He was able to do that because Intel had released detailed specifications for the Intel Architecture. Since 2003, Intel has become more visibly active as a contributor to OSS. The following paragraphs highlight some examples of how Intel has been working with the OSS community over the years.

The PC BIOS had not changed for over 20 years. Intel launched the Tiano project to replace it. This was done in partnership with CollabNet, establishing the extensible framework interface (EFI) dev kit. From this, Intel learned how open source can drive industry change.

In 2003 Intel joined other vendors in a virtualization research project called Xen at Cambridge University in the UK. In 2004 Intel started contributing a large amount of code to the open-source project. Today a large ecosystem exists around virtualization, and Intel has been contributing to many projects in that space. Xen helped catalyze Intel feature adoption by vendors of virtualization products.

The telecom industry was a highly proprietary, vertically integrated industry that overinvested during the dot-com era. Intel was a founding partner of the Open Source Development Labs (OSDL), contributing to the kernel and the Carrier Grade Linux (CGL) specification. When the dot-com bubble burst, the carriers needed to cut costs, and Intel's involvement with CGL helped the Intel Architecture break into the telco industry.

In the late 1990s, Merced, the Itanium platform, solidified numerous operating system porting commitments. Intel worked with many OS vendors and indirectly contributed to the Linux kernel. Linux and Itanium helped Intel gain access to the RISC market.

Initially, Intel made Linux kernel contributions via proxy. This meant that Intel was not very visible as a community member. After long, difficult internal negotiations on open sourcing drivers, Intel started contributing code directly to the kernel. This direct participation in the community has accelerated Intel technology adoption.

Influencing Java was . . . challenging. Intel, like numerous other industry players, requested that Sun open source Java. Eventually, Intel participated in the launch of the Harmony project with other industry players, including IBM. Harmony was a clean-room OSS Java implementation. Eventually, this encouraged Sun to release an OpenJDK.

More recently, Intel has been working on Moblin, an optimized software stack for Atom-based clients. This software stack is aimed at mobile Internet devices, netbooks, cars, etc. See http://moblin.org for more information. Intel also launched LessWatts.org, an Intel open source project to make Linux greener.

DISK STORAGE

Summarized by Christopher Stewart (stewart@cs.rochester.edu)

■ ***Idle Read After Write—IRAW***

Alma Riska and Erik Riedel, Seagate Research

When users issue writes to a disk, they assume their exact data has been stored. However, mechanical anomalies can cause the data actually stored on disk to deviate from the user's original data (a.k.a. data corruption). Worse, such corruption can be silent, causing the user to wrongly believe their data was correctly written to the disk. Alma Riska presented Idle Read After Write (IRAW), a low-overhead approach to detecting silent data corruption. IRAW issues a disk read for recently written data during periods when the disk is idle. The data returned by the read is compared to a cached copy of the actual data the user intended to write to the disk; if the two differ, appropriate recovery actions are taken (e.g., retry).

Compared to a standard disk, IRAW improves reliability by validating writes soon after they occur. An alternative is to validate each write immediately after it happens (RAW). RAW improves reliability, but it degrades performance by placing an additional disk operation on the critical path of every write. In comparison, IRAW delays the validation until the disk is idle, and therefore it hides the cost of the additional read from the end user. IRAW therefore requires enough idle time for the additional disk operations to complete. Alma presented empirical evidence from five disk traces, all of which had more than enough idle time to perform the delayed reads.

Empirical results using IRAW show that it indeed has low overhead. One experiment showed that the performance of an IRAW-enabled disk almost matched that of a standard disk for a Web server application. Further, IRAW may not degrade other performance-enhancing disk operations. For instance, many applications can benefit by enabling IRAW and idle wait simultaneously. Finally, Alma showed that the footprint of IRAW in the disk cache was not too large for today's disks.

Adam Leventhal from Sun Microsystems asked whether IRAW could be applied at the filesystem level. Alma said that it is possible, but the file system will probably be less effective at identifying true idle time on the disk. Geoph Keuning from Harvey Mudd College asked whether it was even important to be concerned with the amount of cache space dedicated to IRAW, since volatile memory is getting cheaper. Alma said that one design goal was to make IRAW practical for today's disks, which meant keeping the footprint below 4–6 MB.

■ ***Design Tradeoffs for SSD Performance***

Nitin Agrawal, University of Wisconsin—Madison; Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy, Microsoft Research, Silicon Valley

Solid-state disks (SSDs) can perform certain I/O operations an order of magnitude faster than rotating disks. They have the potential to revolutionize storage systems. However, little is known about the limitations of their architecture. Nitin Agrawal discussed several inherent challenges for SSD devices and proposes solutions. The analysis is based on a detailed understanding of the architecture of SSD devices. For instance, a write to an SSD block requires that the block's contents be erased and rewritten. Further, SSD blocks can only be erased a certain number of times. Such architectural properties affect the performance and reliability of SSDs.

The granularity of writes affects the performance of SSDs. Specifically, workloads that perform writes to random locations on disk perform orders of magnitude worse than those that perform random reads. Empirical evidence showed a difference of 130 random writes per second compared to almost 20,000 random reads per second. Nitin demonstrated that properly mapping logical pages to physical blocks can improve the performance of random writes. A second performance challenge faced by SSDs is bandwidth bottlenecks. Striping and interleaving are good solutions to mitigate the bandwidth bottleneck by distributing I/O for logical blocks across multiple channels. Intuitively, this solution exploits the potential for parallelism in storage access patterns. Finally, SSD blocks wear down after a certain number of erasures and rewrites. To maximize the lifetime of the device, Nitin proposed a novel wear-leveling algorithm that increases the usable lifetime of an SSD by delaying expiry of any single block.

Jason Flinn from the University of Michigan asked Nitin about the benefit of wear-leveling, given that the whole device will wear out eventually anyway. Nitin said that wear-leveling reduces the long-term cost of SSDs, since the failure of individual blocks could force a company to purchase a new device when only a small portion of its capacity is unusable. Sean Rhea noted that wear-leveling is most beneficial for applications that frequently write to only a few pages but access many pages for reading (i.e., small hot set and large cold set). Nitin agreed.

■ ***Context-Aware Mechanisms for Reducing Interactive Delays of Energy Management in Disks***
Igor Crk and Chris Gniady, University of Arizona

Igor began by saying that most disks now support different power modes for energy conservation. The disk can be powered down during idle times to consume less energy and then spun up to an operational power mode when I/O requests arrive. In today's interactive systems, the additional latency for I/O requests that interrupt idle periods (i.e., the time for a disk spin-up) is typically seen by the end user (who then gets miffed and maligns the system as slow and unresponsive). Igor presented a mechanism to hide spin-up latency from end users by preemptively changing the disk to full-power mode before I/O requests happen. The key is to identify end-user GUI events that signal that a disk I/O is imminent. When such events happen during an idle period, the disk can be moved to an operational power mode in anticipation of the impending request. For instance, a mouse click on a "file open" button may be a good signal that an I/O request is imminent and the disk should preemptively be spun up.

Compared to today's default policy (no preemptive spin-up), the proposed solution can hide spin-up delays from the end user. Further, by considering the context of the GUI event, the proposed solution can achieve better energy conservation than a naive solution that preemptively spins up after every mouse click. Context information was collected by intercepting calls to the X windows server. Specifically, each X windows event updated a table that tracked the number of times that the event occurred in a particular context and the number of times it was followed by I/O. After data was collected for a long period of time, the event contexts that were most likely to be followed by disk I/O were tagged as good predictors. Empirical results show that preemptive action based on the identified predictors does hide the latency of disk spin-up from end users, while conserving more energy than a naive approach that does not consider the context of the event. Further, Igor mentioned that the proposed system allows users to trade off the latency they see for more energy conservation by adjusting the threshold at which an event qualifies as a predictor.

Yu Chen from Fermilab asked whether they were able to accurately predict disk requests for systems that had a large file system cache. Igor noted that the difference between file system requests and disk I/O was a significant challenge. In the current implementation, they identify the GUI events likely to cause file system requests and apply a heuristic to predict disk requests. Christopher Stewart from the University of Rochester noted that user satisfaction, as measured by Mallik et al. at ASPLOS 2008, could guide the setting of the threshold that determines when an event qualifies as a predictor. Igor agreed that the combination of the two works could be beneficial. However, he noted that GUI events can predict I/O well in advance, so a combination of the techniques may significantly affect performance.

NETWORK

Summarized by Matthew Sacks (matthew@matthewsacks.com)

■ ***Optimizing TCP Receive Performance***
Aravind Menon and Willy Zwaenepoel, EPFL

Aravind Menon demonstrated the ability to improve TCP performance by focusing on the receive side of the TCP protocol. Menon argued that receive-side optimizations are missing, contributing to lesser performance of the TCP protocol. Linux was used as the demonstration OS for his concepts, although the same principles can be applied to any operating system. Menon shows that there are two types of overhead: per-byte optimizations and per-packet optimizations. Per-packet overhead costs are the primary overhead contributor on newer CPUs, whereas on older processors the issue was with per-byte overhead.

Menon presented two types of performance improvements in his talk: receive aggregation and TCP acknowledgment offload. Receive aggregation aggregates multiple incoming network packets into a single host packet accounting for a 45%–86% increase in performance. Receive aggregation requires that the packet must be the same TCP connection, must be in sequence, and must have identical flags and options. Receive aggregation works best when receiving at a high rate of transfer. To implement this method in Linux the network driver must allocate raw packets rather than sk_buffs.

For acknowledgment offloading, the normal method of generating ACK packets, by a one-to-one mapping, is replaced by a template to generate the ACK packets, which in turn avoids buffer management costs. This must be done at the device-driver layer. Nonprotocol overhead has the greatest impact on TCP performance such as buffer management, and, specifically for the Linux driver, it also processes MAC-level analysis of each packet.

■ ***ConfiDNS: Leveraging Scale and History to Detect Compromise***
Lindsey Poole and Vivek S. Pai, Princeton University

Lindsey Poole presented a new project called ConfiDNS, which is based on the CoDNS cooperative DNS resolver system. CoDNS is a wrapper for local DNS resolution that allows faster lookups and high availability for DNS lookups. CoDNS utilizes PlanetLab for ensuring high availability as a distributed service.

ConfiDNS takes the CoDNS project and addresses the security vulnerabilities in CoDNS, which is susceptible to contamination from a single resolver being propagated throughout the entire system. The way ConfiDNS works is that when the local resolver fails, it forwards the request to peer nodes on the PlanetLab network (a feature that was present in CoDNS). ConfiDNS preserves a history of lookups and the client can specify policies for DNS lookups.

Another problem encountered with CoDNS is DNS lookups served by global content distribution networks, which may return multiple IPs from different locations for the same hostname. ConfiDNS addresses this problem by implementing a peer agreement algorithm that compares results from multiple resolutions from different geographic locations and then returns a result.

ConfiDNS proves that you can improve DNS resolution performance without compromising security. DNS attacks on the local system are much easier to carry out. ConfiDNS protects against attacks such as cache poisoning or spoofing, and it improves performance at the same time.

- ***Large-scale Virtualization in the Emulab Network Testbed***
  Mike Hibler, Robert Ricci, Leigh Stoller, and Jonathon Duerig, University of Utah; Shashi Guruprasad, Cisco Systems; Tim Stack, VMware; Kirk Webb, Morgan Stanley; Jay Lepreau, University of Utah

Emulab, a network testbed at the University of Utah, allows researchers and engineers the ability to specify a network topology including server systems to which you have root access. One of the difficulties that the Emulab maintainers experienced was a limitation in the amount of hardware available to them; therefore, a virtual solution for the network and systems was needed to power the Emulab testbed. One of the requirements of the virtual solution was that the virtual environment needed to retain the same fidelity of experiments running on the testbed so that the results would not be affected. At first FreeBSD jails were used to address this; however, jails alone were found to fall short in addressing the issue of network virtualization, so the Emulab team designed a more robust virtualization platform that expanded on the FreeBSD jail's limitations.

The team at Emulab implemented a robust network virtualization solution by developing a virtual network interface device, which is a hybrid encapsulating device and bridging device. The "veth" interface allows creation of unbound numbers of Ethernet interfaces, which then communicate transparently through the switch fabric. Veth devices can be bridged together or with physical interfaces to create intra-node and inter-node topologies. In addition to virtual network interfaces, the Emulab team also had to implement virtual routing tables that are bound to each jail and virtual interface based on the Scendaratio and Risso implementation, which implements multiple IP routing tables to support multiple VPNs. Also, for the virtual nodes themselves, the Emulab team designed a resource-packing methodology called "assign" which "packs" virtual hosts, routers, and links into as few physical nodes as possible without overloading the physical nodes. This method allows up to a 74:1 compression ration of virtual nodes/networks to physical hosts.

The research done on the Emulab testbed in addressing these scaling issues with virtual networks and nodes has enabled the team to scale efficiently while keeping the same fidelity as strictly physical hardware by using virtual interfaces and resource packing. The efficiencies achieved in the Emulab virtualization implementation now allow experiments to be executed on up to 1000 nodes, permitting powerful simulations without impact onm the fidelity of the experiments.

INVITED TALK

- ***Millicomputing: The Future in Your Pocket and Your Datacenter***
  Adrian Cockcroft, Netflix, Inc., and Homebrew Mobile Club

  Summarized by Tom Clegg (tom@tomclegg.net)

Low-power computing devices such as mobile phones—which Adrian Cockcroft calls "millicomputers," because their power requirements are measured in milliwatts rather than watts—are increasing in capacity faster than their hundred-watt datacenter counterparts. In this talk, Cockcroft gave an overview of the current state of low-power technology and cheap open hardware in particular, considered some of the applications that become possible as mobile devices approach the capacity of personal computers, and outlined a speculative "enterprise millicomputer architecture" employing thousands of low-cost nodes per rack.

In 2007, the iPhone was notable for running a full Mac OS rather than a cut-down embedded operating system—it ships with 700 MB of system software. Clearly, portable millicomputers such as the iPhone provide a real application platform. Cockcroft showed photos of a prototype "myPhone"—a Linux-based GSM/EDGE phone with many built-in features and connectivity options, and CPU and RAM specifications similar to the iPhone. In 2008, the emergence of Google Android as an open source alternative to the iPhone platform has generated a lot of developer interest. The highest-performance smart phone hardware will raise the bar further with 256 MB RAM, 16–64 GB storage, twice the CPU speed, and faster networking. AT&T plans to implement HSPA release 7 in 2009, which will deliver speed "exceeding 20 Mbps" and has a "clear and logical path" to 700-MHz 4G access in the 2010 timeframe, which should increase speed to nearly 100 Mbps. Meanwhile, short-range low-power networking is reaching 480 Mbps as Ultra-Wideband Wireless USB starts to roll out. Nonvolatile storage is steadily becoming cheaper, and emerging storage technologies promise dramatic speed increases in a few years. In the CPU market, we can expect 1-GHz quad-core processors to arrive in 2010.

Given the pace of mobile technology advances, the time is coming into view when pocket devices with wireless docking can replace laptop computers, just as laptops replaced desktop computers for many users. Combining workstation computing power with mobile connectivity, we might see "life-sharing" applications such as full-time video conferencing and virtual world integration. Integrating accessories such as an accelerometer, compass, and brainwave reader, we have a system with many possible uses such as computer-assisted telepathy, ambient presence, immersive personal relationships, and better ways to monitor and care for physically disabled people.

In addition to mobile applications, these tiny low-power computers have potential applications in the datacenter. They could help reduce power consumption, which is already a limiting factor in many situations. Cockcroft presented one possible architecture to demonstrate how a computing cluster might be constructed using low-cost mobile device boards. Modules packed onto a 1U enterprise motherboard yield a fully distributed heat model that is much easier to cool than a typical server board. Two groups of seven modules are connected via USB switches to each of eight gateway/load balancer nodes, each having two gigabit network interfaces. Thus, each rack unit has a total of 112 CPUs and 28 GB of RAM, consuming 24 W when idle and 160 W at peak power. Adding 8 GB microSDHC cards with 20 MB/s I/O each, we have 896 GB per rack unit of storage with 2240 MB/s I/O. The $14,000 cost of this system is comparable to a 1U Sun server with similar specifications—but the millicomputer offers much faster storage I/O with zero seek time and more network bandwidth, using little more than half the power.

Software implications of this platform include a small application memory limit (256 MB) on par with mainstream systems from 2001. Management implications include the need for lightweight monitoring, aggregation tools, and load balancing. This platform would be well suited to horizontally scalable applications such as Web content delivery, legacy applications that could run on five-year-old machines, storage I/O-intensive applications, and graphical video walls.

One participant pointed out that the low bandwidth between nodes could be a serious limitation. Cockcroft explained that the USB approach was taken to minimize power consumption, and the CPU power is not enough to saturate a gigabit network interface in any case. This feature of the design makes it more suitable for applications with low IPC demands, such as Web servers. It would also be possible to use other low-power interconnects, perhaps based on FPGA technology, which would give better interconnect bandwidth. It should also become less of an issue as RAM size increases. Another participant suggested a heads-up display with facial recognition software as an interesting mobile application. Cockcroft added that, although signal processing chips can be a big power drain, many processing tasks can be postponed until nighttime, when the device is plugged into a charger and it's acceptable for it to get a bit hotter than comfortable pocket temperature. Another participant brought up the possible impacts of mobile technology on the way we interact with services; Cockcroft referred to "taking the friction out of interactions" with always-on networking and services such as continuously updated status tracking. Another participant wondered whether this mobile power could simply do away with the role of the data center; Cockcroft offered that, although there tends to be a pendulum alternating between client and server focus, there will likely always be a place for centralized services, but certainly more can happen in the pocket.

Current information on millicomputing can be found at http://millicomputing.blogspot.com/.

FILE AND STORAGE SYSTEMS

Summarized by Zoe Sebepou (sebepou@ics.forth.gr)

- ***FlexVol: Flexible, Efficient File Volume Virtualization in WAFL***

  John K. Edwards, Daniel Ellard, Craig Everhart, Robert Fair, Eric Hamilton, Andy Kahn, Arkady Kanevsky, James Lentini, Ashish Prakash, Keith A. Smith, and Edward Zayas, NetApp, Inc.

John Edwards presented their work on a new level of indirection between physical storage containers (aggregates) and logical volumes (FlexVol volumes). An aggregate consists of one or more RAID groups, and its structure resembles that of a simple file system, keeping the changes made on the individual FlexVol volumes. The main goal of FlexVol was to provide new functionality by decoupling the physical device management from the data management. The decoupling strategy gives administrators the flexibility to enforce different policies on different volumes and to dynamically grow or shrink the volumes.

The mapping between the virtual block addresses of FlexVol and the physical addresses used by aggregates requires extra processing and disk I/O to deal with the address translation of each indirect block. This challenge is addressed with two main optimizations: dual block numbers and delayed block freeing. Block pointers in a FlexVol volume have two parts: the logical location of the block in the container and its physical location. In delayed block freeing, free space is held by the aggregate, not the volumes, so one counts the number of delayed free blocks and performs a background cleaning after a specific threshold. These optimizations help to reduce the overhead and result in at most a small degradation in the system's overall performance compared to traditional volume approaches.

The evaluation of FlexVol was made through the use of micro-benchmarks, including the comparison of read and write in sequential and random access patterns. Their results indicate that FlexVol performance is almost identical to that of the traditional volumes, and in the worst cases the performance difference is from 4% to 14% (mostly in random cases involving metadata overhead in write operations). Finally, Edwards provided some insight into the current use of FlexVol and its services, showing the growing adoption of FlexVol by their customers.

■ ***Fast, Inexpensive Content-Addressed Storage in Foundation***
Sean Rhea, Meraki, Inc.; Russ Cox and Alex Pesterev, MIT CSAIL

Sean Rhea presented Foundation, a preservation system based on content-addressed storage (CAS) aimed at providing permanent storage of users' personal digital artifacts. Sean pointed out that the increasing use of computers to store our personal data would lead to the undesired situation that this data would be unavailable in the future. Indeed, as software and hardware components depend on each other to make an application operate and provide the desired functionality, a user in the future would need to replicate an entire hardware/software stack in order to view the old data as it once existed. To overcome this problem, the authors, inspired by Venti, designed and developed Foundation. Foundation differs from Venti mostly in that instead of using an expensive RAID array and high-speed disks, it only uses an inexpensive USB hard drive, making the deployment of this system easy and possible for consumer use.

Foundation permanently archives nightly snapshots of a user's entire hard disk containing the complete software stack needed to view the data (with user data and application and configuration state of the current system captured as a single consistent unit). To eliminate the hardware dependencies, Foundation confines the user environment to a virtual machine. As in Venti, the use of content-address storage allows Foundation to have limited storage cost, actually proportional to the amount of new data, and to eliminate duplicates through the use of a bloom filter; other filesystem-based approaches miss this benefit.

The major components of Foundation include the Virtual Machine Monitor (VMM), the filesystem Snapshot Server (SMB), the virtual machine archiver, and the CAS layer, whose main use is to store the archived data on the inexpensive external disk and/or replicate it using a remote FTP server. The users operate on an active virtual machine which runs on top of the VMM. The VMM stores the state of the virtual machine in the local filesystem and every night the virtual machine archiver takes a real-time snapshot of the active VM's state and stores the snapshot in the CAS layer. The SMB server is used to interpret the archived disk images and present the snapshots in a synthetic file tree, accessible by the active VM over the server.

To eliminate several of the problems that appear in similar systems such as Venti, their proposed solution to reduce disk seeks is to reduce as much as possible the hash table lookups. In the case of writing, lookups occur when the system needs to update a block index and when determining whether a block has been accessed before. In these cases Foundation uses a write-back index cache that is flushed to disk sequentially in large batches. During read operations, lookups are required in order to map hashes to disk locations. In this case they start with the list of the original block's hashes, they look up each block in the index, and they read blocks from the data log and restore them to disk. Moreover, with the use of CAS they take advantage of the fact that, given a block, CAS gives back an opaque ID. This allows block locations to be used as IDs, completely eliminating read-indexing lookups and thus still allowing for potential duplicate finding using hashing.

For the evaluation of the Foundation system, the authors focused on the performance of saving and restoring VM snapshots. The important metrics taken into consideration were how long it takes for Foundation to save the VM disk image and how long it takes to boot old system images and recover old files from the directory tree. Foundation's algorithm in its two modes, by-hash and by-value, was compared against Venti's algorithm. The results indicate that Foundation operates efficiently and gives higher read and write throughput in the majority of the tested cases compared to Venti. Sean Rhea concluded that Foundation is a consumer-grade CAS system that requires only a USB drive and can be used not only as a preservation system but also as an inexpensive household backup server. Moreover, it can automatically coalesce duplicate media collections and operates efficiently without requiring a collision-free hash function.

■ ***Adaptive File Transfers for Diverse Environments***
Himabindu Pucha, Carnegie Mellon University; Michael Kaminsky, Intel Research Pittsburgh; David G. Andersen, Carnegie Mellon University; Michael A. Kozuch, Intel Research Pittsburgh

Himabindu Pucha described dsync, a file transfer system that can correctly and efficiently transfer files in a wide range of scenarios. By choosing to use all the available resources (the sender, the network peers, and the receiver's local disk) and by constantly monitoring recourse usage, dsync overcomes performance limitations present in other similar systems such as rsync and peer-to-peer systems such as BitTorrrent. Although the primary resource used by dsync is the network, dsync dynamically chooses, if necessary, to spend CPU cycles and disk bandwidth to locate any relevant data on the receiver's local file system in order to enhance performance.

dsync retrieves chunks over the network either from the sender or from any available peer that has downloaded the same or similar data. It also makes the optimization to look at the receiver's local disk for similar data by spending some of the system's CPU resources to compute the hash of data from the local disk and for scheduling purposes. Specifically, dsync source divides each file (or file tree) in equal-sized chunks and by hashing the chunks computes for each chunk a unique ID. A tree descriptor is then created describing the file layout in the file tree, the metadata, and the chunks that belong to each file. So, given a tree descriptor, dsync attempts to fetch the file chunks from several resources in parallel using the optimal resource at any given time.

The evaluation of dsync was done for several transfer scenarios; results for single receiver (one source—one receiver) and multiple receivers (in homogeneous/heterogeneous environments—PlanetLab nodes) indicate that dsync can effectively use the available resources in any environment. Moreover, the back-pressure mechanism allows for optimal resource selection and the heuristics used quickly and efficiently locate similar files in real file systems.

One questioner asked whether they have attempted to find a solution that is globally good, given that resources are to be shared among several receivers. The answer was that currently each receiver greedily uses the resources to minimize its download time, but they would like to look at strategies that enable cooperation among receivers to improve their performance.

KEYNOTE ADDRESS: THE PARALLEL REVOLUTION HAS STARTED: ARE YOU PART OF THE SOLUTION OR PART OF THE PROBLEM?

David Patterson, Director, U.C. Berkeley Parallel Computing Laboratory

Summarized by Christopher Stewart (stewart@cs.rochester.edu)

Patterson began by saying that his speech was motivated by the revolution under way in computer architecture: Microprocessors are out; parallel architectures are in. Patterson argued that the design shift from microprocessors is inevitable, so the systems community would do best by embracing parallel architectures and finding solutions to the new challenges they present. “I wake up every day and can’t believe what is happening in hardware design,” Patterson said. “We are in a parallel revolution, ready or not, and it is the end of the way we built microprocessors for the past 40 years.”

Although the end of the microprocessor is inevitable, Patterson noted that the current movement toward parallel architectures could fail without ever achieving success. In particular, past companies based on parallel architectures have all failed. But this time, he argued, the consequences of failure would likely be more severe and widespread. Despite the history, Patterson said that he is optimistic that the parallel revolution could succeed this time, for several reasons. First, there will not be fast microprocessor alternatives to parallel architectures. Second, the open-source community will build software that takes advantage of parallel architectures. Third, emerging software trends (especially software as a service) are well suited for parallel architectures. Fourth, FPGA chips will decrease the time necessary to prototype new designs. Finally, necessity is the mother of innovation.

Of course, Patterson’s optimism was restrained, since many obstacles must be overcome before the parallel revolution can be realized. In the remainder of his talk, Patterson described several challenges, or research themes, as they relate to the systems community and the approaches being taken by the Parallel Computing Lab to solve them. The challenge that he mentioned first is that there is not yet a “killer app” for parallel architectures. Patterson argued for an application-centric solution in which researchers take cues from domain experts. So far, his research group has identified potential applications such as the re-creation of 3-D sound in ear buds, accelerators for hearing aids, image-based search, modeling of coronary heart disease, face recognition, and a parallel Web browser. Adapting single-threaded applications written in old languages was the next challenge addressed. Patterson argued that such applications can be transparently improved by identifying common design patterns that can be parallelized. Following the lead of Christopher Alexander’s book *A Pattern Language*, Patterson argued for 13 design patterns, which he called motifs, that if properly researched could improve performance for a range of applications.

Patterson’s third discussion point was about the difficulty of developing parallel software. He advocated a two-layer approach. The first layer is the efficiency layer, which would be developed by 10% of the programming population. Software at this level consists of smart and lightweight operating systems, hypervisors, and compilers that automatically compose and optimize applications. The second layer is the productivity layer, where novice programmers encode domain-specific logic in high-level languages.

The fourth challenge was to develop a scalable lightweight operating system for parallel architectures. Current virtual machine monitors are a good step in this direction.

Finally, power conservation remains an important issue, even for parallel architectures. Patterson’s group is using runtime data on power consumption and performance to inform compiler-level autotuners, the OS scheduler, and adaptable software components. This challenge is especially important for datacenters and handheld devices.

Patterson concluded by urging the systems community to seize this opportunity to reinvent “the whole hardware/software stack.” His parting words were, “Failure is not the sin; the sin is not trying.”

Andrew Tannenbaum noted that a crash every two months is not acceptable to most people, yet it seems to be the best that we can do with sequential programming. Since parallel programming is harder by at least an order of magnitude, how will we create software that satisfies user demands for reliability? Patterson agreed that reliability is an important problem for parallel software. He suggested revisiting software solutions that were proposed for previous parallel architectures and emphasized that a solution is critical for the parallel revolution to be successful.

Rik Farrow complimented Patterson’s research agenda and broad vision. He suggested that the systems community should also consider redesigning basic primitives, such as the operating system’s trapping mechanism and methods for inter-processor communication. Patterson agreed and noted the need for cooperation between the systems and architecture community in optimizing such primitives.

Jeff Mogul wondered whether Patterson's approach would fit the needs of the common developer. In particular, Patterson's motifs seemed to reflect the patterns in scientific computing and not necessarily everyday applications. Patterson argued that the motifs do cover a wide range of applications. But he noted that motif-based research is just underway, and the real benefit will be evident as more applications are developed for parallel architectures.

WEB AND INTERNET SERVICES

Summarized by Tom Clegg (tom@tomclegg.net)

- ***Handling Flash Crowds from Your Garage***
  Jeremy Elson and Jon Howell, Microsoft Research

Jon Howell began by observing that a single server in your garage can provide enough power to deploy a cool new Web application and make some money with minimal startup costs. However, if your service gets popular too suddenly, the burst of traffic can easily bring down your garage server completely. Utility computing services make it possible to accommodate flash crowds cheaply by adding servers on short notice and turning them off when they're no longer needed. Howell presented a survey of techniques for using utility computing to achieve load balancing and fault tolerance for Web services.

The survey covered four basic approaches: storage delivery networks, HTTP redirection, middlebox load balancing, and DNS load balancing. Each technique was evaluated using five criteria: applicability to different types of applications, limits of scalability, implications for application development, response to front-end failure, and response to back-end failure.

Storage delivery networks are easy to use and are suitable for serving idle content such as video files. HTTP redirection works by assigning each client to a single back-end server. This client-server affinity makes application development easier, but it is possible for clients to be bound to a broken back-end server, and a front-end failure prevents any new sessions from starting. An experiment with 150 clients and 12 back-end servers resulted in only 2% load on a single front-end server, suggesting that a single redirector could handle 7,500 clients. A middlebox load balancer associates clients with back-end servers by looking at layer 4 (TCP source port number) or layer 7 (HTTP cookie). An advantage to this technique is that it does not involve the client's participation. However, a front-end server failure is fatal to all sessions. DNS load balancing assigns clients to back-end servers by selecting and reordering a list of IP addresses when responding to queries. DNS load balancing scales very well, but it is complicated by DNS caches, resolvers, and client software. Experiments showed a huge variance in failover time on different operating systems, with the Mac OS X resolver library taking up to 75 seconds to failover to a second IP address. Also, a significant portion of clients sort the list of IP addresses and contact the lowest-numbered server first, thereby defeating the load balancing system. A hybrid approach might use a static delivery network for static content and a load-balanced cluster for active content or use DNS to balance load among several fault-tolerant middlebox load balancers, which can compensate for the sluggishness of DNS failover.

Howell shared some lessons learned from a CAPTCHA service (Asirra) and a password reminder service (Inkblot-Password), both of which handled flash crowds reasonably well. The CAPTCHA service used DNS load balancing to select a back-end server, which provides a session ID so that misdirected queries can be identified and forwarded to the correct back-end server. Occasional misdirected requests were forwarded to the correct server. Some requests failed because of utility computing back-end failures, but users could simply retry. An attempted denial-of-service attack was apparently abandoned after it failed to bring down the service.

One attendee observed that the middlebox and DNS techniques have complementary characteristics; Howell agreed that it would be worthwhile to evaluate a hybrid approach using those two techniques. Another question was why DNS address list sorting didn't prevent the DNS load balancing from being effective; Howell noted that Linux accounts for a relatively small portion of clients and that the DNS servers could help work around the behavior by returning only a subset of the full back-end server list to each query. In response to another audience question, Howell said he would be able to make the survey data available to the public.

- ***Remote Profiling of Resource Constraints of Web Servers Using Mini-Flash Crowds***
  Pratap Ramamurthy, University of Wisconsin—Madison; Vyas Sekar, Carnegie Mellon University; Aditya Akella, University of Wisconsin—Madison; Balachander Krishnamurthy, AT&T Labs—Research; Anees Shaikh, IBM Research

Most Web servers rely on overprovisioning to handle flash crowds, because it is difficult to obtain data about server resource limitations. Administrators are reluctant to perform stress tests on production servers, and testbed environments are often configured so differently that test results would not be a good indicator of the production Web server's performance. Pratap Ramamurthy presented a technique for measuring resource limitations of a production Web server without adversely affecting regular usage.

The "mini-flash crowd" service employs a distributed set of clients, synchronized by a controller, to simulate flash crowds. The controller conducts a number of experiments, each designed to test the limitations of a specific resource; for example, to test network bandwidth, the clients download large static files from the target server. Each experiment begins by launching a small number of simultaneous requests and measuring the service's response time, then performing further tests with increasing numbers of simultaneous clients. The experiment stops when the response time has increased by a user-configured threshold. This prevents the experiment from having a detrimental effect on the real users of the target service.

Before conducting a series of experiments, the controller crawls the target server and classifies objects by size and type in order to select appropriate requests for the different resource tests. It also measures the round-trip response time for each client; when conducting tests, it compensates for the difference between clients so that the target server receives all of the requests within the shortest possible time interval. The service was used to test some "cooperating" target sites, whose administrators were aware of the tests and made their server logs available to the testers. These tests were conducted with a 250-ms response time threshold and the results were provided to the service operators; in some cases the results exposed some unexpected limitations and helped to diagnose known problems. Tests with a lower response time threshold (100 ms) were conducted on a number of other public Web sites in the wild. The results of these tests were categorized according to Quantcast popularity rank, which showed that the more popular sites tend to be better provisioned and accommodate bigger client loads but that even unpopular servers often have well-provisioned network connectivity. A survey of phishing sites showed that their request handling capabilities are similar to low-end Web sites (ranked 100,000–1,000,000).

In response to a questioner, Ramamurthy said that the MFC source code will be made available. Another attendee expressed curiosity about the response time curve beyond the 100-ms threshold. Ramamurthy offered that the relevance of larger response times depends on the type of application; for example, longer response times are more important for a search index than for a binary download site. Another attendee suggested that the tests cannot be considered "nonintrusive" if they affect the target service's response time enough to be worth measuring. Ramamurthy replied that the response time increases only for the short time that the test is being conducted and that 100 ms is a relatively small impact for testing servers in the wild; in effect, the choice of response time threshold is a compromise between nonintrusiveness and the likelihood that the results will be indicative of critical resource constraints. Another questioner addressed the problem of treating Web servers as "black boxes": The profiler might be measuring the performance of a load balancer more than that of the back-end servers. Ramamurthy agreed and mentioned that different types of tests can be developed to make more fine-grained inferences in the case of a "cooperating" server.

■ ***A Dollar from 15 Cents: Cross-Platform Management for Internet Services***

Christopher Stewart, University of Rochester; Terence Kelly and Alex Zhang, Hewlett-Packard Labs; Kai Shen, University of Rochester

Internet services are becoming more popular, and the datacenters that support them are becoming more complex. The use of multiple hardware and software platforms in a datacenter is commonplace. Multi-platform management can allow high performance at low cost, but choices tend to be made on an ad hoc basis because there are too many permutations of configurations to test exhaustively. Christopher Stewart presented an approach to optimizing performance using a predictive model which can be calibrated with readily available data and used to guide server purchasing decisions and make the best use of multiple platforms in a heterogeneous environment.

Often, management recommendations must be made without modifying production systems in any way; it is impossible to obtain profiling information using source code instrumentation and controlled benchmarking. Therefore, Stewart's approach relies only on data that is readily available without touching production systems. It uses trait models derived from empirical observations of production systems, together with expert knowledge of the structure of processors and Internet services. The key principle is to derive trait models from production data for hard-to-characterize platform parameters and to use expert knowledge to compose traits for performance prediction. A trait model characterizes only one aspect of a complex system: For example, a processor metric such as cache misses can be predicted from a system configuration variable such as cache size.

The effectiveness of Stewart's method was demonstrated by calibrating a trait model on one processor and using it to predict application performance characteristics on a system with a different processor. The calibrations and predictions were made for three different applications. The model offered superior accuracy over a wide range of request mixes, compared to commonly used predictors such as benchmarks and processor clock speed. As well as service time, it was able to make accurate predictions of total response time, using a previously developed queueing model which can be calibrated in production environments. Stewart discussed potential management applications, including platform-aware load balancing, in which distributing requests to the platform best configured to their architectural demands may yield better performance than the typical weighted round-robin approach.

One attendee asked whether the model's predictions were accurate for future performance as well as past performance. Stewart explained that his method was to use the first half of a month's data to calibrate a model, then compare the resulting prediction against the data from the second half of the month. He also mentioned that the predictions were tested against the first half of the month, with favorable results, although that data was not included in the paper. Another attendee wondered whether the method would suffer from the introduction of new architectures, because of the need to develop new empirical observations and not having suitable models on hand. Stewart observed that the trait models are attractive because they can be constructed cheaply; developing new models for new platforms can be done quickly enough. Stewart also clarified that the queue model refers to the application-level queue—users waiting for responses—not the operating system's run queue.

INVITED TALK

■ ***Xen and the Art of Virtualization Revisited***
Ian Pratt, University of Cambridge Computer Laboratory

Summarized by Ward Vandewege (ward@gnu.org)

The Xen project mission is to build the industry standard open source hypervisor. To maintain Xen's industry-leading performance, Xen tries to be first to exploit new hardware acceleration features and helps operating system vendors to paravirtualize their operating systems. Security is paramount to maintaining Xen's reputation for stability and quality. Xen supports multiple CPU types (e.g., x86, ia64, PowerPC, and ARM, with more to come). With its roots as a university project, Xen wants to foster innovation and drive interoperability between Xen and other hypervisors.

Virtualization is hot for a number of reasons. Virtualization allows clearing up the mess created by the success of "scale-out" caused by moving applications from big iron to x86: the so-called server sprawl with one application per commodity x86 server, leading to 5%–15% typical CPU utilization. This is a result of the failure of popular OSes to provide full configuration isolation, temporal isolation for performance predictability, strong spatial isolation for security and reliability, and true backward application compatibility. With virtualization, old applications can be run on old OSes instead of relying on less than perfect OS backwards compatibility.

The first virtualization benefits are server consolidation, manageability, ease of deployment, and virtual machine (VM) image portability. Second-generation benefits include avoiding planned downtime with VM relocation, dynamically rebalancing workloads to meet application SLAs or to save power, automated systems that monitor hosts and VMs to keep apps running, and "hardware fault tolerance" with deterministic replay or checkpointing.

Security of the hypervisor code is obviously very important, but hypervisors can also improve security in a number of ways. Hypervisors allow administrative policy enforcement from outside the OS—for instance: firewalls, IDS, malware scanning, all running outside of the Xen domU. OSes can also be hardened with immutable memory. The hypervisor also shields the OS from hardware complexity by abstracting away the complicated real world with multi-path IO, high availability, etc. Breaking the bond between the OS and hardware simplifies application-stack certification: Application-on-OS, OS-on-hypervisor, and hypervisor-on-hardware can all be certified more easily, which enables virtual appliances. Virtual hardware also greatly reduces the effort to modify or create new OSes. This opens the door to application-specific OSes, the slimming down and optimizing of existing OSes, and native execution of applications. Finally, hypervisors enable hardware vendors to "light up" new features more rapidly.

Paravirtualization means extending the OS so it is aware that it is running in a virtualized environment. This is important for performance, and it can work alongside hardware enhancements found in modern CPUs.

Memory management unit (MMU) virtualization is critical for performance. It is challenging to make it fast, though, especially on SMP. Xen supports three MMU virtualization modes: direct pagetables, virtual pagetables, and hardware-assisted paging. OS paravirtualization is compulsory for direct pagetables and is optional but very beneficial for virtual and hardware-assisted paging.

Network interface virtualization is tough to achieve. In addition to the high packet rate with small batches, data must typically be copied to the virtual machine when received, and some applications are latency-sensitive. Xen's network IO virtualization has evolved over time to take advantage of new NIC features. Xen categorizes smart NICs in levels 0 through 3. Level 0 NICs are conventional server NICs, whereas level 3 ones are more exotic, with very advanced features. Smarter NICs reduce CPU overhead substantially, but care must be taken that by using smarter NICs the benefits of VM portability and live relocation are not lost.

Xen Client is a frontier for virtualization: a hypervisor for client devices. Hypervisors on small computer systems will allow "embedded IT" virtual appliances that could run intrusion detection systems, malware detection, remote access, backups, etc., independent of the user-facing operating system.

To conclude: open source software is a great way to get impact from university research projects. Hypervisors will become ubiquitous, offering near-zero overhead and being built into the hardware. Virtualization may enable a new "golden age" of OS diversity, and it is a really fun area to be working in!

WORKLOADS AND BENCHMARKS

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

- ***Measurement and Analysis of Large-Scale Network File System Workloads***
  Andrew W. Leung, University of California, Santa Cruz; Shankar Pasupathy and Garth Goodson, NetApp, Inc.; Ethan L. Miller, University of California, Santa Cruz

Andrew Leung presented results from a three-month study of two large-scale CIFS servers at NetApp, the first trace study that analyzes CIFS servers. One server had a total storage of 3 TB, with most of it used, and it was deployed in a corporate datacenter. The other server had a total storage of 28 TB, with 19 TB used, and it was deployed in an engineering datacenter.

Andrew highlighted some of the interesting findings in the study. They found that more than 90% of active data is untouched during the three-month period. The read/write byte ratio was 2:1, whereas it was 4:1 in past studies. The number of requests is high during day and low during the night (as expected). Read/write access patterns have increased (as workloads have become more write-oriented). Some 64% of all files are opened only once and 94% of files are opened fewer than five times, with 50% of reopens happening within 200 ms of the previous open. Files are infrequently accessed by more than one client. Even when they are accessed by more than one client, file sharing is rarely concurrent and they are mostly read-only.

One member from the audience asked whether they analyzed how file sizes grew over time. Andrew replied that they did not analyze this but a significant amount of the data came in single open/close sets. He then asked about the average data transfer rate. Andrew replied that the access patterns varied a lot from one day to the next and it is hard to put down a number. In response to a question about the size of the system they studied, Andrew replied that he would call it a medium system. The Q&A session ended with a member of the audience commenting that the results should be fed back to the spec benchmarks.

- ***Evaluating Distributed Systems: Does Background Traffic Matter?***
  Kashi Venkatesh Vishwanath and Amin Vahdat, University of California, San Diego

Kashi Vishwanath posed the question, "What sort of background traffic should be used while evaluating distributed systems?" To answer this, they performed a literature survey of 35 papers from SIGCOMM, SOSP/OSDI, and NSDI from 2004 to 2007. They found that 25% of the papers did not use any background traffic to evaluate their system, 15% used simple models (constant bit rate or Poisson models) to model their background traffic, 33% employed live deployments for their measurements, and 25% used complex models for their measurements. Using their test setup, they first compared simple models for generating background traffic and swing to ascertain whether their traffic generator was responsive and realistic. They concluded that simple methods can result in significant inaccuracy and that you need traffic generators that are more realistic. Further, they evaluated the effect of background traffic on various classes of applications. They found that Web traffic (HTTP) is sensitive to the burstiness of background traffic, depending on the size of the objects being transferred. Multimedia apps are not very sensitive to traffic burstiness, as they are designed to tolerate some jitter. Bandwidth estimation tools are highly sensitive to bursty traffic. Based on these results, they concluded that applications should be evaluated with background traffic with a range of characteristics.

Someone from the audience asked whether they went back and tried to evaluate how their findings would affect the results from the papers in their literature survey. Kashi replied that they did do that and found that some applications changed quite a bit with the amount of background traffic.

- ***Cutting Corners: Workbench Automation for Server Benchmarking***
  Piyush Shivam, Sun Microsystems; Varun Marupadi, Jeff Chase, Thileepan Subramaniam, and Shivnath Babu, Duke University

Piyush Shivam presented this paper. Their goal was to devise a workbench controller that plans the set of experiments to be run based on some policy, acquires resources and runs the experiments, and further plans the next set of experiments to be run based on the results. The challenge is to do this efficiently (i.e., running as few experiments as possible) while achieving statistical significance. As an example they use finding the peak rate on a Linux NFS server and present various algorithms and policies for doing this (strawman linear search, search, binary search, linear, and model guided). Their results show that their automated workbench controller achieves their goals at lower cost than scripted approaches that are normally used.

A member of the audience commented that using the peak load is misleading and that the median case is more important. He then asked whether they tried varying the workload mix. Piyush replied that the peak was just an example they used in the paper and that you could try varying the workload mix. Next, Piyush was asked what happens when the parameter space explodes. Piyush replied that the response surface method lets you choose only 2% of the overall possible space.

SECURITY AND BUGS

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

- ***Vx32: Lightweight User-level Sandboxing on the x86***
  Bryan Ford and Russ Cox, Massachusetts Institute of Technology

  Awarded Best Student Paper!

Russ Cox presented Vx32, a lightweight sandbox for the x86 architecture. Vx32 is not OS- or language-specific, but it is tied to the x86 architecture. Most x86 OSes don't use all segments, and users can create their own segments. Vx32 takes advantage of this and runs the code to be sandboxed natively in its own segment. But the sandboxed code can change the segment registers. Vx32 prevents this by using dynamic instruction translation and rewriting code to a "safe" form. They evaluated Vx32 by running various benchmarks and by building four applications. For benchmarks, the overheads are low when there are no indirect branches (i.e., no instructions to be translated). The applications that they built were an archival storage system, an extensible public-key infrastructure, a port of the Plan 9 OS on top of a commodity operating system, and a Linux system call jail. The first two applications have between 30% slowdown to 30% speedup compared to native execution. Linux jail has an 80% overhead.

A member of audience asked what they planned to do about 64-bit systems as they do not have segmentation registers. Russ replied that they can switch to a 32-bit mode while running Vx32's 32-bit code segments. Next, Russ was asked whether Vx32 lives in the same segment as the code being sandboxed. If so, could self-modifying code attack it? Russ replied that the translated code lives in a different segment than Vx32. Lastly, Russ was asked how Vx32 was different from a binary instrumentation tool such as Pin. He replied that Vx32 is much faster than in Pin; you can either get performance or safety but not both.

- ***LeakSurvivor: Towards Safely Tolerating Memory Leaks for Garbage-Collected Languages***
  Yan Tang, Qi Gao, and Feng Qin, The Ohio State University

Memory leaks can occur even in garbage-collected languages such as Java and C#. One reason is that programs keep pointers to objects they don't use anymore. For long-running programs, this results in performance degradation as they take up more and more heap space and eventually crash the program. Their system, LeakSurvivor, identifies such "potentially leaked" (PL) objects and swaps them out from both virtual and physical memory. They replace references to PL with a unique kernel reserved address. Access to these addresses will result in a swap-in. They also maintain an index that keeps track of all outgoing pointers to an object. They implemented LeakSurvivor on top of Jikes RVM 2.4.2. They evaluated their system by running it with programs that had known memory leaks (Eclipse, Specjbb2000, and Jigsaw). Eclipse and Specjbb survive with good performance for much longer than they do without LeakSurvior. Jigsaw, even though it runs for much longer with LeakSurvior, eventually crashes because their leak detector could not detect "semantic leaks" present in Jigsaw. The overhead of LeakSurvivor is low (2.5%) when it is running programs that don't have leaks.

In response to whether they can meet QoS guarantees when they run LeakSurvivor on a Web server, the authors replied that they currently cannot make performance guarantees. As to whether they have to save virtual memory for a 64-bit machine, the authors explained that, in a 64-bit machine, you have infinite virtual memory and LeakSurvivor might hurt performance. When asked whether they have any plans for providing feedback to developers so that developers can fix their leaks, they admitted that they currently did not have this functionality. As to whether they had any heuristics for turning LeakSurvivor on and off, the authors replied that they currently turn it on all the time, but it is not really hard to add this function.

- ***Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing***
  Dan Wendlandt, David G. Andersen, and Adrian Perrig, Carnegie Mellon University

Dan Wendlandt presented a method to reduce the vulnerability to man-in-the-middle (MITM) attacks of some of the common protocols such as SSH and HTTPS. SSH's model of host authentication is one of "trust-on-first-use," in which the user decides whether an unauthenticated key is valid or not. This and the fact that the user must manually verify the validity of any key that conflicts with a cached key make the user very vulnerable to MITM attacks. The Perspectives approach to mitigate this is to have a bunch of notaries in the network. Instead of trusting the SSH key, a client can verify the key from the notaries. The notaries probe machines on the network and build a record of the keys used by the services over a period of time.

The notaries provide the client with spatial redundancy (observation from multiple vantage points) and temporal redundancy (observation over time). The notaries offer a better perspective to the clients and enable them to make better security decisions. Further, the client implements key-trust policies that trade off between security and availability; for example, it might accept a key even when the number of notaries that report a key is less than a quorum.

Someone asked how a client can know how many notaries are present. Dan replied that the clients can download a notary list, but he also pointed out that someone accessing a server that has just been deployed will not get temporal security. Next, Dan was asked whether Perspectives would help with the Debian bug. Dan replied that Perspectives will not help you detect bugs in the OpenSSH implementation.

- ***Spectator: Detection and Containment of JavaScript Worms***
  Benjamin Livshits and Weidong Cui, Microsoft Research

Benjamin Livshits proposed a distributed taint mechanism for detecting and containing Javascript worms. Javascript worms are hard to find and fix, as Web 2.0 technologies allow the worms to propagate themselves by generating appropriate HTTP requests. Simple signature-based solutions are insufficient, as worms are polymorphic. Their idea for detecting worms is as follows. They tag each page uploaded on the server. This tag is downloaded to clients whenever the Web page is downloaded. They also inject Javascript code so that the tags are propagated at the client side and are preserved when pages are updated. They look for worms by checking for long propagating chains. They have an approximation algorithm that is designed to scale for graphs containing thousands of nodes. They evaluated Spectator for scalability and precision by performing a large-scale simulation of MySpace and a real-life case study (on siteframe).

YuanYuan Zhou asked whether the tag should be unique or whether it can be global. Benjamin replied that it really is not an issue and that it can be global. YuanYuan then asked if the tags can be removed by the worms. Benjamin replied that they generally cannot be removed, as the tags are HTML, but there are some particular cases of worms where it can be a problem.

INVITED TALK

Summarized by Zoe Sebepou (sebepou@ics.forth.gr)

- ***Using Hadoop for Webscale Computing***
  Ajay Anand, Yahoo!

Ajay Anand described their experiences using Apache Hadoop and what led them to start developing this product. He started his talk by stating the problem Yahoo! has in collecting huge amounts of data, implying petabytes of storage capacity and a vast number of machines to deal with the processing of this data in a secure and accurate manner, while avoiding hardware outages.

Hadoop constitutes an open source implementation of a Distributed File System (HDFS) and a map-reduce programming model combined in one package. Hadoop is designed to support many different applications providing them with the required scalability and reliability, which otherwise would be extremely costly to implement in each application. Hadoop is written in Java so it does not require any specific platform. Its main components are a Distributed File System based on the architectural characteristics of the Google File System (GFS) and a Distributed Processing Framework based on the map-reduce paradigm.

Hadoop architectural characteristics include many unreliable commodity servers and one single metadata server, but it ensures reliability by replicating the data across the available data servers. Because the system was designed for the requirements of their environment and in general for Web-scale applications that make simple sequential access involving one writer at a time and as a consequence do not require strict locking features, Hadoop receives performance advantages from the simplicity of its design. Indeed, the core design principle behind Hadoop is to move the computation as close to the data as possible; processing data locally is definitely more effective than moving the data around the network.

HDFS, which is Hadoop's file system, operates using two main components: The name nodes keep information about the files (name, number of replicas, and block location); the data nodes provide the actual storage of the data. The files in HDFS are striped across the available data servers and are being replicated by a settable replication factor to avoid unavailability resulting from node failures. HDFS keeps checksums of the data for corruption detection and recovery. Every time someone requires access to a specific file, it contacts the name nodes and, after obtaining information about the exact location of the data, it directly acquires the data from the data nodes. In case of a data-node failure, the name node detects it by periodically sending heartbeats to the data nodes. After a failure, the name node chooses a new data node to store new replicas. With the use of checksums, the clients can identify data corrupted by a node outage and ask some other available data node to serve their request. However, name-node outage still remains a single point of failure.

Ajay continued his talk by analyzing the map-reduce technique used by Hadoop to enhance the system's performance by providing efficient data streaming by reducing seeks. The map-reduce mechanism follows a master-slave architecture. Specifically, the master, called Jobtracker, is responsible for accepting the map-reduce jobs submitted by users, assigns map-reduce tasks to the slaves, called Tasktrackers, and monitors the tasks and the Tasktrackers' status in order to reexecute tasks upon failure. The Tasktrackers run map-reduce tasks upon instruction from the Jobtracker and manage the storage and transmission of intermediate outputs. Ajay pointed out that some future improvements are still to be made in the map-reduce mechanism; Yahoo! is currently working on these issues. He explained that Hadoop still does not have an advanced scheduling system. The slaves of the map-reduce framework can manage one or more jobs running within a set of machines and the mechanism does work well for dedicated applications; however, in the presence of shared resources their mechanism would not be sufficient. Consequently, he described the Pig programming environment, an Apache incubator project initiated by Yahoo!. Pig is a high-level, easy-to-use dataflow language used to generate map-reduce jobs and provides extensible data processing primitives.

Ajay concluded his presentation with the current uses of Hadoop inside and outside the Yahoo! environment, also providing measurements depicting the advantages gained by using the system.

Bar Kenneth from VMware asked whether Yahoo! had considered exploring the use of virtual machines to solve problems with loss of data locality in Hadoop. The reply was that in fact virtualization is an issue they are very interested in and that they will be exploring this possibility. Moreover, their goal is to be able in the future to say that the *job* is the VM and what they actually want is to be able to replicate the jobs across the machines.

Rik Farrow wondered, if the name nodes are really critical for Hadoop, why there is no high availability for them and why they have yet to develop a mechanism to support this feature. Ajay answered that this issue is on the list of their things to do but is not at the top because most of what they are running are batch jobs and not online operations. In addition, their main priority is to enhance other things such as the scheduling mechanism to provide name-node balancing.

A second question from Rik Farrow was whether Hadoop has a shared memory architecture. The answer was that it does not. In fact, each computer node has its own memory and this memory is not shared across machines.

A questioner from Sun Microsystems asked about the algorithms used for chunking and data distribution, as well as for the fault-tolerance mechanism and the load balancing of the data placement in Hadoop. Ajay explained that the basic concept is to have three replicas, two within a rack and one outside, to spread things around as much as possible inside their environment. The same questioner asked about the communication protocol between the HDFS clients and the name nodes of Hadoop, wondering whether there is a separate path for the metadata communication and the heartbeat messages. The reply was negative; in Hadoop all the communication is taking place through the same network, without any isolated network for metadata purposes.

Another questioner asked about bottlenecks in the network bandwidth, the disk bandwidth, or the CPU utilization of their system. The speaker said that at Yahoo! they try to collect data and to do more and more profiling to identify the bottlenecks. The main bottlenecks already observed are the network and the memory in the name-node side.

In response to an additional question about how Hadoop handles a global failure and how things return to normal again, Ajay replied that Hadoop continues working in the case of node failures as long as they are not name nodes. To the final question of how many times and how often they have to upgrade their system, the answer was that in the case of upgrade everything has to come down; usually they upgrade the system once a month, with the whole process taking less than four hours.

MEMORY AND BUFFER MANAGEMENT

Summarized by Varun Marupadi

- ***A Compacting Real-Time Memory Management System***
  Silviu S. Craciunas, Christoph M. Kirsch, Hannes Payer, Ana Sokolova, Horst Stadler, and Robert Staudinger, University of Salzburg

Modern memory managers lack predictability—the time to allocate a chunk is dependent on the global memory state. In addition, fragmentation of memory is not dealt with well. To address these shortcomings, Silviu Craciunus and his co-authors have developed Compact-fit, a memory management system that responds in linear time to the size of the request and is able to trade off performance for lower levels of fragmentation.

The system works by dividing objects into differently sized "classes." Within each size class, there is allowed to be only one partially filled page. This allows quick (linear time) deallocation, since exactly one object needs to be moved per deallocation. The authors present two implementations—one actually moves data in physical memory when an object is freed (the "moving implementation") and the other manages an indirection table that allows only table information to be changed without moving the data itself. In either implementation, by increasing the number of pages that may be partially filled, some performance may be gained at the expense of more fragmentation.

Experimental evaluation shows that allocation and deallocation times show good fidelity with the theoretical predictions, but they are slower than existing memory allocators, owing to the overhead of managing fragmentation. Compact-fit is able to allocate objects effectively even with high levels of fragmentation. In response to a question, the authors said that Compact-fit will not move objects from one size class to another at the current time.

- ***Prefetching with Adaptive Cache Culling for Striped Disk Arrays***
  Sung Hoon Baek and Kyu Ho Park, Korea Advanced Institute of Science and Technology

Sung Hoon Baek and Kyu Ho Park study the neglected field of prefetching schemes for striped disk arrays. Prefetching from striped disks has several new problems, including loss of expected parallelism owing to short reads, nonsequential short reads, and the absence of cache management for prefetched data.

To manage these risks, the authors present Adaptive Stripe Prefetching (ASP), which uses new schemes for prefetching an entire stripe when a request for a block comes in, adaptive culling of the cache to preferentially evict prefetched blocks that have not been requested, and an online model to tune the ratio of prefetched to cached blocks to maximize the total hit rate.

The system was evaluated with a variety of benchmarks. It performs as well or better than any existing prefetching schemes. A question was raised regarding the performance of the system in the presence of writes. The response was that the system is primarily focused on read-heavy workloads but should work in the presence of writes as well. It was also pointed out that one of the benchmarks (Dbench) simulates a read-write workload.

- ***Context-Aware Prefetching at the Storage Server***
  Gokul Soundararajan, Madalin Mihailescu, and Cristiana Amza, University of Toronto

A problem with today's prefetching schemes is that they break down under high levels of concurrency because it is hard to detect access patterns when requests from many sources are interleaved. To address this, Gokul Soundararajan and his colleagues presented QuickMine, a system that allows application contexts to be visible to the storage server so that it can more accurately detect access patterns.

Every block request is tagged with an identifier corresponding to a higher-level application context (Web, database, or application). It is claimed that this is minimally intrusive and easy to create for any application. However, it does require minor modifications to the source code. Mining the context-tagged requests can generate block correlations for both sequential and nonsequential accesses. The system was evaluated by modifying the MySql database to pass context information and running a number of three-tier Web-based applications on it. For all benchmarks, the cache miss rate and latency were drastically reduced by using QuickMine.

In a lively question session, several attendees asked about extending the work to other contexts. In particular, file-based storage rather than block-based storage could be dealt with by using the filename+offset rather than the block number. Extension to other applications requires only localized instrumentation changes. Extension to other classes of applications would be more intrusive and is a topic of ongoing research. Schemes using fuzzy contexts or machine learning techniques to infer context could be used and are worth exploring, but Gokul believes context is still necessary, because very different queries follow the same code path through libraries.

INVITED TALK

Summarized by Matthew Sacks (matthew@matthewsacks.com)

- ***Google Hacking: Making Competitive Intelligence Work for You***
  Tom Bowers

Tom Bowers takes the ideas presented in Johnny Long's book *Google Hacking* and applies the concepts of using Google as a hacking utility for servers and other machine-related vulnerabilities to information in and of itself. The amount of information that can be gathered using the world's largest database is astounding. Tom went on to demonstrate how to gather information about a particular organization or individual by leveraging unconventional techniques for using the Google search engine.

By using Google as a utility for competitive intelligence, one can find out a wealth of information about competitors, as well as seeing what type of information is being leaked about the individuals in a company or organization and the organization itself. 80% of all competitive intelligence is done through public sources. Also, the U.S. Supreme court has ruled that information found on Google is public information.

Tom also presented the basic method for performing competitive intelligence using Google by building a competitive intelligence profile.

As an example, using Google Earth Pro (which provides more frequent updates than the standard Google Earth), Tom can map out a competitor's facility to determine where he might be able to gain easy access. From there he could use wireless scanning techniques to access the competitor's data from unsecured wireless networks. Also, using Google hacking Tom showed that a large majority of Web cams are available through the public Internet from a standard Google search!

In this talk Tom revealed the world of competitive intelligence and its primary information-gathering utility: Once a competitive profile has been built, the job of gathering additional detailed information becomes rather simple. Most of the work done in competitive intelligence can be done from one's own office or home.

WIDE-AREA SYSTEMS

Summarized by Varun Marupadi (varun@cs.duke.edu)

- ***Free Factories: Unified Infrastructure for Data Intensive Web Services***
  Alexander Wait Zaranek, Tom Clegg, Ward Vandewege, and George M. Church, Harvard University

Alexander Zaranek and his colleagues explained that this work was initiated to help process the large amounts of data needed to sequence human genomes. A free factory is a set of several 12- to 48-node clusters, some of which are co-located with data-acquisition instruments. The clusters are connected via relatively slow networks. A free factory runs *freegols*, which are application-centric virtual appliances that run within a free factory. Different users use and develop different freegols for their particular needs.

A portion of the cluster's resources is configured as warehouse instances, which provide processing, cache, and storage services. The remainder of the resources hosts Xen virtual machines for hosting freegols. The storage services within a cluster are implemented as a three-tier hierarchy:

a memory cache, a distributed block cache, and a long-term archival storage service.

More information can be found at factories.freelogy.org.

- ***Wide-Scale Data Stream Management***
  Dionysios Logothetis and Kenneth Yocum, University of California, San Diego

Dionysios Logothetis presented Mortar, a platform for building queries across federated distributed systems. Such queries are useful for remote debugging, measurement, application control, and myriad other uses. Mortar allows operators to aggregate and process data within the network itself, building multiple overlays to process data from remote sources.

Mortar builds a set of static overlay trees that overlap in order to tolerate node and network failures. By carefully building trees, it is possible to generate routes that are network-aware and resilient at the same time. Mortar avoids problems arising from static clock skew by using relative time offsets rather than absolute timestamps. By isolating data processing from data routing, it is possible to use aggregate operators that are not idempotent or duplicate-insensitive. By using multiple static overlay trees, Mortar is able to make progress when as many as 40% of the nodes have failed.

Questions were raised about how queries that require knowing the source of the data could be implemented. Dionysios replied that such queries are problematic because of the nature of aggregation itself. Other attendees wondered whether the system might fail from corner cases in the heuristics and static tree-based routing. Dionysios explained that the effect of topology on the system has not yet been fully studied, so it is hard to give a definite answer.

- ***Experiences with Client-based Speculative Remote Display***
  John R. Lange and Peter A. Dinda, Northwestern University; Samuel Rossoff, University of Victoria

John Lange presented work on speculatively executing window events on a remote display. The goal is to reduce the user-perceived latency when using a remote service. The predictability of events sent by VNC and Windows Remote Desktop was presented; VNC appeared to be much more predictable than RDP. John says that this may be primarily due to the higher level of abstraction that RDP uses, along with the much lower event rate. A Markov model was used to predict future events based on past events and user input. This also allowed control over the tradeoff between accuracy and latency.

A user study was presented for VNC prediction. Although not conclusive, the study did show that users are at least moderately accepting of display errors during misprediction. A question was asked about what constitutes an error. John explained that an error may be anything from garbage on the screen to subtle artifacts in the window. Another attendee asked about overhead. John replied that, after training, there was almost no CPU overhead but there was some memory overhead.