conference reports

THANKS TO OUR SUMMARIZERS

2007 USENIX Annual Technical Conference

Santa Clara, CA
June 17–22, 2007

KEYNOTE ADDRESS

- ***The Impact of Virtualization on Computing Systems***
  Mendel Rosenblum, Stanford University
  Summarized by Francis David (fdavid@uiuc.edu)

Mendel Rosenblum began by describing what virtualization means and identified several important properties that can be attributed to a virtualization system. Hardware is multiplexed or partitioned among multiple virtual machines. Isolation and encapsulation provided by a virtual machine decouple it from the hardware and allow for operations such as suspending and resuming a virtual machine or migrating it from one physical computer to another.

VMWare Workstation now supports some interesting new functionality. It is possible to record and play back execution events in a virtual machine with low overhead, something that may prove invaluable for debugging systems. Virtual Infrastructure is yet another product that supports suspending, migrating, and resuming of virtual machines. Virtualization fuels a changing view of hardware in a data center. Compared to a traditional architecture with individual services tied to individual machines, virtualization enables a virtual server environment where all physical hardware is pooled together, allowing for quick and easy reconfiguration of resources available to a service in response to the load experienced. High availability is achieved because the loss of a physical server can be compensated by running virtual servers on the remaining hardware. Consolidation of virtual machines onto a minimal set of physical machines results in significant energy savings, because the unused physical machines may be powered down. Most of these features are automatically managed by VMware's product and are extremely easy to configure, involving a single check box in a GUI in most cases.

Modern operating systems have evolved to achieve the goal of supporting as many applications as possible while simultaneously providing security, reliability, manageability, and good performance. These driving forces, together with the need for innovation, have resulted in complex operating systems. Virtualization technology has resulted in a change in the view of the role of an operating system. An operating system running in a virtual machine doesn't need complex hardware

management and does not have to support a broad range of applications. The OS starts to look like a library that is bundled together with an application, and these virtual machines are termed virtual appliances. This concept is expected to have a big impact on the way software is distributed. Work on the Terra system at Stanford focuses on using different operating systems to support different classes of applications in a virtualized environment. Thus, virtualization provides us with renewed opportunities to improve the efficiency, reliability, and security of system software.

In response to a question about increases in complexity of the virtual machine monitor because of the need to add more device drivers into it, Mendel stated that they expect to build better systems now that they have learned from past mistakes. Also, exploiting new hardware features such as an IO-MMU allows VMWare to reduce the number of drivers that affect the reliability of the system. Andrew Tanenbaum argued that virtualization technology bears significant similarity to microkernels. I asked Mendel whether virtualization has a place in a desktop computer and whether the need for sharing will result in the breakdown of the barriers erected between virtual machines. He replied that the significant sharing among applications today may have been a bad idea to begin with and starting all over again using virtualization may be a reasonable approach going forward. Rik Farrow questioned whether running applications within a VM really provided any additional security to the application and whether using a virtual appliance was the right granularity for a protection domain. Mendel said that in many cases it is not.

TRICKS WITH VIRTUAL MACHINES

Summarized by Francis David (fdavid@uiuc.edu)

- ***Energy Management for Hypervisor-Based Virtual Machines***

  Jan Stoess, Christian Lang, and Frank Bellosa, University of Karlsruhe, Germany

Jan Stoess started the presentation by highlighting the importance of energy management in operating systems. Although there are several existing approaches to OS-directed energy management, modern virtual machine (VM) environments present a unique challenge to energy management because of their distributed and multilayered software stack. A distributed energy accounting scheme is proposed that delegates energy management to a host-level component and a guest-OS-level component.

The host-level energy manager collects energy usage information from all device drivers and tracks energy usage for individual VMs. The guest OS energy manager uses virtualized energy accounting to provide fine-grained application-level energy management. In a prototype implementation using L4 and paravirtualized Linux guests, energy accounting was performed for the CPU and disk. CPU energy consumption was estimated by using processor performance counters for various events. A weighted sum of the various counted events is used to obtain the processor energy consumption. Disk energy accounting uses a time-based approach. Request transfer time is used to directly compute energy usage. Both the CPU and disk energy models also account for idle usage.

Recursive, request-based energy accounting is used to accurately measure the energy spent by each virtual machine. This is important because a virtual device may map to multiple physical devices. For example, energy consumption of a virtual disk is a combination of the energy consumption of the physical disk and the CPU energy consumption of the virtual disk driver. The hypervisor also supports throttling of CPU allocation and shaping of disk requests to regulate energy consumption of virtual machines.

Experiments show that the energy accounting models are quite accurate and that host-level and guest-level enforcement of energy constraints works well. In the future, Jan hopes to support fully virtualized systems and multimodal devices such as multispeed disks.

A member of the audience pointed out that the disk energy accounting model does not consider spin-up and spin-down costs. In response to a question about energy constraints not resulting in fair sharing of time, Jan stated that it probably makes more sense to change scheduling algorithms to provide fair and managed sharing of energy rather than time.

- ***Xenprobes, a Lightweight User-Space Probing Framework for Xen Virtual Machine***

  Nguyen Anh Quynh and Kuniyasu Suzaki, National Institute of Advanced Industrial Science and Technology, Japan

Nguyen Anh Quynh presented research on Xenprobes, a framework for probing inside virtual machines. The probing framework is based on the Xen virtualization system and allows users to register probe handlers for arbitrary instruction addresses in a Xen virtual machine. For example, a handler can be registered for the `mkdir` system call on a Linux virtual machine to intercept all such system calls on the Linux VM.

Xenprobes exploits the debugging architecture of Xen to inject software breakpoints into the probed VM. Unlike the Kprobes framework, which is tied to Linux, Xenprobes can work with any guest OS supported by Xen. Also, probe handlers can be written in userspace instead of in kernel-space as required by Kprobes. Two types of probes are supported. An XProbe is a pair of handlers that are called just before and just after a probed instruction is executed. It is possible to register null handlers. XrProbe handlers are invoked at the beginning of a function and when it returns. XrProbes allow for examining function call arguments and return values. All probes have full access to VM memory.

The probing process starts with probe registration, which is managed by the framework. Multiple probes at the same address are supported. Probes can be enabled and disabled individually and also from within other probe handlers. All probes are removed when the VM shuts down. Microbenchmarks show that injecting probes into a VM causes a large overhead; the null syscall takes 400 times longer when using an XrProbe and 180 times longer when using an XProbe with one handler. A macrobenchmark that examined the time to decompress the Linux kernel with probes placed in the mkdir, chmod, and open syscalls has more reasonable performance. XProbe and XrProbe experience 6% and 39% increases in execution times, respectively.

A member of the audience pointed out that the code-modifying approach used by the probing framework for breakpoints will not support OS code that checksums itself for verifying integrity.

- ***Virtual Machine Memory Access Tracing with Hypervisor Exclusive Cache***

  Pin Lu and Kai Shen, University of Rochester

Pin Lu addressed the issue of estimating the optimal amount of memory that should be allocated to virtual machines (VMs). The hypervisor does not normally have access to information about memory usage inside a VM. In order to obtain information about VM memory usage, Pin advocates that part of the memory that would normally be allocated to the VM be used as a hypervisor cache of VM memory. The cache is designed to contain items that are evicted from the VM memory (exclusive cache). It can be shown that when using LRU for replacement, the hypervisor cache approach does not incur any extra misses when compared to the normal allocation of that memory directly to the VM.

The use of the hypervisor cache enables the use of cache miss ratio curves for direct memory allocation to virtual machines. In particular, this scheme allows the determination of cache miss behavior when using memory sizes that are less than the current allocation. The memory allocation to individual VMs is dynamically adjusted, with the objectives of minimizing the geometric mean of each VM's miss ratio and ensuring that there is a bounded performance loss for each VM when being allocated less memory than its baseline allocation.

The proposed design is not fully transparent, as it requires that the OS notify the hypervisor when there is a page eviction from VM memory. Also, there is some overhead because of page copying and management of the cache. A prototype system was constructed using the Xen virtual machine. Experiments show that the throughput degradation for several non-CPU-bound workloads is less than 20%. Experiments also show that the miss ratio curve prediction is reasonably accurate as well. In a multi-VM experiment, the system correctly reallocates memory to substantially reduce the systemwide page-miss metric.

Pin compared his work with VMWare's ESX server. VMware's ESX server only estimates the working set size and uses that measurement to allocate memory. Experiments show that accessing a large segment of memory sequentially without any reuse causes ESX server to allocate large amounts of memory even though the extra memory allocation does not reduce cache misses.

INVITED TALK

- ***Life Is Not a State-Machine: The Long Road from Research to Production***

  Werner Vogels, VP and CTO, Amazon.com

  Summarized by Ramakrishna Kotla (kotla@cs.utexas.edu)

Werner Vogels started this talk by introducing himself as a "recovering academic," to emphasize his theme: Why it is hard to take research ideas into production systems, and what can be done about it? His talk mainly focused on issues involved in building and deploying large-scale distributed systems while presenting examples in a wide range of fields to draw analogies.

Werner noted that incremental scalability is the key to building and deploying production-grade large-scale distributed systems such as the one that they have at Amazon.com, which supports seven Web sites with 63 million customers, 1.1 million active sellers, 240,000 Web service developers, 14,000 employees, and 20 fulfillment centers. He defined scalability of a system as the ability to: (1) provide increased performance with increasing resources; (2) provide always-on service; (3) handle heterogeneity; (4) be operationally efficient; (5) be resilient to failures; and (6) be cost-effective as the system grows.

Werner stressed the point that it is very hard to take research ideas into production, presenting many examples and explaining the reasons for this problem. He noted that it usually takes about 20 years before a research idea gets adopted into mainstream systems—for instance, the spreadsheet and the Web. He then explained how hyperlinks and markup languages were developed in the mid-1960s, TCP/IP-based networks came to life in the 1970s, and it wasn't until the mid-1990s that the combination of these three turned into the Internet and Web that we use today.

He then pointed out several important reasons behind the slow adoption of research ideas: (1) making unrealistic assumptions about system characteristics and the environment; (2) not accounting for uncertainty in the real world; (3) multiple differing research approaches in solving a problem.

He pointed out that research in the academic world focuses on the details of the technology itself and is not very focused on the application context of the technology,

which results in slow or no adoption of these ideas in production systems. For example, many research approaches assume failures are uncorrelated, whereas correlated failures do happen in reality. Next, he explained that many systems fail to take into account the uncertainty that is present in large-scale systems that are complex and nondeterministic in nature. Finally, he pointed out that having differing and competing approaches in the research community makes it harder for system builders to choose the right approach to solve their problems.

Werner suggested that systems should involve fewer assumptions and explicitly account for uncertainty so that they can be easily adopted and deployed in real-world applications. He suggested using Occam's Razor to select the approach with fewest assumptions when there are competing approaches.

Werner Vogels maintains his personal blog at http://www.allthingsdistributed.com.

NETWORK MONITORING AND MANAGEMENT

Summarized by Andrew Baumann (andrewb@cse.unsw.edu.au)

- ***Hyperion: High Volume Stream Archival for Retrospective Querying***

  Peter Desnoyers and Prashant Shenoy, University of Massachusetts

  Awarded Best Paper!

Peter Desnoyers presented this paper on Hyperion, a high-performance packet monitor that keeps a packet history, enabling new capabilities in network forensics and management. The primary challenge for the system was handling high packet rates while supporting online querying and indexing and running on commodity hardware.

To support the unusual workload, which required guaranteed write throughput but also included read activity, the authors implemented a specialized filesystem named StreamFS. StreamFS is log-structured but does not use a cleaner; instead, it performs its own data aging simply by overwriting old data as needed. It also interleaves slower and faster tracks on the disk to achieve balanced write speeds.

To support fast querying, Hyperion uses a multilevel index structure based on signatures. Use of the Bloom filter signature ensures that signature queries have no false negatives. Finally, a distributed index layer is provided to distribute summaries of index data to other nodes, allowing for faster queries in a multinode network monitoring system.

Benchmarks show that the full system runs on commodity hardware with negligible packet loss up to 175,000 packets per second, at which point it became CPU-bound. With faster hardware, the authors expect to achieve even higher throughput. The questions that were asked focused mainly on possible optimizations to and uses for the system.

- ***Load Shedding in Network Monitoring Applications***

  Pere Barlet-Ros, Technical University of Catalonia; Gianluca Iannaccone, Intel Research Berkeley; Josep Sanjuàs-Cuxart, Diego Amores-López, and Josep Solé-Pareta, Technical University of Catalonia

This paper, presented by Pere Barlet-Ros, covered the problem of how to manage overload in continuous network-monitoring applications. These applications are difficult to construct, owing to the unpredictable nature of network traffic, the increasing processing requirements involved, and the problem of efficiently handling extreme overload situations, which is necessary because overprovisioning is not feasible. The general approach is to shed load when necessary, to get the best possible quality of query results.

The system they have developed extends Intel's CoMo (continuous monitoring), which is a modular passive monitoring system. Modules may perform any amount of processing work. The problem is to manage the CPU usage of the whole system while treating modules as black boxes. The load-shedding scheme automatically finds correlations between network traffic features and the CPU usage of query modules, uses those correlations to predict the CPU load of future queries, and uses that prediction to guide load shedding.

About 50 query-agnostic network traffic features are used; they are lightweight and have a deterministic worst-case computational cost. A linear regression is performed to correlate features to observed CPU load. This regression is expensive to compute, so a feature-selection algorithm is used to remove irrelevant and redundant predictors, resulting in two or three relevant features per query.

Load shedding occurs when the total CPU load predicted for all queries exceeds the available cycles; each query can select packet- or hash-based flow sampling when load shedding is required. Results show that the predictive load-shedding mechanism is much better than the usual reactive load-shedding. Future work includes developing load-shedding techniques for queries that are not robust against sampling, and applying similar techniques to manage memory and disk load. The source code is available from http://loadshedding.ccaba.upc.edu/.

- ***Configuration Management at Massive Scale: System Design and Experience***

  William Enck and Patrick McDaniel, Pennsylvania State University; Subhabrata Sen, Panagiotis Sebos, and Sylke Spoerel, AT&T Research; Albert Greenberg, Microsoft Research; Sanjay Rao, Purdue University; William Aiello, University of British Columbia

William Enck presented this work on managing complex router configuration files at massive scale, such as at a large ISP. Precise specifications for individual routers are hard to create, and the level of complexity can be overwhelming. This leads to cut-and-paste errors. Current automated solutions such as network management tools

don't account for so-called dirty input data, which can result in a syntactically correct configuration but incorrect operation. The proposed solution is to use a template language to establish consistent device configurations and to follow an iterative approach, where users validate the data inputs. This has been implemented in a tool named PRESTO.

In PRESTO, code snippets for router configuration scripts are defined in active templates, which are close to the system's native configuration language. However, PRESTO is also a framework for data modeling. The information used to evaluate configuration templates is stored in an SQL database; as well as simple value expansion, template code can be repeated for multiple database rows, and database values can also be used as conditions and arguments to functions. The master template is constructed from a series of "configlets," allowing composition of configurations. To manage the dirty data problem, a two-step architecture allows users to first supply the input data and then validate the output.

The PRESTO tool is now in use for many applications, and anecdotal evidence suggests that the two-step process helped to solve the dirty-data problem. Asked whether a better router command language would obviate the need for such a tool, William responded that centralized control and a multistage process to avoid dirty data would both still be necessary, and that the use of templates allowed network architects to select specific optimizations for different situations.

INVITED TALK

Exploiting Online Games

Gary McGraw, Cigital

Summarized by Jan Stoess (stoess@ira.uka.de)

Gary McGraw described the difficulties of attaining security for online games. Because online games are distributed programs that rely on client-side software, securing them remains a challenging problem. Gary referred to the problem as the "trinity of trouble": online games usually have a permanent connection to the Internet, they are complex entities consisting of massively distributed code, and they typically evolve on the fly in completely unexpected directions.

Gary emphasized, however, that online games act as a bellwether for other distributed systems, since they have a large and growing user community, and since they have become a considerable economic factor. World of Warcraft, the most widely used online game, is typically being played by hundreds of thousands of users simultaneously. With eight million registered users worldwide, the revenue from subscriptions alone already adds up to $1.3 billion, let alone the price to purchase the game itself. There also exists a significant middle market for selling transferable game items such as game weapons or skills. As a result, cheating and breaking online games has an economic aspect, since it allows traders to shortcut the tedious acquisition of valuable items.

Gary then asked whether discussing exploits publicly was actually a bad idea, given that it may spread abusable knowledge, and given that people may suffer from consequences for publicizing exploits, such as Dan Farmer, who was fired by SGI for releasing the SATAN suite. Gary argued that such discussion was still necessary, because otherwise the mechanisms to secure online games would be inefficient. To his own regret, the public is interested mostly in the breaking of systems, less so in their securing.

Gary then explained that current legislation mostly aims at counteracting privacy rather than preventing fraud. However, game-cracking used to be a serious problem and can nowadays easily be prevented via online verification. The current laws are therefore inappropriate to counter online fraud; in fact, game cheating is not even considered illegal. As a result, game companies typically use end-user license agreements (EULAs) with special clauses against fraud. But EULAs are problematic, since no one really reads them, and since many companies use egregious types of EULAs, whose conformity with the laws is more than questionable: Sony's EULA allows a rootkit to be installed on the client; Blizzard allows monitoring by means of spyware; Gator even disallows the removal of the software.

Gary then again illustrated the emerging economic importance of online games: The game market is expected to grow from $6 billion in 2005 to $12 billion by 2010. Games have become a field of study for economists, and there exist exchange rates converting the virtual currency of online games into real currencies. There also exist thriving secondary markets, where game items are sold for prices up to $100,000, and where over half a million Chinese people earn regular income as game item providers, sleeping in cots near the computer between their work shifts.

Gary then explained how game cheats work. There are basically two kinds of techniques, exploits and bots. Exploits leverage game bugs to induce unintended game behavior (e.g., teleporting). Bots perform legal inputs but in an automated fashion, to allow the acquisition of game items that would otherwise require tedious human interaction. Simple bots inject keystrokes and mouse movements to repeatedly perform an action until a certain skill has been achieved. More complex tools directly read or write memory locations, inject cheating functionality into system libraries, or hijack the game's system thread to call internal functions directly. Some state-of-the art tools rely on multiple cores, transparently transferring control from an unmodified game thread to a modified instance on a different core.

Gary outlined the way to overcome the security problems of online games. His approach is founded on three pillars

that should govern the design of secure software architectures: a risk management framework, software security touchpoints, and knowledge. Software security touchpoints denote the phases and points during the software design process where developers should think how the system may be exploited. For further reading, Gary referred to his book, *Software Security,* published by Addison-Wesley.

In response to a question about why one would use a dual-core system to exploit a game rather than a kernel debugger, Gary said that programming at user level was less complex than in-kernel development. Another questioner wondered whether virtualization could help in preventing hacking of online games. Gary responded that this may be the case in the beginning, but eventually there will be exploits for virtual machine systems. Finally, Gary was asked where the name "warden" came from in Blizzard. Gary said that it was named like this in the EULA itself.

PROGRAMMING ABSTRACTIONS FOR NETWORK SERVICES

Summarized by Peter J. Desnoyers (pjd@cs.umass.edu)

■ *Events Can Make Sense*

Maxwell Krohn, MIT CSAIL; Eddie Kohler, University of California, Los Angeles; M. Frans Kaashoek, MIT CSAIL

Maxwell Krohn presented Tame, a new programming model for event-based applications which attempts to combine the (relative) ease of programming associated with thread-based models of concurrency with the power and responsiveness of events. The primary difficulty in event-based programming has been termed "stack ripping" by Adya et al. (USENIX Annual Tech '02) and was illustrated with an example of a simple function with several blocking operations; putting this in event-based form required splitting it into a separate function per blocking point, turning a five-line function into half a screen of code. Other common operations are difficult to express with threads; however, an example of performing multiple blocking operations in parallel was given (with the same function) and required a full screen of bookkeeping code.

Tame is implemented as a source-to-source translator and a set of C++ libraries, and it provides a simple set of primitives for high-performance network programming that are designed to provide the expressiveness and performance of events, combined with the readability of threaded code. The four primitive types are events (e = event<>;), wait blocks (twait{..code..}), variable closures (tvars{..decls..}), and rendezvous objects (rv = rendezvous<>;). An event can be triggered (e.trigger();), and a twait block will execute all the statements within the block and then wait until all events that were created during block execution have been triggered. tvar is used to declare heap-based local variables that will survive across calls to twait. Finally, rendezvous objects provide finer-grained control over waiting. The example given was maintaining a window of outstanding operations, which could be done in a straightforward fashion using rendezvous.

Results compared Tame with Capriccio, a high-performance thread package, measuring threaded and Tamed versions of a small Web server. Throughput was equivalent, and Tame memory consumption was much lower (3x physical and 5x virtual), owing to its use of a single stack. An informal user study of usability was performed by giving students an assignment where half used Tame and the other half libasync; Tame users averaged 20% fewer lines of source and 50% fewer lines of header files. Tame is in production use in OKWS, an event-based server, and on the okcupid.com commercial Web site.

Tame was described as continuing Adya et al.'s work on threads and events and making practical some ideas from Haskell and Concurrent ML. Other related work included protothreads, Duff's device, and the porch checkpointer. Code is available at www.okws.org, and Eddie Kohler's fork can be found at www.read.cs.ucla.edu/tamer.

Greg Minshall asked about several sources of possible programming errors (functions returning at wait points and long twait blocks). The response was that the first was no worse than the same logic in threads, and the second was bad programming practice. Another question elicited the advice that it is a good idea for caller functions always to create new events to pass to callees. Finally, Oleg Kiselyov pointed out that Tame events were first-class delimited continuations, which was related to some issues preventing exceptions from being used in Tame.

■ *MapJAX: Data Structure Abstractions for Asynchronous Web Applications*

Daniel S. Myers, Jennifer N. Carlisle, James A. Cowling, and Barbara H. Liskov, MIT CSAIL

Daniel started by explaining how AJAX applications work, and just how dreadful the programming environment is for creating a Web application that responds quickly to user input by fetching server-side data. He then presented MapJAX, a language for this purpose implemented as a small set of Javascript extensions, with source-to-source translation producing pure Javascript. MapJAX provides threads, locks, and a simple map object for accessing server-side data. In addition, for performance MapJAX includes parallel for loops and integrated caching and prefetching of map data.

The core language object is a read-only key-value map associated with a server URL, which supplies values to populate the map. Values are retrieved on demand and cached. In addition a user-defined prefetch method can be associated with a map to provide a list of prefetch keys based on request history.

After describing the parallel for primitive, pfor, Daniel pointed out problems with results being returned in arbi-

trary order, and the difficulty of preventing this with locks without eliminating parallelism. Instead, MapJAX provides an RLU (reserve/lock/unlock) lock, where each loop of the pfor can reserve a position in the lock queue and only take the lock after the blocking call has returned. After a brief implementation description, the remainder of the talk focused on two test applications that were implemented both with MapJAX and manually with standard AJAX techniques—a search/suggest application such as Google Suggest and a map viewer modeled on Google Maps.

Network latency and several different bandwidth levels (256 to 4096 kb) were provided by dummynet, and latency results were presented. The request/response application was very latency-sensitive, with small requests, and AJAX and MapJAX performances without prefetching were similar. However, prefetching gave the MapJAX implementation a significant benefit at higher bandwidths. The map application was much more bandwidth-intensive (4.8K per image tile). With prefetching, MapJAX performance was similar to AJAX at low bandwidth and showed a slight improvement at 1 mb/s, but increased to a 20:1 difference in latency at 4 mb/s.

Related work includes a number of libraries for AJAX development, although none of these provides thread or locking features. The RLU lock appears to be novel in this work. TAME was also mentioned as being somewhat related. A number of future enhancements are planned: network speed characterization and adaptive prefetching based on this, persistent caching on the client, and mutable server-side data structures.

Geoff Kuenning (Harvey Mudd) expressed the concern that the intriguing RLU lock added another way for programmers to screw something up (locking) that was already hard enough. What if you never act on the reservation? The authors' reply was that a wait/notify mechanism might be possible. Mike Swift (University of Wisconsin) asked whether they had found any painful limitations in Javascript; they hadn't, although there were weird implementation problems such as nested functions being excessively slow.

■ ***Sprockets: Safe Extensions for Distributed File Systems***

Daniel Peek, Edmund B. Nightingale, and Brett D. Higgins, University of Michigan; Puspesh Kumar, IIT Kharagpur; Jason Flinn, University of Michigan

Daniel Peek presented Sprockets, a system for implementing extensions (such as loadable Apache modules) in a safe fashion, unlike other user-space mechanisms, many of which seem to require a tradeoff between performance without safety (using dynamic loading and direct linkage in the same address space) and safety without performance (forking isolated processes).

Sprockets was developed to support extensions to the EnsemBlue distributed file system (e.g., to support camera protocols and MP3 metadata). These can be implemented by extending a user-space server, but safety was essential to avoid corrupting the entire distributed file system. The result is Sprockets, which uses binary rewriting to safely run extensions within the address space of the invoking application. Sprockets has three goals: safety, upgradability, and speed. Extensions should only change state in the core system via return values, and they cannot cause externally visible effects. The system must be easy to add to, and extensions must be easy to implement. Because extensions in EnsemBlue are often invoked very many times, the per-invocation costs should be minimized.

Existing alternatives were contrasted: direct dynamic loading (fast and totally unsafe), fork and exec (difficult to use because of IPC, slow, and do not eliminate safety issues resulting from system calls), and fork without exec (easier to use, but still slow and with syscall safety issues). Sprockets executes extensions in the invoking process address space, with no OS intervention, using binary instrumentation. It uses the Pin tool from Intel and the University of Colorado (rogue.colorado.edu/pin), which dynamically rewrites code and caches it. This allows extension code to safely call core routines, as the core routines will be rewritten and cached when executed as an extension.

Rewriting is used to validate system calls and their arguments, and to trace memory writes to the core application address space, keeping a log of these writes so that the original contents can be restored. To speed up the logging process, inline checks are added to avoid logging the sprocket stack and other known safe areas. Sprockets guards against the following extension bugs: memory leaks/memory stomping (via log/restore), segfault (via signal handler), infinite loops (via timer and signal handler), and mishandled file descriptors (a sprocket can only read/close files it opened, which are automatically closed when it finishes). To keep threads from seeing changes caused by misbehaving sprockets, all application threads are halted before entering a sprocket.

Evaluation was performed by developing and testing three different sprockets, as well as some nonsprocket versions. To test safety, different bugs were injected into the extensions; procedural extensions caused application failure in all cases, fork with no exec did not catch the bad system call, and sprockets caught all of them. Measured execution time for a single extension call was 500 to 750 ns for the extensions tested, contrasted with 1.5–3 ms for the fork/no exec versions of the same functionality. This was considered adequate performance for a system providing strong safety guarantees.

The first question asked about extensions generating code. Pin handles this and will happily rewrite and instrument the generated code. Since Daniel mentioned that Sprockets was not totally safe against malicious extensions, Andrew Baumann (University of New South Wales) asked just

what they could do; they could try to undermine a mechanism such as the undo log, and this might be preventable with some additional checks. George Forman (HP) asked whether a copy-on-write address space could be used for multiple sprockets. The authors haven't looked at this but have considered transactional memory.

INVITED TALK

Second Life

Rob Lanphier and Mark Lentczner, Linden Lab

Summarized by Xiaoning Ding (dingxn@cse.ohio-state.edu)

The two speakers introduced Second Life (SL) and demonstrated its features using an SL avatar. Then the speakers introduced the architectural changes of the distributed system used by SL. Second Life is a metaverse or a virtual world. The speakers emphasized that SL is not a game, although users may play games in it. SL provides an engine for users to create content in it.

SL is a big system and is still growing rapidly. To illustrate how big the system is, the speakers gave the following numbers. In the system, as many as 100 million SQL queries are made each day, over 45 TB of data have been created by the users, the peak bandwidth reaches 10 Gbps, and 1 PB of traffic is generated each month. The big system supports a tremendous number of activities of a large population. SL has 7.2 million registered accounts, of which about half a million are active residents. More than 60% of residents in SL participate in content creation. This is very different from other platforms. For example, most people read Web pages, but only about 10% of people create Web content. People in SL build items, exchange items, buy and sell items, and have a GDP of 40 million U.S. dollars. SL has its own virtual currency, the Linden dollar (L$), which is exchangeable for U.S. dollars. There are 1.9 billion Linden dollars in circulation. Some people even make real money through the virtual world.

The primitive SL system consists of viewers, simulators, a spaceserver, and a userserver. Viewers comprise client-side software running on users' computers. Simulators are server processes running on machines called sim nodes. Each simulator simulates one geographic region. When an avatar moves to a new region, the sim node may be dynamically changed. In the architecture, spaceserver handles message routings and userserver handles user logins and instant messages. The primitive architecture was improved by adopting a central mysql database and a database server, which performs queries for the simulators. The architecture also introduced an HTTP server and a mail server to increase the functionality of SL. These two servers, together with sim nodes and the userserver, were connected to the central database.

The architecture currently used by SL is similar to the architecture just described. The most important change is to improve the scalability of the database system by clustering and partitioning. Other changes include the use of an individual login server to handle user authentication, etc. In the current architecture, all sim nodes are connected to the databases. When the sim nodes are in remote physical locations from the databases, accessing databases becomes a performance bottleneck. Moreover, a database failure may affect all sim nodes. In the future, the architecture will be changed significantly, so that viewers are connected directly to agent domains and region domains. An agent domain includes the data storage system and computers responsible for handling avatars, including their inventories. A region domain includes the data storage system and computers for simulating the environment and physics of multiple regions. By using the domains, the data and communications are localized and both scalability and availability may be increased.

DISTRIBUTED STORAGE

Summarized by Xiaoning Ding (dingxn@cse.ohio-state.edu)

SafeStore: A Durable and Practical Storage System

Ramakrishna Kotla, Lorenzo Alvisi, and Mike Dahlin, The University of Texas at Austin

Awarded Best Paper!

Ramakrishna presented a distributed storage system called SafeStore, which is designed to maintain long-term data durability practically despite a broad range of threats such as hardware and software faults, operator errors, attacks, and disasters. SafeStore maintains data durability by applying fault isolation aggressively. It spreads data redundantly across multiple autonomous storage service providers (SSPs) to prevent data loss caused by physical, geographical, and administrator faults. It restricts the interface between the data owner and the SSPs to guard data stored at SSPs against faults at the data owner site. Outsourcing data storage to SSPs also reduces hardware and administrative costs by exploiting economies of scale. The challenge of using SSPs is that the users have only limited or no control over SSPs. To achieve high end-to-end durability practically, SafeStore uses the following techniques.

First, it spreads data efficiently across autonomous SSPs with an informed hierarchical coding scheme. SafeStore uses both inter-SSP and intra-SSP redundancy; that is, it first stores data redundantly across different SSPs and then each SSP replicates data internally. The informed hierarchical coding scheme deals with the problem of distributing overall redundancy optimally between inter-SSP and intra-SSP redundancies. Under the scheme it is assumed that each SSP exposes a set of redundancy factors it supports and the data owner can control intra-SSP redundancies by

choosing a factor. The scheme first maximizes inter-SSP redundancy heuristically, by choosing minimal intra-SSP redundancies. Then it distributes the remaining redundancies uniformly across intra-SSP redundancies. The heuristic is based on the intuition that the durability depends mainly on maximizing inter-SSP redundancy and is only slightly affected by intra-SSP redundancies. The informed hierarchical coding scheme can achieve close to optimal durability.

Second, SafeStore performs an efficient end-to-end audit mechanism on SSPs. The audit mechanism quickly detects data losses at SSPs, so that data can be recovered before unrecoverable loss happens. In the audit mechanism, the auditor sends the SSP a list of object IDs and a random challenge. The SSP computes a cryptographic hash on both the challenge and the data and then sends the information back. The auditor randomly spot-checks a portion of the objects by retrieving the corresponding data and verifying the cryptographic hash. SafeStore classifies SSPs into three groups: honest and active SSPs, which verify data integrity proactively and notify data losses; honest and passive SSPs, which rely on audit to check data integrity; and dishonest SSPs, which may lie even if data is lost. Regular audits detect data losses quickly with passive SSPs and spot-checks detect dishonest SSPs with high probability. The audit mechanism is cost-effective because most of the audit work is offloaded to SSPs and there are only occasional spot-checks that need data transfers for a small subset of objects. The audit mechanism provides two 9s or better durability.

The evaluation shows that SafeStore can provide high durability practically and economically, with cost, availability, and performance competitive with traditional systems. More information can be found at http://www.cs.utexas.edu/~kotla/SafeStore.

- ***POTSHARDS: Secure Long-Term Storage Without Encryption***

  Mark W. Storer, Kevin M. Greenan, and Ethan L. Miller, University of California, Santa Cruz; Kaladhar Voruganti, Network Appliance

Mark W. Storer presented a secure, long-term storage system called POTSHARDS. POTSHARDS provides security by secret splitting, instead of encryption, which is unsuitable for long-term storage because of the difficulties of key management and updating cryptosystems over long time periods.

To store data, POTSHARDS breaks the data into *n* pieces, called shards, in a way that *m* out of *n* shards must be obtained to recover the data and any set of fewer than *m* shards contains no information about the data. Then POTSHARDS returns the shard IDs to the user and spreads the shards across multiple independently managed archives. The user maintains a private index that maps the data to shards for fast retrievals. Keeping the index private reduces the threat of inside attackers. To retrieve data, the user provides shard IDs and authentication information. POTSHARDS requests *m* shards from different archives and rebuilds the data from the shards. Thus the security is enforced by hiding shard locations and performing authentication on multiple independent archives.

POTSHARDS uses approximate pointers to recover data even if all indices of a user's data have been lost. Approximate pointers point to multiple "candidate" shards that might be the next that can be used to reconstruct the data. Approximate pointers provide sufficient clues for users to recover their data by trying only candidates they own. Meanwhile, the approximate pointers greatly increase the workload of intruders, because the intruders have to try all the candidates. POTSHARDS also uses a secure distributed RAID technique to provide availability and data recovery.

The performance evaluation on a prototype implementation shows that POTSHARDS can write data at 3 MB/s and can read data at 5 MB/s, which are acceptable rates for archiving. The throughputs were tested with 12 clients. With more clients, POTSHARDS can achieve higher throughputs.

- ***Dandelion: Cooperative Content Distribution with Robust Incentives***

  Michael Sirivianos, Jong Han Park, Xiaowei Yang, and Stanislaw Jarecki, University of California, Irvine

Michael Sirivianos presented the work of Dandelion, a system designed to provide robust incentives for cooperation in commercial P2P content distribution with sufficient scalability. The speaker emphasized that Dandelion is not a distributed storage system.

Although the popular BitTorrent protocol has incorporated tit-for-tat incentives, it does not encourage seeding and still allows modified clients to free-ride. With robust incentives, Dandelion increases the network's aggregate upload bandwidth by motivating clients to upload even when they are not interested in the network's content and by preventing free-riding.

Dandelion builds a virtual credit system, in which a client honestly uploading to its peer is rewarded by virtual credit and a client obtaining the correct content is charged the same amount of credit. In the system, credit can be redeemed for rewards such as discounts or monetary awards; thus, clients are always encouraged to upload. The system prevents free-riding because the only way a client can obtain valid content and can earn credit is by paying credit or uploading valid content, respectively. To prevent client cheating, Dandelion uses a cryptographic fair exchange scheme based on symmetric key cryptography, in which the server acts as the trusted third party mediating the content exchanges. Thus, Dandelion trades scalability for robust incentives. To prevent Sybil attacks, Dandelion maintains only authenticated paid accounts.

Michael Sirivianos and his team implemented a prototype of Dandelion and evaluated it on PlanetLab. With a server running on normal commodity hardware (dual Pentium D 2.8-GHz CPU and 1 GB RAM) with a moderate network bandwidth (1 to 5 Mbps), Dandelion can support about 3000 clients downloading 256KB chunks with a rate of 256KB/s. When the network bandwidth is low (less than 4 Mbps), the bandwidth is the bottleneck. When the bandwidth is high (larger than 5 Mbps), CPU is the bottleneck. The evaluation also shows that robust incentives for seeding substantially improve downloading times, especially for small files, and the performance of Dandelion clients is comparable to those in BitTorrent.

One member of the audience asked what the differences between Dandelion's and eMule's incentive schemes are. The presenter responded that eMule peers maintain pairwise credit balances and give high service priority to peers with high credit. This in essence is a tit-for-tat scheme and has the same weaknesses as BitTorrent's TFT, i.e., it provides no incentives for seeding, and it is susceptible to free-riding.

INVITED TALK

Specializing General-Purpose Computing: A New Approach to Designing Clusters for High-Performance Technical Computing

Win Treese, SiCortex Inc.

Summarized by Jan Stoess (stoess@ira.uka.de)

Win Treese presented SiCortex's new approach to designing clusters for high-performance computing. Supercomputers are often based on specialized hardware and programming environments, which are expensive and obstructive to rapid innovation. General-purpose computing, in turn, uses standard software and commodity PCs and shows an amazing technology curve. However, commodity PCs are optimized for desktop and server systems rather than for the specific demands of technical computing. The challenge is therefore to unite the best of both worlds: to build a supercomputer that uses general-purpose hardware and a standard programming environment but that is also specifically designed for technical supercomputing.

Win presented a brief sketch of the history of supercomputing: supercomputers such as Cray, CM1, or BlueGene are all expensive machines, each with a different programming environment. They place high demands on the skills of their users and maintainers. Also, supercomputing companies tend to have a hard time generating any income. An alternative and cheaper model is therefore to take lots of cheap PCs and cluster them using commodity interconnect technology such as Ethernet—as was done in the Beowulf project. Even with optimized interconnects such as Myrinet or Infiniband, PC clusters are still cheaper than their big, specialized brothers.

Typical workloads of supercomputers are climate simulations, mechanical design problems, or life science simulations. The applications usually run for weeks and consume every available cycle. They tend to operate on huge data sets in a cache-unfriendly manner, and they place high demands on the communication infrastructure. Programs are usually written in Fortran, a language that permits many optimizations to be made during compilation. More recently, Java and Python have become popular as well, mostly because of the benefits in programming productivity.

By now, HPTC computing on PC clusters has become a mainstream phenomenon. Sales of Linux cluster hardware have reached $6 billion in 2006. The advantages of PC clusters are their cheap prices for hardware, software, and interconnects; their support for emerging software standards such as Linux, MPI, or Fortran; finally, their amazing technology curve.

However, many challenges remain for PC-based supercomputing. Because PCs were originally designed for personal computing, PC clusters often show little computational efficiency, have high power consumption, and generate a lot of heat. Also, the parts are likely to fail and the interconnect is slow. However, software development and standards play a significant role in the overall costs of a supercomputer. A potential way out is, therefore, the replacement of commodity hardware with a specialized system that still retains support for the standard cluster programming environment.

Win then explained how SiCortex has built such a system. They aimed to realize a 1,000-node cluster with near-microsecond communication delay in a cabinet-sized box. Their main design principle was "logic of power." Lower power consumption results in less heat, which, in turn, allows components to be located closer together and interconnect wires to be shorter. Reduced heat also increases the reliability, and reduced power consumption saves energy during memory stalls. Win introduced two supercomputers presently manufactured by SiCortex. The big model, the SC5832, has a floating-point capacity of 5832 gigaflops, 972 6-core nodes at 500 MHz each, 7.8 GB of memory, and roughly 2,900 interconnects. It consumes about 18 KW and fits into a 5x5x6–foot cabinet. The smaller model, with 648 gigaflops, 108 nodes, and 2 KW power consumption, fits into a half-width standard 19-inch rack. The software platform consists of a standard, open-source Linux environment with support for GCC, Fortran, MPI, and even Emacs. SiCortex also provides an integrated Lustre file system that can be used for direct-connect storage, for external storage servers, or even for RAM-based file systems.

Win concluded with the observation that general computing techniques and specialized knowledge on supercomputing applications can be mixed very well to support

powerful and usable high-performance technical supercomputing.

Someone asked about the price of the systems, and Win gave ballpark numbers of $200,000 for the small box and $1.5 to $2 million for the big one. Another question was how nodes could be replaced on a failure. Win stated that the hardware was hot-swappable, but the system still lacked support for software hot-swapping. He noted, however, that software hot-swapping would be straightforward to implement. Asked how software could be ported to the supercomputers, Win responded that single-CPU programs would need redevelopment to exploit parallelism; other programs can simply be recompiled for MIPS targets. Win was asked how the system deals with multicast or broadcast messages. He explained that the DMA engines provide hardware commands to build up broadcast trees efficiently. Finally, a questioner inquired how jobs are scheduled in their system. Win said the system ships with a job scheduler developed by Lawrence Livermore National Laboratory.

DATA AND INDEXING

Summarized by Peter J. Desnoyers (pjd@cs.umass.edu)

■ *Using Provenance to Aid in Personal File Search*

Sam Shah, University of Michigan; Craig A.N. Soules, HP Labs; Gregory R. Ganger, Carnegie Mellon University; Brian D. Noble, University of Michigan

Sam Shah presented a system that tracks causal relationships between files in a system to improve desktop search results. In the first part of the talk he described the approach taken in this work, comparing it with previous work; the rest of the talk focused on results from a rigorous usability study with test subjects. Google Desktop and others use static indexing based solely on file contents; Soules and Ganger [SOSP '05], however, have shown that dynamic information (patterns of use) can be used to reorder search results and give user-perceived improvement in search performance. In particular, provenance information—indications of which files were referenced to create another file—can be used to infer relations between files.

The current state of the art for this uses temporal locality—that is, if read(A) is followed by write(B) within a T-second time window, then A and B are related. This captures reading an email and then copying it into a text document; however, it fails in a number of cases. For example, one test user kept a CAD application open all day; using temporal locality inferred relationships between the CAD files and all other user activities.

The authors instead use causal relationships—A and B are associated if read(A) and write(B) occur in the same process, or in two processes with an IPC path during the intervening interval. Each approach handles some cases better than others. These relationships are used to create a DAG (Directed Acyclic Graph) with edges weighted by the number of relationship events seen, and then a breadth-first traversal method is used to redistribute relevance weights returned by static search, resulting in a new weighting used to order results for the user. Measurement of both causal and temporal relationships was implemented on win32, using binary rewriting to interpose between applications and the file system. The search application was based on Google Desktop, using the GDI API with provenance-based reordering of results.

Shah discussed why a typical corpus-based approach to measuring precision and recall, as used in many information retrieval studies, was not appropriate to this application. Instead, a real user study over a period of time was needed. A full randomized controlled trial would be best but would require too many (300) subjects. Instead, four techniques (content-only, temporal locality, causality, and randomizing the results from content-only) were compared against each other using a repeated measures experiment. Twenty-seven non-CS undergraduate test subjects used the search system for a month, and all queries were recorded. In a single test session afterward, seven successful queries were chosen (i.e., the user selected at least one result). For each query the results of the four search algorithms were presented side by side, and the user rated each on a scale of 1–5.

Causal relationship–enhanced search was found to be rated somewhat better (17%) than temporal-enhanced search, temporal was statistically indistinguishable from content-only search, and randomized results were 36% worse. Shah speculated that the reason for the superiority of the causal approach was its improved handling of noise—it added fewer false relationships. Finally, some performance results of the tracing overhead and search overhead were given; they were not sizable but could use some tuning.

Sam closed by expressing the hope that this work would inspire the OS community to consider user studies in their own work. He stated that it really wasn't that painful and that there were many important insights to be gained.

The first question dealt with trace overhead; more efficient methods evidently exist and could be used, with some effort. Another questioner asked about the similarity to Web search and whether this algorithm would apply there or page rank would apply here. Sam pointed out that page rank is based on links that are deliberately created, whereas this system infers links automatically. Terrence Kelly asked why there aren't many proper controlled user studies; the perception is that it's hard and that the IRB is a barrier. Sam stated that it isn't very painful, but that if you can get work published with poor studies, there's little incentive for good ones. Mike Abbot asked what happens if you combine causality and temporal relationship; this is a topic for future work.

■ ***Supporting Practical Content-Addressable Caching with CZIP Compression***

KyoungSoo Park and Sunghwan Ihm, Princeton University; Mic Bowman, Intel Research; Vivek S. Pai, Princeton University

KyoungSoo Park presented CZIP, which uses a compression scheme based on content-based naming, where blocks of data are referred to by hashes of their contents. This allows for redundancy elimination, as the same block of data may be incorporated by reference in multiple locations or multiple files. It is used in a number of systems, including LBFS, VBWC, Venti, and others. CZIP defines a format for encoding and decoding files using content-based naming, as well as components to easily add this functionality to existing or new applications.

A number of uses for content-based naming were described: file distribution (e.g., a Linux release, where the contents of the CD ISOs are duplicated in the DVD ISO); virtual machine (VM) images (e.g., migration), where the base OS is identical across machines, and uncacheable Web content, which typically changes slowly and in portions.

CZIP splits a file into chunks, using either a fixed chunk size or Rabin's fingerprint, hashing the chunks, and then representing the file by the sequence of chunk hashes plus the chunks (possibly compressed) themselves. In network applications CZIP can be deployed on the server side, the client side, or both. On the server side, CZIP can either be used to compress files or for in-memory caching only; on the client (or proxy) side a range request can be used to read the header of a CZIP-encoded file, and then only those chunks not already cached need be retrieved.

CZIP compressed the Fedora Core 6 release by a factor of 2; because of duplication between CD and DVD ISOs, bzip2 achieved no compression. On data without duplication (the Wikipedia DB) CZIP gave no compression, whereas bzip2 gave 4x compression. Apache and MySQL server VM images showed high overlap (compressibility) with most parameters, and five VMs used as engineering desktops for three weeks showed very high (96%–99%) overlap. Finally, samples taken every 30 minutes of a number of dynamic Web pages (Google News, CNN, others) showed data savings of 37% to 90% with CZIP. A CZIP-aware Apache server serving the Fedora Core 6 distribution to 100 clients achieved a memory savings of a factor of 2, resulting in higher throughput as the compressed working set fit within physical RAM. A content distribution network (CDN) based on Codeen was shown to decrease origin server bandwidth used to one-quarter that of the non-CZIP version.

A CZIP release at http://codeen.cs.princeton.edu/czip should be available soon.

Terrence Kelly asked why the authors claim that this technique can avoid a round-trip delay compared to duplicate transfer detection (DTD); evidently DTD needs another RTT to check the checksum, whereas CZIP can avoid the delay if it expects and gets redundancy. Jason Flinn asked how useful CZIP is for an individual client, and whether there were any numbers for its effect on client latency. Although the server gets more benefit, clients could still benefit, and experiments to measure client latency haven't been performed yet.

■ ***Short Paper: Implementation and Performance Evaluation of Fuzzy File Block Matching***

Bo Han and Pete Keleher, University of Maryland, College Park

Bo Han presented an evaluation of fuzzy file block matching, an extension to content-addressable storage for efficiently representing small updates to files. Instead of identifying each block by a single hash, and only substituting exact matches, a hash vector that can be used to measure block similarity is produced for each block; error-correcting codes may then be used to "correct" a similar block to create the desired block. This extends work by Tolia et al. (USENIX '03) by providing a replicatable implementation and performance evaluation.

Files are chunked via Rabin's fingerprint, and then each block is described by a "shingleprint." These are computed by taking a sliding m-byte window within the block, starting at each byte offset and hashing that window, and then subsampling the resulting set of hashes by taking the S hashes with the smallest numerical values. The fuzzy matching algorithm declares two blocks to be similar if their prints share T out of S values. An ECC code for each block is then generated by using a 255/223 Reed-Solomon code on small parts of the block. Finally, a possible architecture for using fuzzy matching to maintain and update a cache was described.

Experiments were performed using the GNU Emacs source tree versions 20.7 through 21.4, counting how many files in version N+1 could be recovered from version N plus ECC information for version N+1. These experiments were performed for a number of different parameters, including the sliding window size and the ECC organization.

Related work includes Venti, which uses hashes as block IDs, REBL, which uses super-fingerprints, TAPER, which uses Bloom filters, and LBFS. Bo concluded by saying that the main advantage of fuzzy matching is the possible savings of network bandwidth and that more experiments with expanded data sets, ECC codes, and parameter combinations were needed, as well as integration of fuzzy matching into an existing distributed file system.

Terrence Kelly asked whether any evaluation of the effectiveness of this proposal in the wide area would be difficult because of the need for traces of both requests and replies. Bo replied that they had only looked at the local version; future work might examine this, and in that case it might well be an issue. To a question about hash collisions and security Bo replied that these may need consideration in the future as well.

INVITED TALK

■ *Live Malware Attack!*

Paul Ducklin, Sophos

Summarized by Jan Stoess (stoess@ira.uka.de)

Paul Ducklin presented a demonstration of how typical malware exploits a computer system, and how a malware analyst can find out the nature and behavior of such malware. Since malware usually relies on Internet access, an analyst must provide a "simulated Internet" environment that causes the exploit to install and activate itself but also allows monitoring and inspection of how the exploit proceeds.

Paul presented such a "deductive and analytical" inspection environment, which consisted of two virtual computers running within the QEMU simulator on his host laptop. The first virtual computer contained a Windows installation acting as the client, and the second hosted a Linux instance establishing the "simulated Internet." The simulation consisted of a set of tools providing standard services such as DNS, HTTP, and Mail in a way that client requests could be monitored and transparently rerouted to services or files in the control of the malware analyst.

Paul then installed a sample malware program in the Windows client. The program exploits a bug in the library code processing Windows Meta Files (WMF). The malware attack occurs via a WMF file embedded in a Web page. Since the library routine does not check for overflows during retrieval, it also copies the malware program on the stack; it then overwrites the stack return address to activate the malware code.

Paul then proceeded to inspect the code within the file. It turned out to be scrambled, and no additional knowledge could be inferred without installing it into memory. Paul therefore downloaded the malware again, but this time he ran the browser within a debugger. In addition, he modified the malware's source code to trap into the debugger after activation. Paul could then single-step the code, to observe that the next steps would be to download a Trojan horse rootkit via a SOCKS proxy. Paul executed that rootkit together with a file system call monitor; he could thereby determine whether the rootkit had installed itself into several new files but had also hidden itself from inspection in the file system.

Paul ended his enjoyable presentation by demonstrating how the rootkit actually stashed itself. For that purpose, Paul attached WinDbg, a local kernel debugger, to the client Windows installation. The malicious driver had changed the Windows system call table to transparently rewrite file system access calls in such a way that they would ignore the newly generated files containing the rootkit code.

SYSTEM SECURITY

Summarized by Francis David (fdavid@uiuc.edu)

■ *From Trusted to Secure: Building and Executing Applications That Enforce System Security*

Boniface Hicks, Sandra Rueda, Trent Jaeger, and Patrick McDaniel, Pennsylvania State University

Boniface Hicks started by pointing out that mandatory access control (MAC) works well for naturally separated processes. Several operating systems use MAC to restrict the flow of information between processes based on security levels. Unfortunately, there are several applications that defy classification at a security level and are marked as "trusted" and are allowed complete access to the system. An example of such an application is a log rotation server that needs to rotate logs for multiple applications at different security levels. The goal of this research is to allow MAC policies to be enforced within applications as well, in order to prevent their circumvention by compromising a trusted application.

To enforce system security policies within applications, Boniface proposes the use of security-typed languages. An operating system service called SIESTA has been designed and built that can enforce SELinux security labels at compile time within applications written in the Jif security-typed language. The security labels used by the operating system are tracked within the application and the language ensures that information flows are in compliance with the MAC policies. Declassification or downgrading the security level of an information flow is also allowed if it is specified in the policy.

A secure logrotate service and a secure email client were written to demonstrate this concept. Experiments show that the performance of these secure applications suffers some deterioration when compared to native applications. This overhead is close to 100% for logrotate. Boniface mentioned that this may be because the Jif code was not optimized. In his response to a question, he noted that the system does not avoid covert channels.

■ *From STEM to SEAD: Speculative Execution for Automated Defense*

Michael E. Locasto, Angelos Stavrou, Gabriela F. Cretu, and Angelos D. Keromytis, Columbia University

Michael Locasto said that self-healing systems are needed because typical computer defense systems crash the process that is being protected during an attack. Self-healing systems provide a less drastic approach to recovering from an attack. This work extends and addresses some shortcomings of their previous system for self-healing, which was called STEM. The new system provides self-healing capabilities for unmodified binaries without the need for source code. In STEM, every function is treated as a transaction that is speculatively executed. Error codes are

returned when a problem is detected. The SEAD system improves on this design by utilizing binary rewriting to avoid source code modification and recompilation. It also supports repair policies to customize an application's response to an attack. Virtual proxies are used to ameliorate the output commit problem, and process behavior profiles can be created on aspects of data and control flow.

Repair policies can be used to perform better error virtualization and avoid returning incorrect return values. They also support memory rollback for aborted transactions and forced modifications to memory. Another advantage of using repair policies instead of generating and deploying new code is that a policy can easily be switched off if it turns out to be incorrect.

Performance evaluations show that although the system does not have a human-discernable impact when applied to regular applications, there is a rather significant impact on an application's startup time. As part of future work, Michael is looking at ways to automatically generate repair policies.

Several members of the audience pointed out that software vulnerabilities will still exist after the system self-heals and suggested that it might be better just to fix the code than to fix bad code with policies. Yet another limitation that was pointed out was that it is not easy to decide upon the meaning of return values when there is no access to source code.

- ***Dynamic Spyware Analysis***

  Manuel Egele, Christopher Kruegel, and Engin Kirda, Secure Systems Lab, Technical University Vienna; Heng Yin, Carnegie Mellon University and College of William and Mary; Dawn Song, Carnegie Mellon University

Manuel Egele began his presentation by noting that spyware is a growing threat to Internet users. Browser Helper Objects (BHOs) are widely used to implement spyware and are the focus of this work. Detection of such spyware using signature-based techniques misses newer spyware, and behavior-based techniques are required to detect them. The system described in the presentation tracks the flow of sensitive data through the system and observes the behavior of BHOs. For example, URLs that are typed in are considered sensitive information; the system checks for BHOs that misuse this information. The system provides comprehensive reports about file, network, IPC, and OS actions.

QEMU was modified to enable the tracking of data and control dependencies. The control dependencies tracking approach makes use of a control flow graph generated from disassembled instructions. A browser session of a user is prerecorded and this captured session is replayed for analysis of spyware behavior. Experiments show that the system is effective at detecting and presenting an analysis of several spyware samples. It was also able to detect a spyware sample that was not detected by several commercial products. Manuel noted that running the analysis using the QEMU emulator caused a factor-of-10 slowdown in execution.

INVITED TALK

- ***LiveJournal's Backend Technologies***

  Brad Fitzpatrick, LiveJournal

  Summarized by Peter J. Desnoyers (pjd@cs.umass.edu)

Brad Fitzpatrick described the process of evolving the LiveJournal implementation from the college hobby project for college and high school friends in 1999 to the system it is today, with over 10 million accounts. Slides for this talk (or at least equivalent talks) are available at http://www.danga.com/words.

LiveJournal combines blogging, forums, social networking, and aggregation. In its current form it includes the following open-source components, all of which evolved as the system scaled: memcached, mogilefs, perlbal, gearman, theschwarz, djabberd, and openid.

Brad started by discussing scaling, and in particular he said that the absolute speed of a solution isn't as important as whether it scales linearly—does it run 2x or 10x as fast when you use 2x or 10x as many servers? If not, you'll reach a point where you have to restructure the system to grow, which they did several times.

LiveJournal started out on a single machine with Apache and MySQL, and it worked fine until it got slow. The Apache and MySQL servers were split onto two machines, which created two points of failure and soon became CPU-bound on the Apache machine. This was replaced by three servers with load balancing—a number of solutions were used, but none was completely satisfactory.

Then it became I/O-bound on the MySQL server, which was split in two with MySQL replication. This scaled to twelve mod_perl (application) machines and a MySQL master with six slaves. Then database writes became the bottleneck: Each read was handled on one MySQL machine, but writes were broadcast to the cluster, so all the MySQL machines were spending all their time writing.

This problem was solved by partitioning the database. Users map to one of a number of high-availability MySQL clusters, with a single global master and slave for nonuser metadata. Each cluster is a high-availability pair using the DRBD network block driver for shared storage, plus slaves to add more read capacity.

In the rest of the talk Brad described the components they had developed.

Memcached is a single cache for everything, running wherever there is spare memory. It presents a simple dictionary data type with a network API, and the server for a particular data item is determined by hashing. The main

problem is that you can't add or remove servers at runtime without rehashing everything, but this has been solved recently by using consistent hashing.

Perlbal is a "fast, smart, manageable HTTP Web server/reverse proxy/load balancer." It has two key features: internal redirects, allowing a backend to hand-work (e.g., serving a large file) back to perlbal without the user seeing a redirect, and verification of backend connections.

Mogilefs is a simple read-mostly replicated file system, which, like memcached, can be deployed wherever there are resources.

Gearman is a distributed job/function call system. Workers register functions they implement, and callers invoke with function and arguments encoded as simple strings. Among the uses mentioned were database connection pooling and keeping large libraries (e.g., ImageMagick) out of the mod_perl processes that needed to invoke them.

Theschwartz is sort of like gearman, but instead of being lightweight and unreliable (i.e., the caller has to wait and retry if necessary), theschwartz is reliable and can be used to schedule something and forget it, such as sending email after updating a page.

Djabberd is needed because the other XMPP servers weren't flexible enough to integrate closely with the rest of LiveJournal (e.g., automatically use the existing LiveJournal user picture as an avatar). It's lightweight and just about every function can be hooked.

Ted Ts'o asked several operations questions: Brad guesses they have 150–200 machines, they still run memcached anywhere there's spare memory, and they put disks into netbooting Web nodes for mogilefs, and both practices drive operations batty (a recurring theme). They only have a single data center, but they are expanding into another in Oakland. In response to a question from Simson Garfinkel about their multiday outage in 2005, Brad pointed out that geographic redundancy is much harder and more expensive than redundancy within the same facility.

Although the operations people are scared of it, they use virtualization (Xen), especially to isolate applications that might "explode" and use too much memory (ImageMagick again). Configuration was originally generated from a single YAML file; it is being migrated to cfengine. Failures were discussed; a surprising number were due to other customers at the same facility pressing the Big Red Button. This brought up the issue that it's hard to get a storage stack to sync properly; for example, often even with battery-backed RAID cards the disks are running in write-back mode, and data the DB thinks is committed will be lost if power goes out.

CLOSE TO THE HARDWARE

Summarized by Andrew Baumann (andrewb@cse.unsw.edu.au)

■ *Evaluating Block-level Optimization Through the IO Path*

Alma Riska, Seagate Research; James Larkby-Lahet, University of Pittsburgh; Erik Riedel, Seagate Research

Optimizations in the disk IO path are traditionally thought to be more effective at high layers (such as the file system) where more semantic information about requests is available. However, with advances in modern disk hardware allowing implementation of complex optimizations such as request reordering, this work presented by Alma Riska evaluates the impact of performing optimizations at the disk level. The approach used is based on measurements of the Postmark benchmark and kernel compiles on Linux, with variations in the file system, IO scheduler, and disk drive (with each of the drives used supporting request reordering).

Results show that higher in the IO path, at the file system and IO scheduler, the focus should be on optimizing the amount of disk traffic. The best-performing file systems and IO scheduler algorithms were those that achieved better rates in IO workloads. Lower in the IO path, the focus is on reordering and coalescing requests, which is most efficiently performed at the disk level.

One particularly interesting result was that under write-back caching, which is generally assumed to perform better than write-through caching, increasing the queue depth yields inconsistent performance. Under high load the disk queue fills up, causing more requests to block at the device driver, which is unaware of the disk's queue. Write-through caching with queuing and reordering performs as well as write-back without queuing, and it doesn't suffer from reduced reliability in the case of power failures.

One audience member observed that an advantage of write-back caches was the ability to merge repeated writes; however, Alma noted that previous work at the 2006 conference had shown that repeated writes were extremely rare. She also acknowledged that the performance of write-through and write-back caching was only equivalent in regard to throughput and that the response time for write-through caches would be higher.

■ *DiskSeen: Exploiting Disk Layout and Access History to Enhance I/O Prefetch*

Xiaoning Ding, Ohio State University; Song Jiang, Wayne State University; Feng Chen, Ohio State University; Kei Davis, Los Alamos National Laboratory; Xiaodong Zhang, Ohio State University

Xioning Ding presented this work on DiskSeen, a prefetching scheme that uses knowledge of the disk layout and access history information to improve on the performance of

traditional file-level prefetching, which can incur nonsequential accesses and thus result in excessive seeking.

DiskSeen operates below the file-system level, maintaining a disk block table storing the access history of logical block numbers, which it uses to detect sequences for prefetching. Two types of prefetching are supported: sequence-based prefetching, which occurs when eight or more contiguous blocks are accessed, and history-aware prefetching, which uses the disk block table to detect and prefetch historic access trails that may not be contiguous.

A performance evaluation of DiskSeen in Linux shows that it reduces the time taken for repeated runs of prefetching-friendly applications by 20%–50%. The time for the first run of each application is also improved, but not as much, owing to the lack of history trails in the disk block table. The performance improvement is greatest for applications, such as CVS, that access multiple areas of the disk and thus have the greatest seek overhead. One TPC benchmark degraded by 10% with DiskSeen; however, the authors plan to fix this problem with dynamic adjustment of the timestamp threshold.

■ ***Short Paper: A Memory Soft Error Measurement on Production Systems***

Xin Li, Kai Shen, and Michael C. Huang, University of Rochester; Lingkun Chu, Ask.com

This study, presented by Xin Li, looked at the frequency of soft errors, which are transient hardware errors in memory caused by environmental factors. Previous studies have suggested error rates in the range of 200–5000 failures in time (FIT) per megabit; however, there are no results for modern hardware in a production environment.

This study used several data sources: a server farm at Ask.com, a number of desktops at the University of Rochester, and nodes on PlanetLab. Results for the Ask.com servers were collected from ECC memory error statistics; results from the other machines were collected by a user-level application that attempted to detect memory errors by writing a pattern in memory and checking it for errors. In total over 300 machines were used; however, the 70 PlanetLab nodes were only able to allocate 1.5 MB of free memory in which to look for errors.

The only transient errors found were two in the Ask.com server farm. This corresponds to an error rate orders of magnitude lower than previously reported. The study also unintentionally found some permanent errors (nine errors in the 212 Ask.com machines). The authors conclude that the actual transient error rate is much lower than previously reported, and they speculate that this is due to changes in hardware layout and a reduction in chip size. Future work includes further data collection, identifying error modes that can escape hardware protection, and studying how real software systems will be affected by such errors.

INVITED TALK

■ ***MapReduce and Other Building Blocks for Large-Scale Distributed Systems at Google***

Jeffrey Dean, Google

Summarized by Jan Stoess (stoess@ira.uja.de)

Jeffrey Dean gave a presentation on how Google manages access to the billions of documents available on the Web and via other Google-provided services such as email, personal files, its closed database system, broadcast media, and print services.

Jeffrey explained that his company faces an ever-increasing computational need, which is driven mainly by three factors: (1) more queries, from more people using Google and using them more frequently; (2) more data from the growing Web community and from new products; (3) the need for better search results (finding the right information and finding it faster). Jeffrey then presented three software projects Google has developed to address the increasing computational requirements: GFS, MapReduce, and Bigtable. Following the hardware design philosophy of Google, the software systems run on a very large number of commodity machines, which offer a good price/performance ratio.

The large number of nodes and the huge amount of data lead to unique requirements for the file system. Google has therefore developed its own distributed file system, named GFS. Since Google has control over all applications, libraries, and the operating system running on its machines, it can optimize access to the file system on all levels. A GFS setup consists of a master server holding the metadata and multiple chunk servers holding the actual data. Each file is broken into chunks of 64 MB, each of which is stored redundantly on multiple chunk servers. The chunk servers use a standard Linux file system. Currently, Google stores more than five petabytes of data using GFS.

For processing the data, Google has developed a special programming environment called MapReduce. A MapReduce program consists of a map function and a reduce function. The map function extracts relevant information from each input record and stores it as key/value pairs. The reduce function then aggregates the intermediate values, for instance by means of a hash function. As a simple example, Jeffrey explained how the word frequency of an input text can be determined with MapReduce. In this case the map function splits the input at spaces and has the value "1" for each word. The reduce function sums all values that have the same key. The MapReduce environment is implemented as a library that deals with most of the technical aspects such as the highly parallel and distributed execution of the algorithm, or the reading and writing of data. It thus drastically reduces the programming effort and lets MapReduce users focus on their real goals. The MapReduce implementation is optimized to be fast,

robust, and scalable; it has been used for many Google applications such as Google Ads, Froogle, Google Earth, and Google News. As of June 2006, Google had developed more than 6000 MapReduce programs.

Finally, commercial database management systems do not perform well with the amount of data Google has to manage. Commercial systems also have high licensing costs. Furthermore, the special nature of the stored data and the way of processing it allow for several simplifications and optimizations compared to full-featured database management systems. Google has therefore implemented its own database management system, called Bigtable. The basic data model of Bigtable is a distributed multidimensional sparse map. Each cell is identified by a (row, column, timestamp)-triple. A Bigtable for Web pages, for example, can hold different properties (column) of a Web page located at a specific URL (row), using multiple versions (timestamp). Bigtable is used for several Google products, such as its Web search and Google Earth.

For more detailed information, Jeffrey referred to http://labs.google.com/papers.html, a page featuring one paper for each of the three presented technologies.

NETWORKED SYSTEMS

Summarized by Xiaoning Ding (dingxn@cse.ohio-state.edu)

- ***Addressing Email Loss with SureMail: Measurement, Design, and Evaluation***

  Sharad Agarwal and Venkata N. Padmanabhan, Microsoft Research; Dilip A. Joseph, University of California, Berkeley

The paper addresses the silent email loss problem. In the paper, a silent email loss is defined as the case in which an email is never received by the intended email recipient but neither the sender nor the intended recipient is notified of the loss. Sharad Agarwal first presented the results of their measurements on email losses, and then presented their design of SureMail. SureMail augments the existing SMTP-based email system by notifying the intended email recipients about email losses.

Sharad Agarwal and his team performed an experiment to understand how often legitimate user emails were lost. In the experiment, many email accounts were used to send and to receive emails over several months. Then the sent emails and the received emails were matched to check email losses. The measurements show a silent email loss rate ranging from 0.71% to 1.02% and a total email loss rate ranging from 1.82% to 3.36%. They also compared the email loss rates for two normal msn.com accounts and two msn.com accounts with content filters disabled and found that the accounts with disabled content filters have much lower loss rates. This indicates that the majority of losses were from content filters.

Because the measurements show that the existing SMTP-based email system works most of the time, SureMail is designed, not to replace the existing system, but to augment it with a separate notification mechanism. A notification is a short, fixed-format fingerprint of an email. When an email is sent, the notification is also delivered via an in-band channel or an out-of-band channel. The in-band channel uses email headers, and the out-of-band channel uses separate services such as DHT, Amazon S3, or dedicated notification servers. To prevent spoofing by spammers, SureMail uses a reply-based shared-secret scheme. The scheme sets up a shared secret based on an email and its reply between two correspondents, and it uses the shared secret to authenticate the notifications.

The evaluation on the out-of-band channel shows that 99.9976% of notifications can be delivered successfully at a very low incremental cost.

- ***Wresting Control from BGP: Scalable Fine-Grained Route Control***

  Patrick Verkaik, University of California, San Diego; Dan Pei, Tom Scholl, and Aman Shaikh, AT&T Labs—Research; Alex C. Snoeren, University of California, San Diego; Jacobus E. van der Merwe, AT&T Labs—Research

Patrick Verkaik presented the design and implementation of IRSCP (Intelligent Route Service Control Point), which is an architecture enabling flexible route control for inter-domain traffic without changing existing ISP infrastructure. IRSCP is the follow-up work on RCP.

In the IRSCP architecture, a route control application uses external information, such as network condition, to guide the route selection process in IRSCP. IRSCP communicates the selected routes to the routers in the ISP network and in the neighboring ISP network. The route control application works at relatively slow rate. To enable IRSCP to failover instantly in case a route for egress link fails, the route control application provides IRSCP with an explicit ranking of egress links for each ISP router. To prevent forwarding anomalies caused by inconsistent rankings, such as deflection and looping, IRSCP enforces two simple consistency constraints on the rankings. In a large ISP, IRSCP communicates with many thousands of routers, and it is responsible for route decision for each ISP router. Therefore, IRSCP must be robust and scalable. The paper addresses the problems by partitioning and distributing the IRSCP functionality across multiple IRSCP servers.

Patrick Verkaik and his team evaluated IRSCP by connecting IRSCP to an emulated ISP. The results show that IRSCP is capable of managing the routing load of a large ISP. In response to a question about whether converging could be a problem, Patrick Verkaik replied that IRSCP converges quickly.

■ ***A Comparison of Structured and Unstructured P2P Approaches to Heterogeneous Random Peer Selection***

Vivek Vishnumurthy and Paul Francis, Cornell University

Vivek Vishnumurthy talked about the heterogeneous peer selection problem in structured and unstructured P2P networks. Heterogeneous peer selection means that peers with higher capacities should be selected proportionately more often. To support heterogeneous peer selection, Vivek and Paul implemented Swaplinks (published at Infocom 2006) for unstructured P2P networks and adapted Bamboo DHT, using their extensions to the Karger/Ruhl load-balancing algorithm for structured P2P networks. Then they compared the performance of Swaplinks and the adapted Bamboo DHT (called KRB) in making heterogeneous selections based on capacities.

The basic idea of Swaplinks is to build a random graph, in which each node tries to keep its degree proportional to its specified capacity. Thus the unbiased fixed-length random walks on the random graph result in the desired selection probability. The basic idea of KRB is to adjust peers' ID spaces dynamically based on their loads and their capacities, so that all the peers have close relative loads. The relative load of a peer in KRB is its load divided by its capacity.

The authors tested Swaplinks by emulating a 1000-node P2P network on 20 CPUs and evaluated KRB by simulating a same-scale P2P network. Three conclusions were drawn from the comparison: (1) Swaplinks makes more accurate selections than KRB does; (2) KRB's performance approaches Swaplinks only under low churn and moderate capacity distribution; (3) KRB is sensitive to parameters, and it is harder to set optimal values for the parameters in KRB than in Swaplinks.

INVITED TALK

■ ***Perfect Data in an Imperfect World***

Daniel V. Klein, Consultant

You can find a summary of Dan's talk in the April 2007 issue of *;login:*, in the summaries of LISA '06.

KERNELS

Summarized by Rik Farrow (rik@usenix.org)

■ ***Transparent Checkpoint-Restart of Multiple Processes on Commodity Operating Systems***

Oren Laadan and Jason Nieh, Columbia University

Oren Laadan explained that modern applications consist of multiple processes, so we need a method for capturing global state. This mechanism should be transparent for both applications and sysadmins to use, and it should also not require kernel modifications. Current approaches use modified libraries (an incomplete solution), modified kernels (which is invasive and difficult to maintain), and the use of hypervisors (which implies adding an OS layer and more overheard).

Their approach uses a loadable kernel module that virtualizes just the set of processes to be checkpointed, called a POD, or PrOcess Domain. A POD has a private virtual namespace and is decoupled from the OS. Checkpointing uses auxiliary processes with COW (Copy On Write) and buffers that hold data until it can be committed. Checkpointed processes can have their data filtered to compress it, transform data for another OS version, or adjust data structures. Quiescing a process can be done with SIGSTOP, forcing a known state with minimal stack synchronization. A Process Forest is the set of dependent processes, related either via a parent process or through shared resources all within the same process group.

Oren showed sample output of the DumpForest algorithm, which includes information on dead processes as an artifact of the design. He compared the performance of Checkpoint to OpenVZ and XEN, showing that checkpoint and restart times were 3 to 55 times faster than OpenVZ and 5 to 1100 times faster than Xen. Checkpoint times, at 100 ms, were fast enough not to be noticed by a human being.

Warner from Google said that he had worked with checkpoint in the early 1990s and wondered about network connections in flight, files, and the fact that some processes expect to see the same PIDs after restoration. Oren answered that PIDs are virtualized, isolated via POD, so they don't change. Filesystem issues are addressed using filesystem snapshotting technology. Network connections that are inside the POD are easy to handle. For those outside the POD, we do have a way to move connections if the lag time is short. We can even restart connections on the other side transparently. Someone else asked why the performance is so different than that with the OpenVZ approach. Oren said they were surprised and puzzled too. In restart, OpenVZ was always 1 second longer in OpenVZ. For checkpoint time, they think that teardown and freezing takes longer. They can freeze a process in 1–3 ms. Phil Pennock of Google asked what security analysis had been done and the implications for SUID programs. Oren replied that you have to trust the image that you are restarting.

■ ***Reboots Are for Hardware: Challenges and Solutions to Updating an Operating System on the Fly***

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

Andrew Baumann presented this paper about dynamic updates to the K42 operating system. Previous work exists for applications but is unsuitable for an OS because of low-level languages and concurrency issues in the kernel. They enabled dynamic update by using modularity in the ker-

nel, in contrast to prior work (DynAMOS and LUCOS) that applies patches by rewriting code on the fly, or AutoPOD, which uses virtualization. This work fits somewhere in the middle and uses modules to focus on maintenance changes.

K42 is a scalable research OS that supports Linux API/ABI, is object-oriented, and has each resource managed by a set of object instances. All objects go through an object translation table (pointers) that allows substitution of objects on the fly. A dynamic update is just a series of hot swaps, but you need to replace every module affected. The previous work on K42 would allow some updating via hot-swapping, but not those that include changes to interfaces. A total of 58% of changes affect interfaces in K42. They looked at stable kernel releases of Linux kernels and saw that more than half of the changes affected interfaces as well.

To support dynamic updates, you write an adaptor that can rewrite calls via the old interface to support new function parameters. In testing, the adaptor has 220 cycles of overhead. First, you update the provider object with an adaptor, then update the clients of that object so that they use the new interface, then remove the adaptor. This only works for backward-compatible changes and accounts for about 80% of the changes to the K42 kernel over its history.

Someone asked, What about the remaining 20% of the changes? Andrew answered that outside of module code, such as low-level exception handlers, you can't use this technique. But sometimes you can move the change to a module instead of outside one. Applying a patch produced a drop of 10% while running ReAIM throughput benchmark, with 170 files open. In summary, there is negligible performance impact and 79% of maintenance changes can be updated.

In response to Francis David's question of when it is safe to apply the patch, Andrew explained that the patch is applied as a series of hot-swap operations and must achieve quiescence of the object first, and that makes it safe. Terrence Kelly of HP Labs asked whether it was safe to apply another patch when another lazy update is in process. If you applied another patch, it would mark all the objects as needing updates again, but not break anything.

■ ***Short Paper: Exploring Recovery from Operating System Lockups***

Francis M. David, Jeffrey C. Carlyle, and Roy H. Campbell, University of Illinois at Urbana-Champaign

Francis David started by mentioning that Linux has a lockup detector that works via a timer interrupt that checks a timestamp for continual updates via a low-priority kernel thread. But this mechanism fails if a lockup occurs when interrupts have been disabled. Watchdog timers present an external alternative, as they can generate a nonmaskable interrupt (NMI) if they time out, forcing the system to reboot. However, such reboots mean loss of state, even though the contents of RAM may still be correct.

Francis pointed out that they did their work on ARM CPUs that do not include support for NMI.

In their approach, they replace the interrupt handler for the NMI with code that resets the processor, enables the MMU, reinitializes the interrupt controller and interrupts, and then modifies locked-up threads. In Linux, they simply kill off the offending threads, but in Choices, an object-oriented OS, they patch in a call to an exception handler and restart scheduling. If these threads hold locks, the locks will be released when the threads die. Choices is fully preemptable, and the patched-in recovery routine pretends to be a thread that is locked up and calls die(), or raises a C++ exception so programmers can decide what to do at this point. For this scheme to work, there must be valid context via the stack frame of the thread running before the NMI.

Francis summed up by saying that the best lockup detection uses both hardware and software detection: hardware when software cannot work, and software for preemptible kernel, where hardware cannot detect failure. Their approach improved recovery in Linux up to 9%, particularly with preemptive kernels.

Someone from the University of Rochester asked whether they explored using NMI at all. Francis said that they did, and their code shows examples of this. They wrote the code to actually recover from an NMI as well. Ben Leslie of Open Kernel Labs pointed out that sometimes the state in the OS has been corrupted. You get one watchdog reset, and you keep on doing this. Do you need a meta-watchdog? Francis's team didn't explore corruption issues, or whether the same issue happens twice. Daniel Peek of the University of Michigan asked how much of the kernel malfunction problem this will solve, for example, kernel panics. If you get a "blue screen," the OS has detected the problem. The authors are addressing the lockup problem when the OS can't recognize the error. Someone else asked how often that happens. In the Linux kernel, the majority of possible bugs were lockup bugs (30% or more bugs could cause an infinite loop or other problem in Stanford static code analysis).

INVITED TALK

■ ***Human Computation***

Luis von Ahn, Carnegie Mellon University
Summarized by Minas Gjoka (mgjoka@uci.edu)

Luis von Ahn started by defining the term CAPTCHA (completely automated public Turing test to tell computers and humans apart) as a program that can distinguish humans from computers.

Luis gave a basic example of the usefulness of CAPTCHAs based on a true story in 1999. Slashdot released an online poll letting its users select the best CS grad school from a list of six universities. The only safety measure to prevent

manipulation of the poll results was to allow one vote per unique IP. However, in a matter of hours Carnegie Mellon and MIT students managed to write programs that would cast thousands of votes into the system. This example demonstrated the need for a mechanism that will only allow humans to participate.

Other applications of CAPTCHAs include free email services, worms, data collection, prevention of comment spam in blogs, and dictionary attacks. For example, in free email services, spammers are prevented from signing up millions of accounts automatically. One workaround for spammers is the use of sweatshops, which hire people in countries with very low wages to solve CAPTCHAs for them. That incurs a minimum penalty per account creation for the spammers. Another workaround is to redirect CAPTCHAs from email service companies to the spammer's own Web sites, which provide services that attract many people (e.g., porn sites). In another example, email addresses can be protected from Web crawlers by using CAPTCHAs.

One of the main aspects of the presentation is the usage of CAPTCHAs to perform human computations. A measure of the amount of human effort produced daily: it is reported that around 60 million CAPTCHAs are solved every day, with each CAPTCHA taking 10 seconds of human time. Three programs are presented to make good use of these wasted "human cycles."

CAPTCHAs can be used to help in the digitization of old books. Every scanned image of a word not recognizable by OCR is used as a CAPTCHA. To confirm the correctness of the human input, every time a scanned image of a word is fed into the system it is combined with a known word. If the answer for the known word is correct then a correct answer is assumed for the unknown word as well.

A very useful application of CAPTCHAs is to accurately label images with words. Luis has developed an enjoyable two-player online game, ESP, which was designed in such a way that playing the game results in labeling images correctly, quickly, and for free. At the beginning of the game the user is paired with another random player and the same image is shown to the two players. The goal of the game is to guess descriptions of the image that are identical for both players, excluding taboos. The two players cannot communicate in any way with each other and anticheating techniques are provisioned for potential collaborators. Luis notes that this game alone could be used to label all Google images within a few weeks. In fact Google offers a similar "Google Image Labeler" service. Those who liked using the game listed various reasons (e.g., it offers a special connection with one's partner, it helps one learn English, and it gives one a sense of achievement).

PeekaBoom is another entertaining two-player online game that takes as input labeled images and finds the objects being labeled. The first player, called Boom, receives an image and a tag assigned to the image. The second player, called Peek, has an empty screen. The goal of the game is to get Peek to guess the tag assigned to the image. Boom can only reveal part of the picture. In addition to that, Boom can give hints about what the tag is (e.g., noun, verb, text in the image).

By combining the region selected for a given object from different pairs of players in PeekaBoom it is possible to get the whole outline of the object in 50% of the cases. This allows the results to be highlighted with boxes inside the search engine. Another advantage of this segmentation is that the resulting training set could be used to advance computer vision research.

In conclusion, the speaker presented a paradigm for dealing with open problems in artificial intelligence. These can be turned into either a test to differentiate between humans and computers or a simple game that people can play online.

In response to a question Luis noted that, after 20 hours of playing, the gender of a player can be guessed with 98% accuracy, and the age with 85% accuracy.

SHORT PAPERS

Summarized by Andrew Baumann (andrewb@cse.unsw.edu.au)

■ *Short Paper: Supporting Multiple OSes with OS Switching*

Jun Sun, Dong Zhou, and Steve Longerbeam, DoCoMo USA Labs

Dong Zhou presented this work on a mechanism to switch between operating systems on shared hardware. Each OS is assigned a unique range of physical memory and has exclusive access to the hardware when it runs. The switch is performed when a switch request signal is received; this can be generated by user action or by events such as timer expiration or incoming call. The OS in the background is essentially suspended, it cannot receive interrupts, and there is no regular time-slicing. The switch operation consists of putting the hardware into a consistent state and then passing control to the other OS.

OS switching is usually implemented as a modification of the existing suspend-and-resume support, so relatively little code is changed in the operating system. Furthermore, because each OS runs with direct access to hardware, there is no slowdown. Limitations of the approach include a lack of concurrency and no security between the OS instances, although the latter could be addressed with hardware support such as ARM TrustZone.

A prototype has been implemented on an ARM9 device, and a video was shown of the device switching between Linux and Windows CE in response to a special button press. Around 100 lines of code were changed in either OS, and all within the board support packages; most of the modified code was in the bootloader. Switching from Linux to Windows CE takes half a second; switching to Linux takes a second longer, mainly because more devices

were enabled than under Windows CE. The speed of switching could be improved by not suspending and resuming all devices; in the extreme case, a hot switch could take less than 1 ms.

■ ***Short Paper: Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center***

Cullen Bash and George Forman, Hewlett-Packard Labs

The overall goal of this work, which was presented by both authors, is to reduce the energy used for cooling a data center. This could offer significant cost-savings, because the power used for cooling doesn't scale linearly with the power used to run the equipment (i.e., a rack using twice the power may require much more than twice the power to cool).

Within a data center, some servers can be more efficiently cooled than others because of the varying recirculation of hot air, so the overall cooling workload can be reduced by moving long-running jobs to those servers that are more efficient to cool and shutting down other servers when they are idle. The approach of this work is to modify a scheduler for batch jobs so that the longest-running jobs are placed on the most cooling-efficient servers.

In this study, part of a data center was physically partitioned, and the power consumed by both hosts and air-conditioning units was monitored. The results showed that controlling job placement alone helps, but the greatest power savings come from combining job placement with shutdown of idle servers. This reduced power consumption by 33% and could save $1 million per year for HP data centers. Future work involves incorporating these techniques into adaptive enterprise software.

■ ***Short Paper: Passwords for Everyone: Secure Mnemonic-based Accessible Authentication***

Umut Topkara, Mercan Topkara, and Mikhail J. Atallah, Purdue University

Umut Topkara presented this work, which aims to develop a secure authentication mechanism in input-constrained environments, such as for disabled users. The assumed input device is a binary switch; in this scenario it is hard for humans to remember long bit strings, and it should be possible to initialize passwords with the same input device. Furthermore, the technique should be secure against dictionary, replay, shoulder-surfing, and phishing attacks.

These problems are solved by the PassWit system, which also maps well to traditional plain-text passwords, and thus is compatible with conventional password systems and input devices. At password initialization time, the user is given a random mnemonic sentence selected from a number of word tables. At authentication time, the user is asked a series of yes/no questions, based on the format "Does your mnemonic contain one of these words?"

To avoid record/reply attacks, different questions are asked each time, and to avoid inferring the mnemonic from the questions that are asked, the questions must be determined at the beginning using combinatorial group testing. To protect against spyware, images or CAPTCHA techniques can be used, and the system inherently protects against phishing, because the mnemonic itself is never entered.

■ ***Short Paper: Virtually Shared Displays and User Input Devices***

Grant Wallace and Kai Li, Princeton University

The final paper of the conference, presented by Grant Wallace, covered work on enabling collaboration on shared displays with multiple input devices, for example, the Princeton Plasma Physics Lab control room where a large shared screen is used to allow multiple users to collaborate. Traditional OSes and windowing systems are not appropriate, because they assume the general model of one user at one display with one set of input devices. The goal for this new system is to allow multiple user workstations to connect with each other and the shared display and to allow the users to seamlessly move cursors and windows between the workstation and the shared display.

Traditional collaboration systems (such as X and VNC) are platform-specific, initialization-constrained, support only one-to-many sharing, or share only at the granularity of an entire desktop. To address these limitations the Fusion collaboration system was developed. It uses a modified VNC server to allow sharing at the granularity of windows rather than the desktop, a modified VNC viewer to simultaneously display windows from multiple users, a modified window manager that supports multiple cursors by time-slicing cursor activity to the system cursor, and a modified X2X utility that captures input from multiple users.

The system has been deployed in two locations and has received very positive user feedback. It enables users to share and compare windows while providing better performance and privacy than the desktop sharing of normal VNC. Further details and source code are available at http://shared-app-vnc.sourceforge.net and http://multicursor-wm.sourceforge.net.

INVITED TALK

■ ***Warehouse-scale Computers***

Luiz André Barroso, Google Inc.
Summarized by Minas Gjoka (mgjoka@uci.edu)

Luiz André Barroso said he intended to describe the characteristics of warehouse-scale computing infrastructure at Google from the hardware standpoint. Nowadays, warehouses are becoming more cost-effective for many companies, and in the near future new technology advancements may produce machines that will have properties that need to be tackled in today's warehouse-sized computers, such as thread concurrency, power saving, complexity manage-

ment, and fault handling. Thousands of programs in different machines should work as a reliable platform running different services.

The first topic discussed was the programming efficiency for such systems. The need for parallelism to handle large amounts of data, heterogeneity, and failure-prone components complicates programming. A single programming system or language may not be enough. Instead Luiz advocated that the solution should be higher-level and use-specialized building blocks for large-scale distributed systems such as MapReduce and BigTable.

When building fault-tolerant software, it is important to guarantee that system interruption does not occur; otherwise some of the worst performance problems may appear. Service-level measurements that monitor performance provide only a partial view. Instead, Google has built a System Health infrastructure that collects health signals from all its servers and stores these signals perpetually in time series. Using this infrastructure, an analysis of hard disk failures was performed out of detailed signals collected during a period of nine months from a five-year inventory database. Understanding when such failures occur should, ideally, give a prediction model for failures that would allow preemptive action. (You can learn more about this project by reading the Pinheiro et al. paper that appeared at FAST '07 or the summary that appeared in the June 2007 issue of ;*login:*.)

Grouping disks by age did not give conclusive results, because of the different hard drive model mixtures in the data. An interesting finding is that temperature has little impact on the average failure rate. In fact, higher failure rates are observed at lower temperatures.

Signals were collected from the standard SMART interface of hard disks, in an effort to build a predictive failure model. The results showed that only a subset of the SMART signals are strong indicators of future failures. For example, drives with scan errors are ten times more likely to fail. However, the predictive power of SMART signals seems limited, since almost half of the failures appear unpredictable when the set of strong indicators is used.

The cost of operation, in terms of energy and maximization of utilization, needs to be taken into account for warehouse building. The former becomes even more important since, unlike hardware costs, energy prices are increasing. Luiz mentioned that energy costs (excluding cooling) can account for up to 20% of the company's IT budget. Part of the solution lies in improving the efficiency of power supplies, which ranges from 55% to 70% nowadays, by reducing conversion losses. It is easy to see that with 55% efficiency the power supply becomes the largest power consumer inside a machine.

The goal of the maximization of utilization is to maximize the facility usage without exceeding contractual capacity limits. A six-month power monitoring study was conducted at Google to examine opportunities in power subscriptions. The study included three machine aggregation levels (rack, power distribution unit, cluster), with each almost an order of magnitude larger then the previous one. The analysis showed that at the cluster level the normalized power never exceeded 71%, which leaves room for more servers to be packed in the warehouse.

Given that current machines usually consume around 50% of their peak rate at idle mode, simulations for potential improvements in power consumption behavior were performed. The idea was to assume the availability of active low-power modes (at most, 5% of peak power). The simulation results showed remarkable improvements for both peak power and energy at the cluster level. It was suggested that power-saving features be implemented for other components in addition to the CPU and that a wide dynamic power range with low consumption at idle mode be included.

In conclusion, Luiz reiterated the benefits of understanding failures and emphasized the potential for power and energy efficiency. Most questions wandered around the issue of power savings.

Information for the climate savers computing initiative referenced by the speaker can be found at http://www.climatesaverscomputing.org/.

PLENARY CLOSING SESSION

Crossing the Digital Divide: The Latest Efforts from One Laptop per Child

Mary Lou Jepsen, CTO, One Laptop per Child
Summarized by Rik Farrow (rik@usenix.org)

Mary Lou Jepson said that because she has been traveling so much, promoting One Laptop per Child (OLPC), it is hard for her to keep track of what time zone she presently inhabits. Some of her jetlag was apparent in her somewhat rambling talk, but I still found what she had to say fascinating. You can find transcripts of recent talks by Jepson at http://www.olpctalks.com/mary_lou_jepsen/ and more about the hardware of the current version of the laptop, the XO rev C, at http://wiki.laptop.org/go/Hardware_specification.

Jepson had been an engineer at Intel specializing in display technologies. She went to work for OLPC as it was realized that the single most expensive part of a PC that is designed to be cheap would be the display. She explained two innovative features of her display design which reduce cost and power while increasing usability. The first innovation is the creation of dual-mode display cells. Most LCD displays rely on backlighting that shines through a color filter, then through the liquid crystal cell, which is more or less transparent. That means the display focuses on chrominance, whereas the human eye is actually more sensitive to luminance. She designed a screen that has greater resolution in

reflective (black-and-white) mode, 200 dots per inch, which makes the screen very easy to read. A significant motivation for this design, besides the use of less power than with backlighting, is that it can store and display textbooks. Instead of buying textbooks for children, countries can buy XOs and upload textbooks, which will pay for the laptop over its expected lifetime. When used with backlight, the display goes from 1200x900 mono to 800x600 color, and it may use as much as ten times as much power because of the backlighting.

The other innovation has to do with refreshing the screen while the CPU is idle. Normally, the CPU must copy data to refresh the display 30 times per second, but by replacing the ordinary display controller chip with one that has memory, the chip takes over refreshes for the CPU, allowing the CPU to sleep until needed.

The XO has other energy-saving innovations, such as the use of a Marvel WiFi chip that includes part of an ARM CPU. That means that a laptop can function as part of a mesh network even when it is otherwise idle, as the Marvel chip handles the processing. The storage is a 1024-MB NAND flash and 256 MB of RAM. Jepson also showed examples of robust servers designed to act as both additional storage and connections to external networks.

The onetime symbol of the laptop, the handcrank, has been replaced with swivel-up ears that contain the WiFi antennas. When swiveled down, the ears cover external connectors, such as USB ports. Power can come from solar cells or generators, including a salad-spinner design for hand-charging.

The display itself is gorgeous, and the system is certainly sturdy. Jepson explained that laptops at a test site in Nigeria were breaking after only three months (but had been expected to last five years). It turns out that the desktops at the test site are slanted, and the laptops were falling off the desks onto the concrete floors at the rate of several times an hour (and continued to work for three months!). The XO is designed so that it can be repaired locally by swapping out parts, and extra case screws are included to replace lost screws.

The laptop runs a version of Linux, and it uses Sugar as the GUI framework. Software is designed so that the laptop can be used by children who cannot yet read. Jepson told us of a project in India where children taught themselves how to read by having some access to computer displays.

The simple, rugged, low-power, and low-cost design makes the XO not just an ideal textbook replacement but a desirable product in other worldwide markets. I commented to Jepson that licensing the display and some of the other technology might be one way of supporting further development of the OLPC vision.

Other questioners had darker views of the project. Tristan Lawrence said that he expected that the laptops will never reach their intended recipients, suggesting that the governments will distribute the laptops to better-off city dwellers, rather than the apparent targets of the project. Jepson answered that they have focused on teaching and designing a low-power laptop with mesh networking and a usable display, not on politics. She did mention Bitfrost, the security used to help prevent theft of laptops. Laptops must be updated with a signed key once every several weeks, or they will stop working. The laptops can also be remotely disabled if stolen. For more on Bitfrost and the security model of OLPC, see http://lwn.net/Articles/221052/.

Aarjav Trivedi of Secure Computing said that he is from India, a country that has so far decided not to buy the laptop. Trivedi declared that content is key and wondered if they had found local content in India. Jepson said that they have been working with local people to scan books. Even though India and China have been cool to the project so far, half the children in the world live in China and India.

Quando Lee asked about textbook cost comparisons. Jepson answered that, for example, in Brazil, textbooks cost $20/year, so over its expected five-year lifetime, the XO pays for itself. Experience in China showed that kids allowed to read anything they wanted learned five times as many Chinese characters as other children with less to read. The same person said that textbooks can last more than five years and that he had used his older brother's books. Jepson responded by saying that textbooks cost $643 per year in Massachusetts.

Marc Fuscinski said that he had visited some site in Sao Paolo, Brazil, but the kids can't take the laptops home. Jepson says that is certainly true. But in other places kids are starting a "right to laptop" movement, and in Cambodia, Thailand, and Nigeria they can take them home.

Someone wondered why the laptop couldn't last more than five years. Jepson replied that the LCD will last for half a million hours in sunlight, but the flash memory has a limited number of write cycles (because of wear leveling). The same person asked whether the laptop is recyclable, and Jepson answered that if a laptop stops working, it can be given to a post office, where it will be sent to a central depot for repair or recycling. The entire device is green, and it costs more to ship it than to recycle it.

George Herbert of Open Software Foundation asked whether the OLPC planned on frontloading open content. Jepson's understanding is that the countries will choose what content they want on the server. For example, they ask the participating country to pick the top 100 books appropriate for kids to read, while they provide a Mathematica lite version, along with reading and drawing tools. Kids can also program the computers themselves, using Python or Logo.

Warren Henson of Google asked about other plans for long-term storage and backup. Jepson said that that sounds like a great thing for Google to do. Right now they are stuck with the server (a low-power device, sealed with a hard disk) and aren't currently addressing that problem, although Google has provided gmail accounts. In response to Henson's mention of alternative projects from other organizations, Jepson said that they would like to work with these groups and try every week. When these efforts fail, the kids lose, in her opinion. Since Jepson spoke, Intel has announced that they plan to work with OLPC, and Intel is now listed as a supporter on the laptop.org site.