

Conference Reports

In this issue:

2011 USENIX Annual Technical Conference

(<http://www.usenix.org/events/atc11/>) 61

Summarized by Raluca Ada Popa, Nadav Amit, Muli Ben-Yehuda, Rik Farrow, Vishakha Gupta, Etienne Le Sueur, Dan Levin, Timothy Merrifield, Muhammad Shahzad, Mark Spear, and Carsten Weinhold

2nd USENIX Conference on Web Application Development

(<http://www.usenix.org/events/webapps11/>) 81

Summarized by Ioannis Papagiannis and Veena Udayabhanu

3rd USENIX Workshop on Hot Topics in Cloud Computing

(<http://www.usenix.org/events/hotcloud11/>) 86

Summarized by Tian Guo, Henrique Rodrigues, Sahil Suneja, and Byung Chul Tak

3rd USENIX Workshop on Hot Topics in Storage and File Systems

(<http://www.usenix.org/events/hotstorage11/>) 98

Summarized by Dutch Meyer, Sejin Park, and Luis Useche

3rd Workshop on I/O Virtualization

(<http://www.usenix.org/events/wiov11/>) 106

Summarized by Sejin Park, Henrique Rodrigues, and Muli Ben-Yehuda

3rd USENIX Workshop on Hot Topics in Parallelism

(<http://www.usenix.org/events/hotpar11/>) 108

Summarized by Amittai Aviram, Bryan Catanzaro, Sean Halle, Craig Mustard, Shane Mottishaw, Sabrina M. Neuman, and Jaswanth Sreeram

2011 USENIX Annual Technical Conference (USENIX ATC '11)

Portland, Oregon

June 15–17, 2011

Welcome, Awards, and Keynote Address

Opening Remarks and Awards Presentation (USENIX Lifetime Achievement Award, Software Tools User Group [STUG] Award, CRA-W BECA Award, and Best Paper Awards)

Jason Nieh and Carl Waldspurger, USENIX ATC '11 Program Co-Chairs; Clem Cole and Margo Seltzer, USENIX Board of Directors

Summarized by Rik Farrow (rik@usenix.org)

Jason Nieh and Carl Waldspurger, the co-chairs of ATC, presented the opening remarks with thanks to the program committee, who performed at least 25 paper reviews each. Out of 180 submitted papers, 38 were accepted. The Best Paper Awards went to “Building a High-performance Deduplication System,” by Fanglu Guo and Petros Efstathopoulos of Symantec Research Labs, and to “Semantics of Caching with SPOCA: A Stateless, Proportional, Optimally-Consistent Addressing Algorithm,” by Ashish Chawla, Benjamin Reed, Karl Juhnke, and Ghousuddin Syed of Yahoo! Inc.

Clem Cole, USENIX President, then presented the Lifetime Achievement Award to Dan Geer. Dan, who was unable to attend because of commitments made prior to learning of the award, created a video presentation for his acceptance speech (<http://www.usenix.org/about/flame.html>). Next, Clem presented the Software Tools User Group Award to Fabrice Bellard for QEMU. QEMU has become essential for many forms of virtualization today, and Fabrice wrote QEMU without expecting or demanding any financial reward.

Margo Seltzer, USENIX Vice President, presented the CRA-W BECA Award to Alexandra (Sasha) Fedorova of Simon Fraser University. This is the Anita Borg early career award, and Margo said that “giving it to my own student makes it even better.” Sasha thanked Margo, her colleagues



at Sun Labs where she spent three years as an intern during her PhD studies, and her colleagues who taught her about the quality of work and work ethics.

Keynote Address: An Agenda for Empirical Cyber Crime Research

Stefan Savage, Director of the Collaborative Center for Internet Epidemiology and Defenses (CCIED) and Associate Professor, UCSD

Summarized by Raluca Ada Popa (ralucap@mit.edu)

Despite progress in the academic approach to security research aimed at bots and viruses, malware continues to grow in the real world. The fundamental flaw of the academic approach is that it examines the problem solely technically; the cycle of improving security measures while spammers become more agile is never-ending.

Stefan Savage proposes an economic approach that seeks to understand the business model and value chain of the attacks in order to focus security interventions at the most sensitive spots. He describes the ecosystem of spam: some parties would pay to advertise for them, some would sell a bot, some would sell storage, and some would even sell entire companies. To measure the value chain empirically, Savage's team had to engage with the attacker. In the Spamalytics paper, the authors infiltrate an existing botnet [Storm] infrastructure to measure, for example, the chance that spam results in a sale. Another insight such economic study brings is that, while CAPTCHAs are technically not a good solution for spam because there is automated software for breaking them, they are a successful economic filter; they limit the set of abusers to those who can afford to hire people to solve CAPTCHAs or use such software effectively.

In the recent paper "Click Trajectories," the authors presented a comprehensive analysis of the resources used to monetize spam email, such as naming, hosting, payment, and fulfillment. They identified a bottleneck in the spam value chain: most spam products are monetized using a small number of banks. This means that convincing banks to not serve spammers, by having U.S. banks refuse to settle transactions with banks identified as supporting spammers, for example, may effectively counter spam. Savage concluded his talk by remarking that a similar type of in-depth economic analysis could likely be applicable to other security problems as well.

During Q&A, an attendee asked whether charging for sending email is feasible or effective at stopping spam. Savage answered that such an approach is unlikely to work, for two reasons: first, many times, spammers use other users' machines to send spam, and, second, it is hard to get everyone to adopt it at once. Another attendee expressed surprise at the little payment that CAPTCHA human solvers receive: ~\$3 per day. Savage's response was that such workers do not even need to be literate; they can just use a printed alphabet. He added that a \$3/day wage is common in China.

Scheduling

Summarized by Timothy Merrifield (tmerrif4@uic.edu)

A Case for NUMA-aware Contention Management on Multicore Systems

Sergey Blagodurov, Sergey Zhuravlev, Mohammad Dashti, and Alexandra Fedorova, Simon Fraser University

Sergey Blagodurov presented his group's work on schedulers that perform contention management on NUMA systems. Because previous work on contention management has assumed uniform memory access (UMA), threads can be migrated between cores without consideration of where their memory resides. Blagodurov and his group contend that because real systems may have non-uniform memory access (NUMA), additional considerations are needed to ensure that thread migrations don't cause added contention.

Experimentally, the group observed that the most dominant contention factors (performed over several benchmarks) in NUMA systems are the InterConnect and the memory controller. Shared cache contention and latency for accessing remote memory were also factors, but not nearly as pronounced as the prior two. This group previously devised DI (or DI-Plain), which detects resource contention between threads (using the last-level cache miss-rate as a heuristic) and attempts to separate competing threads such that they reside on different cores or domains. Blagodurov explained that on a NUMA system, unless the memory is also migrated to the new domain, contention for the memory controller

will remain. This explains DI-Plain’s poor performance on NUMA systems.

From these observations the group designed a new scheduler: DI-Migrate. When performing a thread migration, DI-Migrate also migrates the hot (most accessed) pages for that thread to the new domain. While this showed some improvement, for certain benchmarks it was determined that too many migrations were occurring. Finally, the DINO scheduler was designed to both migrate memory and attempt to reduce the number of migrations, thus reducing the amount of memory that needed to be copied. DINO showed positive results on SPEC 2006, SPEC 2007, and LAMP benchmarks.

Josh Triplett (Portland State) asked about the possibility of migrating a thread to where its memory was located, to avoid the overhead of copying memory to the new thread domain. Blagodurov explained that memory is allocated locally (closest node) by default in Linux, so if a thread is going to be migrated to a different domain it will always be moved away from its memory. Carl Waldspurger asked about how DINO works with thread priorities. Blagodurov answered that priorities can be preserved by trying to keep high-priority threads free of contention. This work has been partially published, but he said that perhaps a paper more focused on prioritization is still to come. The final questioner asked how this approach would work in a virtualized environment. Blagodurov saw no real issues, because the VMs would just be treated as threads by the scheduler; he added that further experimentation would need to be done to verify this.

TimeGraph: GPU Scheduling for Real-Time Multi-Tasking Environments

Shinpei Kato, Carnegie Mellon University and The University of Tokyo;
Karthik Lakshmanan and Ragunathan Rajkumar, Carnegie Mellon University; Yutaka Ishikawa, The University of Tokyo

Shinpei Kato described the fundamental multi-tasking problems inherent in how the GPU is controlled, explaining that the GPU is command-driven (via the device driver) and that if low-priority tasks send many commands to the GPU, the device can become slow or even unresponsive. This theory is backed up by experimental data which shows that the frame-rate of a graphics processing task can be highly influenced by competing, GPU-intensive workloads.

Kato presented TimeGraph, a GPU scheduler that sits on top of the device driver (but still in kernel space) and attempts to schedule GPU commands. For example, if the device is busy when a command is submitted, TimeGraph may queue the command and schedule it sometime in the future. Kato also explained that TimeGraph supports reservations to ensure that the GPU does not suffer due to overutilization by

a single task. TimeGraph also uses a history-based approach to attempt to estimate the execution time for a given GPU command.

In the evaluation section, Kato presented experimental data showing the frame-rates of a 3-D game when co-scheduled with a graphics-intensive, low-priority task. TimeGraph does significantly better than the default (no scheduling) with just priority support and further improves when reservation support is added. Kato acknowledged that there is some overhead in the use of TimeGraph, but believes the benefits far outweigh the drawbacks.

Ali Mashtizadeh (VMware) asked whether memory management (copying between devices) command scheduling was also possible. Kato explained that memory commands could also be scheduled by TimeGraph in a similar way and that perhaps there should be some communication between the user-space libraries and TimeGraph indicating which commands should be queued and which should be sent directly to the driver. Kai Shen (University of Rochester) was curious why TimeGraph was implemented in kernel space by instrumenting the device driver. He asked if there were reasons that TimeGraph could not be implemented on top of the driver (in user space) so that it would work with closed-source drivers. Kato replied that there are two reasons for this: first, many user space libraries are also closed-source, so that will not work; second, a user space scheduler can be preempted, which can compromise performance. A final questioner discussed the difficulty in doing online command runtime prediction and asked how this is done in TimeGraph. Kato said that he agrees that online predictions are very hard and he doesn’t recommend predicting at runtime. He added that perhaps additional coordination between user space (where compilers and runtimes exist) and kernel space could be used to provide additional context to help guide the prediction engine.

Pegasus: Coordinated Scheduling for Virtualized Accelerator-based Systems

Vishakha Gupta and Karsten Schwan, Georgia Institute of Technology;
Niraj Tolia, Maginatics; Vanish Talwar and Parthasarathy Ranganathan, HP Labs

Vishakha Gupta discussed the shortcomings of the current GPU programming models (CUDA, most prominently) from the perspective of the system designer. So much of the logic for working with GPUs is encapsulated in closed-source drivers and runtime libraries, which turns these devices into black boxes with a “one size fits all” approach. With future systems making heavy use of accelerators (GPUs), Gupta recommended that we begin to think differently about these devices.

Pegasus is designed for virtualized systems and runs on top of Xen, with much of the logic residing in Dom0 (Xen's management domain). Gupta explained that the driver and runtime for the GPU will reside in Dom0 and will be accessed by any virtual machine that intends to use the device. In addition, custom modules are added in the virtual machine (front-end) and in Dom0 (back-end), which sit on top of the underlying drivers and provide additional functionality. The front-end driver will queue commands into a buffer shared with Dom0, while the back end will periodically poll the buffer for commands and data to send to the GPU for execution. This technique allows scheduling of GPU commands to occur by choosing which virtual machine to pool from next. Several different scheduling techniques were evaluated over various benchmark applications. Gupta presented data from experimental evaluations showing improvements in both throughput and latency.

Kai Shen (University of Rochester) asked whether such a technique would be feasible on a non-virtualized system. Shen thought that perhaps a scheduling daemon could sit on top of the CUDA runtime and intercept calls and enforce a similar scheduling policy. Gupta agreed and said that such a system was built at Georgia Tech. Shen also asked how this approach would work in non-Xen virtualization environments where hypervisors sit below device drivers. Gupta said that regardless of the virtualization system, the principles of intercepting and queueing calls will continue to be applicable. Because of his interest in open source GPU runtimes and drivers, Shinpei Kato (CMU) asked how the implementation would change if those modules were open and available for modification. Gupta replied that the principles of their approach would be similar but that they would definitely benefit from open source implementations of these drivers.

A final questioner asked about non-compute (graphics) GPU commands and how Pegasus deals with them. Gupta agreed that those commands should also be run through Pegasus but that this is more of an implementation task and would provide little research benefit.

Virtualization

Summarized by Vishakha Gupta (vishakha@cc.gatech.edu)

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO

Irfan Ahmad, Ajay Gulati, and Ali Mashtizadeh, VMware, Inc.

Irfan started the session right after lunch on a light note by presenting the history of "Interrupt Coalescing," starting from the work of Dijkstra to the patents filed so far and the implementations seen. He briefly described existing

techniques and coined the abbreviations MIDL (maximum interrupt delay latency) and MCC (maximum coalesce count) to talk about past work on network interrupt coalescing done in the 1970s. The same techniques would also work for storage devices. However, virtualization has introduced a layer between the real hardware and the virtual machines. The existing techniques based on high-resolution timers become less practical for virtual devices, due to their large overheads. So they looked at the number of commands in flight (CIF) to set the delay timer instead of using a high-resolution timer.

Irfan then presented their technique, called vIC (virtual Interrupt Coalescing), which has been working in the VMware ESX server since 2009. vIC uses CIFs to set the fine-grained delivery ratio. They avoid high-resolution timers because virtual interrupts are different: (1) they are not in hardware-like typical controllers; (2) their execution is emulated on general-purpose cores in time-shared architectures; and (c) timers of 100 microsecond granularity can overwhelm the virtual machine (VM). So, instead, they piggyback delivery of interrupts on the real hardware completion handlers. The naive delivery ratio of one interrupt every n hardware completions doesn't allow for expressing percentage, and the jump in performance can be drastic. Hence, they use two counting parameters: countUp and skipUp. And their method is smart about how they vary the delivery ratio based on the intuition that coalescing 32 CIFs keeps the pipeline full but four CIFs reduces it to half! Low CIFs can drop throughput, so they vary the delivery ratio inversely with CIF. They also have a short circuit for low CIF situations to handle absence of hardware completion. They can always dynamically enable or disable vIC. They have paid special attention to portability and low cost. vIC can be implemented in firmware or hardware, doesn't use floating point or integer division, and does not use RDTSC. It results in a very small increase in VMM code size, and they showed performance benefits for multiple application benchmarks, including certain nuances due to burstiness. He ended his talk with certain questions for future research on developing something similar for networks and whether IPI coalescing would be beneficial.

Raju Rangaswami (Florida International) asked whether they ran into some corner case where they really needed a timer (e.g., 5 sec. SSD delay) when there were a lot of writes in flight. Irfan said there are opportunities in the hypervisor to inspect if interrupts have been delivered and there are optimizations. The current architecture gives them opportunities to inspect the completion queue whenever a VM exits into the VMM. All these things add up to a tight bound on maximum latency, but they can adjust. Raju asked whether they need to worry about the OS doing something it

shouldn't because the time of interrupt delivery gets changed. Irfan replied that there is pathological behavior, especially in certain old operating systems, if an I/O device never returns. So people have looked at longer periods of keep-alive mode. However, all those behaviors have a range in seconds, whereas with vIC, interrupts never sit in the queue for so long and get delivered at the latest with the host timer. Alexandra Fedorova (SFU) asked about the effect of the technique on performance variability of certain workloads and end-to-end performance. Irfan acknowledged that there is some inherent variability in I/O completion times, but they haven't done precise studies to see if that variability is increased due to this technique. However, their experience with I/O path variability indicates that overall variability just becomes more structured. In fact it could be possible to lower variability by slowing down fast I/O in the same way that it happens for hardware variability at times. The last questioner noted that their results were conflicting, since interrupts on current machines take less than microseconds; however, their coalescing period is high, especially as seen for TPC-C. Irfan responded that in virtualization, interrupts cause VM exits, cache pollution, etc. TPC-C is sensitive to cache and just by improving that, they see benefits.

Power Budgeting for Virtualized Data Centers

Harold Lim, Duke University; Aman Kansal and Jie Liu, Microsoft Research

Current methods for budgeting power in datacenters—allocating much lower than peak power for over-subscribed systems and power capping in hardware—fall short because of virtualization. Harold Lim discussed challenges to application-aware power budgeting, such as: (1) the disconnect between physical layout and logical organization of resources that require application-aware capping, (2) the need for multi-dimensional power control (e.g., DVFS and CPU speed caps), and (3) dynamic power allocations in datacenters to respond to variable workloads and power requirements among applications.

He presented Virtualized Power Shifting (VPS), a system for power budgeting for virtualized infrastructures. The main characteristics of VPS are: (1) application awareness to distinguish between different application requirements; (2) dynamic power shifting as workload changes, through its multi-level controllers; (3) exploitation of performance information (if available); and (4) multiple power knobs.

VPS has multiple dynamic controllers: (1) a top-level shared infrastructure feedback PID controller that adapts to dynamic workload and power budget; (2) weighted fair sharing for individual applications, using priorities based on the total application budget for the next time period; (3)

an application-level PID controller to distribute power for each tier as a cascading effect of controlling the power knob for the first tier; and (4) tier-level controllers. There are three tier-level controller options: open loop control (easy and instantaneous but no error compensation), PID control (slower, single control knob but compensates for error), and model predictive control (optimizes performance with multiple knobs but requires performance measurement and systems models that relate control knobs to system states). They evaluated tradeoffs for these techniques against accuracy, speed, amount of application information, and models required. Harold concluded his talk with some performance analysis and showing how VPS gives low budgeting errors compared to physical hierarchy controllers.

Sergey Blagodurov (SFU) wondered, given alternative sources of power and the need to allocate the power budget for a constant set of workloads, whether they would consider prioritizing what to shut down. Harold indicated that they do prioritize between applications and if the overall datacenter power changes, their models should adapt. They have started looking at VM migrations and latency issues, but they had not evaluated shut-down alternatives. Bhuvan Uргаonkar (Penn State) asked if they are trying to achieve power budget at VM granularity and Harold agreed to that except for granularity of an application at the VM level. To a question on accounting for shared resources when VMs are co-located on one server and how much error there could be, Harold said they have used JouleMeter to tackle this and lower errors. The last question was on accuracy, since the system won't work if the model is not on. Harold said they used an uncontrollable power parameter to compensate.

vIOMMU: Efficient IOMMU Emulation

Nadav Amit and Muli Ben-Yehuda, Technion and IBM Research; Dan Tsafir and Assaf Schuster, Technion

Nadav Amit started out with some background on how IOMMUs have solved the problem of making DMA available to virtual machines by splitting devices in protection domains through translation of addresses in an I/O context. This has enabled direct device assignment so that guests can interact with I/O devices directly, delivering the best performance for unmodified guests. However, guest to machine physical translation methods have shortcomings: (1) they cannot do memory over-commitment; (2) there's no intra-guest protection—a guest is vulnerable to faulty device drivers and malicious I/O devices; and (3) there's no redirection of DMA—a guest cannot use 32-bit I/O devices, nested VT, or custom mappings.

Nadav presented vIOMMU, their emulation of a physical IOMMU available in the system. While the I/O devices still

get assigned to guests, the focus of their work is to improve efficiency. IOMMU emulation, similar to handling memory for virtualized guests, is intended to overcome the shortcomings mentioned above. More importantly, IOMMU emulation can be achieved efficiently by: (1) a delayed teardown of IOMMU mappings in the hope that it will be immediately reused (“optimistic teardown”), since each map/unmap results in a VM-exit that can be expensive; and (2) sidecore emulation that avoids VM exits. The guest writes in a shared region of emulated registers that are polled by the sidecore, which emulates hardware behavior. Devices need to have certain properties, including synchronous register write protocol, loose response time, and so on. Unlike IOVA allocation methods, “optimistic teardown” defers unmappings until a quota of stale mappings is reached or a time limit elapses. Stale mappings are always reclaimed, as in the Linux default. Their evaluation of the sidecore emulation shows it’s three times faster than trap-and-emulate, and the optimizations provided by optimistic teardown add to the benefits. Sidecore emulation can reduce mapping overhead significantly with only a modest protection compromise, and future hardware support will help them deliver even better performance.

Josh Triplett (Portland State) asked if their system would work when the hardware did not provide physical IOMMU support. Nadav said that they cannot allow direct access to physical hardware to any guest, which is the case when there is no IOMMU. What would happen if there was a less-than-capable IOMMU? Nadav said that they can always emulate one that is different. Himanshu Raj (Microsoft) asked what Nadav meant by modest protection compromise and whether it would actually turn out to be a correctness compromise. Nadav answered that since they defer as with the Linux default, there is a short period with potentially stale mappings. In this case, you may access memory of the same guest without holding the actual I/O buffer, and you may override or access data when it is not allowed. However, it’s the same security compromise as in native Linux, and should be fine in the absence of a malicious device driver.

Cloud Computing and Data Centers

Summarized by Muli Ben-Yehuda (mulib@cs.technion.ac.il)

HiTune: Dataflow-Based Performance Analysis for Big Data Cloud

Jinquan Dai, Jie Huang, Shengsheng Huang, Bo Huang, and Yan Liu, Intel Asia-Pacific Research and Development Ltd.

In the first talk of this joint USENIX ATC/HotCloud session, Jinquan Dai presented HiTune, a performance analysis tool for “big data” clouds. Jinquan started by explaining the dataflow model for big data analytics, where applications are

modeled as dataflow graphs, the user writes subroutines that operate on the vertices, and the runtime takes care of all of the messy low-level details. This model is implemented in the popular Hadoop system. Although this is a useful model for parallel programming, the complexity of the system means that it often appears to the user as a black box, and in particular, it can be very difficult for the user to understand, analyze, and tune runtime performance.

HiTune is an open source performance analysis tool for Hadoop, a “VTune for Hadoop.” It uses lightweight binary instrumentation to reconstruct the dataflow execution process of the user’s jobs with “extremely low”—less than 2%—runtime overhead. Reconstructing the dataflow execution makes it possible to pinpoint and understand performance bottlenecks. Jinquan presented several case studies from real users where “traditional” tools were not helpful in understanding suboptimal behavior (e.g., because the Hadoop cluster was very lightly utilized), but HiTune presented the information in such a way that the root causes of performance degradation became obvious to the user.

Alex Snoeren from UCSD pointed out that in the case studies that Jinquan showed, once the user had an inkling of what was going on, VTune could have been used to dig deeper and give supporting evidence. He then asked the extent to which these sorts of discoveries could be automated so that customers could do it on their own, without needing expert assistance. Jinquan replied that first, at least in some cases, VTune won’t help because it will show the system as idle (there are no hot spots); second, HiTune is meant to present the data in a way that the user could make sense of it—someone still needs to understand the high-level model and figure out how to fix the problems HiTune uncovers.

Taming the Flying Cable Monster: A Topology Design and Optimization Framework for Data-Center Networks

Jayaram Mudigonda, Praveen Yalagandula, and Jeffrey C. Mogul, HP Labs

Praveen Yalagandula first introduced us to the Flying Cable Monster, “the new research area” of datacenter topology design and wiring, and then proceeded to tame it. The goal of datacenter designers is to design a cost-effective network. Designing a network requires that the designer choose the network topology, which switches and cables to buy, and how to organize servers and lay out racks. Previous works often looked at this problem as a logical optimization problem, but Praveen and his co-authors look at it from a practical standpoint, considering, for example, the need to buy (expensive) cables and wire the datacenter in a way that maximizes performance and minimizes the overall cost.

The flying cable monster is a fearsome beast; the main tool Praveen and his co-authors use to tame it is Perseus, a framework which assists network designers in exploring the different design considerations. The designer first provides Perseus with several inputs (e.g., the number of servers and the desired bi-section bandwidth); Perseus then generates candidate network topologies and potential physical layouts (wiring schemes), computes the overall cost of each design, and provides visual results to the user. Perseus is currently a preliminary prototype, but is already useful. Future improvements include scalability and visualization enhancements, the ability to generate wiring instructions, and the ability to verify that a given installation matches the original design.

This presentation generated relatively many questions. Sergey Blagodurov of Simon Fraser University asked about the cooling problem, where server placement is also affected by cooling considerations, not just the length of wires. Praveen replied that cooling and power considerations are not handled by Perseus at the moment, but could be added. Alex Snoeren remarked that network topology and wiring is obviously a hard problem, but how often does it need to be solved in practice? Praveen replied that it's usually done once per datacenter, but vendors such as HP who sell datacenters do it all the time. Himanshu Raj of Microsoft asked whether "containerized" datacenters—where the datacenter is built and shipped inside a container as a single pre-fabricated unit—change the equation. Praveen replied that Perseus is a tool for the person designing the container, not necessarily the person buying it. The last question had to do with different network topologies supported by the tool—how many do we really need? Praveen replied that different people have different requirements from their networks, and different topologies address those different requirements.

In-situ MapReduce for Log Processing

Dionysios Logothetis, University of California, San Diego; Chris Trezzo, Salesforce.com, Inc.; Kevin C. Webb and Kenneth Yocum, University of California, San Diego

Dionysios Logothetis started by explaining the impetus of "log analytics," storing and analyzing terabytes of logs with valuable information. Organizations mine their logs for such diverse purposes as ad targeting, personalization, brand monitoring, fraud and anomaly detection, and debugging. Log analytics today is an offline process, where logs are first stored somewhere and later queried offline. This offline approach, however, has several drawbacks. First, there is the sheer scale of the problem: Facebook, as a concrete example, collects over a hundred terabytes of logs each day; second, storing-and-querying is susceptible to server failures, which can delay analysis or cause the servers to process incomplete

data; third, there is the issue of timeliness—offline queries hinder development of real-time applications of log analytics such as ad-retargeting based on the user's most recent "likes."

Dionysios and his co-authors address these drawbacks of offline log analytics through in-situ MapReduce (iMR), which turns log analytics into an online process, where logs are continuously generated, filtered, and analyzed on the same servers. Regular MapReduce (MR) takes some input, processes it, and provides results. iMR, in contrast, continuously processes incoming data and provides results. The main insight behind iMR is that it is possible to trade off fidelity for speed. Turning MR into an online process requires dealing with continuous incoming data and with failures. iMR deals with continuous incoming data through "sliding windows" pointing into the incoming data and through the further division of windows into panes. Each window pane is only transferred over the network and analyzed once.

iMR deals with failures, as well as situations where servers get overloaded (both of these conditions can hurt the timeliness of the analysis) by allowing users to trade fidelity with latency through the C2 fidelity metric. This metric allows users to control the fidelity/latency tradeoff and also lets them better understand the impact of failures on the analysis by exposing the temporal and spatial nature of logs. Users specify either maximum latency requirements ("process each sliding window within 60 seconds") or minimum fidelity ("process at least 50% of the total data"), which the iMR runtime satisfies through intelligent selection of which panes to process and on which servers to process them. The authors implemented an iMR prototype on top of the Mortar distributed stream processor.

Praveen Yalagandula of HP Labs asked, since iMR uses aggregation trees, what happens when an intermediate node decides to discard a pane after it has already aggregated data from other nodes down the tree. In this case the intermediate node loses not only its own data but also data from other nodes. Dionysios replied that while iMR tries to be proactive—each node can notify other nodes that it is busy at the moment and cannot handle more data—it's still possible for nodes to become overloaded and shed load this way. Having said that, this is a common problem in aggregation trees, and a variety of solutions exist, although they haven't investigated them in this work. Someone else said that he liked the idea, but there is a potential problem: since the same servers generating the logs are analyzing them, won't "hot" services—those services that generate more logs with valuable data—have fewer processing cycles available for analysis? Dionysios acknowledged that this is indeed a limitation of the

iMR approach and perhaps other load-shedding mechanisms will be useful for dealing with it.

Joint ATC and WebApps Invited Talk

Helping Humanity with Phones and Clouds

Matthew Faulkner, graduate student in Computer Science at Caltech, and Michael Olson, graduate student in Computer Science at Caltech

Summarized by Timothy Merrifield (tmerr14@uic.edu)

Matthew Faulkner began the talk by proposing the possibility of using smartphones to more accurately and quickly respond to rapidly changing (and perhaps catastrophic) situations. The application of smartphones to earthquakes, for example, would allow their accelerometer to act as a sensor for detecting shaking and movement. His ultimate vision is a crowd-sourced sensor network that could accurately detect earthquakes and perhaps guide rescue workers to the areas that experienced the most serious damage. With such a sensor network deployed, Faulkner imagined other applications, such as an early warning system, that could alert train conductors to stop or perhaps an alert to halt elevators in a building and let passengers out at the next floor. Faulkner went on to describe the challenges with such a system, including how to reduce the amount of accelerometer data that needs to be transmitted and how to avoid the possibility of false alarms.

After Faulkner concluded, Michael Olson spoke about his work on providing remote health care via smartphones to those in more rural communities. He explained that some basic health care devices (thermometer, stethoscope, ultrasound, etc.) can be built to interface with smartphones. That sensor data could then be sent to a health care professional on the other side of the world, who could diagnose and recommend treatment for common conditions. He played the sound of an irregular heartbeat and explained that this is an example of a condition that could easily be diagnosed remotely.

The Q&A session began with a questioner wondering why smartphones were used as earthquake detectors instead of laptops, which are more stationary and would produce less noisy data. Faulkner replied that laptops only have a very rudimentary accelerometer that detects free falls for hard drives and that the sophisticated accelerometer in most smartphones is what makes them most appealing. The next questioner wondered how the earthquake data could be provided to emergency officials if the Internet were to become unavailable. Faulkner agreed that this was a concern and proposed that packaging up map data such that it could be delivered physically to emergency responders would have to be a part of any real-world system. Another questioner asked

about the potential for such a system to anticipate future earthquakes perhaps hours or days in advance. Faulkner said that this problem is very difficult and that even if such signals exist, the actual earthquake might be months or years down the road.

Joint ATC and WebApps Poster Session

Partially summarized by Dan Levin (dlevin@net.t-labs.tu-berlin.de)

The poster session at this year's ATC was combined with a social hour, where roughly 25 different research topics were presented in poster form. Among these were included submissions from both the short and full paper sessions. Of these, a few summarized discussions with poster presenters are given here. Sadly, none of the fine beer and hors d'oeuvres could be included in this summary.

Evaluating the Effectiveness of Model-Based Power Characterization

John C. McCullough and Yuvraj Agarwal, University of California, San Diego; Jaideep Chandrashekar, Intel Labs, Berkeley; Sathyanarayan Kuppuswamy, Alex C. Snoeren, and Rajesh K. Gupta, University of California, San Diego

Many computer system power consumption models in use today assert that idle power consumption contributes a large, fixed portion of the overall consumption, with a dynamic, linearly increasing component as utilization increases. This work argues that such models no longer hold true today: both the fixed idle and the dynamic consumption may exhibit far different behavior in today's increasingly complex, multicore hardware. This work presents an analysis of fine-grained component-level power consumption measurement (disks, memory, integrated graphics, LAN, etc.), leading to improvements in device power characterization.

OFRewind: Enabling Record and Replay Troubleshooting for Networks

Andreas Wundsam and Dan Levin, Deutsche Telekom Laboratories/TU Berlin; Srinu Seetharaman, Deutsche Telekom Inc. R&D Lab USA; Anja Feldmann, Deutsche Telekom Laboratories/TU Berlin

Network troubleshooting is challenging for many reasons: high traffic volumes hamper pre-emptive recording for later analysis, pervasive black box devices prevent instrumentation, and distributed protocols may exhibit hard-to-reproduce faults. This work introduces the primitive of replay debugging to networks, implemented as OFRewind, a software tool for recording and replaying traffic with the help of OpenFlow. OFRewind allows network operators to centrally orchestrate the scalable recording of arbitrary flow-level

granularity traffic in an OpenFlow network, maintaining a complete ordering of all flows. This traffic can be recorded in always-on fashion, and replayed in situ or in a lab to reproduce many different types of errors and localize problems.

BenchLab: An Open Testbed for Realistic Benchmarking of Web Applications

Emmanuel Cecchet, Veena Udayabhanu, Timothy Wood, and Prashant Shenoy, University of Massachusetts Amherst

Presented as a demo at the poster session, this project aims toward improving the reality of Web application benchmarking by leveraging real Web browsers for workload generation. The process begins with trace recordings of HTTP connection and activity logs at an operational Web server. From these recordings, browsers can be orchestrated from any point in the Internet to reproduce previously recorded workload towards the desired Web application under test.

Making GPUs First Class Citizens

Vishakha Gupta and Karsten Schwan, Georgia Institute of Technology

Today's computing system architectures are growing ever more heterogeneous. Multicore CPUs, multi-socket systems, and GPUs may all be leveraged to increase system performance. But in such heterogeneous systems, the processing asymmetry must be accommodated and its performance impact must be minimized. This work proposes using VM hypervisors to accommodate different underlying hardware. Using such an approach, this work proposes to investigate the gains and losses in application performance.

Toward Online Testing of Federated and Heterogeneous Distributed Systems

Marco Canini, Vojin Jovanović, Daniele Venzano, Boris Spasojević, Olivier Cramer, and Dejan Kostić, EPFL

Improving the reliability of distributed systems and protocols, for example BGP interdomain routing, is difficult in practice. It is well known that convergence of such systems is not predictable, and a single fault can transform a stable running system into an unstable one. This work proposes to proactively monitor and test the behavior of such systems and recognize deviations from expected behavior. Additionally, exploration of system behavior is proposed by obtaining snapshots of stable live systems, replicating to offline systems, and subjecting these to differing inputs to probe for faults.

Joint ATC and WebApps Plenary Session

Dead Media: What the Obsolete, Unsuccessful, Experimental, and Avant-Garde Can Teach Us About the Future of Media

Finn Brunton, Postdoctoral Researcher at NYU

Summarized by Muhammad Shahzad (shahzadm@msu.edu)

Finn Brunton delivered a very interesting and thought-provoking talk on how different and diverse forms of media emerged in history and how some forms flourished while others either did not get adopted or got dissolved into other forms of media, where “media” stands for devices and platforms to store, transmit, and represent information.

Finn showed through examples from history that media hardly ever “dies”—it just gets transformed and exists in other forms. For example, the pilgrims of late medieval times used pilgrim mirrors. These were convex polished badges made of lead and tin. People would hold them to reflect the image of the holy relics in them and then considered them sacred and kept them because they once held the image of those holy relics. These were like cameras with no film. Finn also gave examples of television with no images (1960s stroboscopic light displays) and 18th-century augmented reality.

Non-recoverable death of media is rare. They continue to be maintained in practice. For example, people still know how to do calligraphy; QWERTY keyboards keep alive manual typewriters. Part of the reason why there are so many dead media platforms is that stuff gets independently invented over and over again with slight variations. For example, Graham Bell and Elisha Gray applied for the telephone patent on the same day. However, looking at the brighter side, these failed media provide a plethora of alternate methods of recording and representing, alternate ideas of what these things are for, alternate futures that could have been. Whenever we struggle with breaking out of the present way of thinking, they are here to suggest completely different means and approaches.

One should look into the future, where one's work is ultimately headed. If the people are not yet ready for that, then track back until the point is reached where the work is understandable and therefore useful. In media, prior forms are needed because they act like handrails. Think of the “office” and “page” metaphors in GUI operating systems. Also consider where the work is going to be in a decade, a century, even a millennium. A great example is the work by Abu Ali Hassan Ibn al-Haytham in optics: almost 1000 years passed but his book *Kitab al-Manadhira* is still enormously influential. Do we have a sense that what we are working at here will be as amenable to preservation and transformation?

The ultimate takeaway of the talk was: Dead media are not really dead; they're just sleeping. So we should try to make what we create as easy as possible for the future to reawaken.

Measurement and Performance

Summarized by Vishakha Gupta (vishakha@cc.gatech.edu)

Exception-Less System Calls for Event-Driven Servers

Livio Soares and Michael Stumm, University of Toronto

Livio Soares presented work on exception-less system calls as techniques for implementing event-driven servers instead of multi-threaded servers. Event-driven systems register an event after each stage, which gets recognized by the OS to avoid idle times since the applications can go ahead and do other work. He pointed out, however, that the benefits currently are not possible for all system calls through the UNIX event primitives, as they only handle ones that use file descriptors.

Livio briefly summarized their previous work, called FlexSC (OSDI '10), which enabled exception-less system calls in the Linux kernel. He then moved to describe "libflexsc," which is an asynchronous system call and notification library suitable for building event-driven applications. Applications modified to use the libflexsc library could benefit from the exception-less system call implementation supported by the kernel. The implementation has been optimized with multicores in mind. The use of this library with FlexSC brings asynchronous event-driven execution on general-purpose processors with a non-intrusive kernel implementation for improved processor efficiency. It facilitates multiprocessor execution where libflexsc provides an event loop for the program, which must register system call requests along with callback functions. The main event loop on libflexsc invokes the corresponding program-provided callback when the system call has completed. Performance evaluation with memcached and nginx saw large benefits, since libflexsc speeds up the large volume of user/kernel communication that happens currently for tracking progress of I/O.

Abel Gordon (IBM Research) asked if they had evaluated performance when kernel threads were run on different sockets. Livio replied that they hadn't performed actual experiments but he would try to keep syscall threads on the same locality domain because of shared syscall entries that will be communicating back and forth. With more sockets the execution might have to be split differently. Does there need to be some kind of affinity with respect to interrupts? Livio said interrupts were being sent to the cores where syscall threads were actively being processed. Their system tries to increase or decrease the number of syscall cores based on application behavior. They change the APIC controller to go

to those cores, so there is some mapping between cores that issue interrupts and those that handle them. Peter Desnoyers (Northeastern) asked if they saw any parallel with the interrupt coalescing work in the form of their deferred work queue. Livio said this work derived from the soft timers work, but his work is more from the application down. Michael Dexter asked about the security risks of using libflexsc. Livio replied that the fundamental property they relied on was MMU; if the OS set up the page tables correctly, only the corresponding process should get access. Besides, there were easier ways to compromise the OS. Someone asked about the complexity of using events in large programs. Livio said he did not have much experience with that, but that they would essentially look like large distributed systems.

Resizable, Scalable, Concurrent Hash Tables via Relativistic Programming

Josh Triplett, Portland State University; Paul E. McKenney, IBM Linux Technology Center; Jonathan Walpole, Portland State University

The way synchronization is done today essentially comes down to waiting, which leads to concurrent programs wasting a lot of time. The various alternatives—finer-grained locking, reader/writer locks, transactional memory—all come down to either some form of wait, resource wastage, or expensive sync instructions on multicores.

Josh Triplett introduced relativistic programming and the notion of utilizing the fuzzy-shared communication via cache coherence protocols. Relativistic programming supports no global order for noncausally related events—readers don't wait for readers or writers and writers do all the waiting when necessary. Therefore, reads become linearly scalable. So that writers don't disrupt readers, data structures have to be consistent after every write. Writers wait on readers to make sure they order their writes. Delimited readers know when they start and end, and they publish pointers to ensure ordering between initialization and publication. The updates can wait for existing readers. The moment writers publish a pointer, readers can immediately see it. For deleting, they garbage-collect nodes to delete after existing readers are gone. Josh then showed how these ideas work through example, resizing primitives for lockless hash tables. He demonstrated the challenges in expanding and shrinking while maintaining consistency. He then showed an evaluation of this implementation and concluded with their future ideas on being able to construct arbitrary data structures and provide a general method for algorithm construction.

Raju Rangaswami (Florida International) asked about how writers waited for readers. Josh said writers use an operation that allows them to check if there are concurrent readers running, since readers make it known, using read-copy-

update. Someone asked how the system tracks the number of readers. Josh said it worked somewhat like semaphores where the system knew the number of readers. To a question on whether this worked well only with read-mostly/read-heavy data structures, Josh said that was the starting focus, since a large number of applications fell into that category. But they are definitely not limited to that. In particular, read-mostly can vary with how much overhead you have. For some of their algorithms, even 1:1 can get a win while at other points there isn't a significant negative. What happens as you push more readers through? You get fewer readers in less time. If the main focus is on write performance, then they might get affected and it depends on the workload. Writers do become a little more expensive with this approach, but readers become a lot faster.

Evaluating the Effectiveness of Model-Based Power Characterization

John C. McCullough and Yuvraj Agarwal, University of California, San Diego; Jaideep Chandrashekar, Intel Labs, Berkeley; Sathyanarayan Kuppusswamy, Alex C. Snoeren, and Rajesh K. Gupta, University of California, San Diego

At a time when power models are seeing greatly increased use, Yuvraj Agarwal took a very different stand and talked about how they are inaccurate and why we need alternatives. Although the need for good power measurement solutions remains, modern systems with a large dynamic range of power are harder to characterize than their prior generation counterparts, which had a much smaller dynamic range but a large base power. Among existing methods, one is to directly measure power, which could lead to a lot of wiring, or one could indirectly model to reduce hardware complexity. But how good are the many power models that have been proposed over time? Yuvraj and his group used an Intel Calpella platform with 50 sense resistors and high-precision power measurement infrastructure to show how the power models returned erroneous results.

He used a large set of common benchmarks like SPEC CPU 2006, PARSEC, Linux build, and so on to demonstrate how the power models did progressively worse on newer platforms, especially multicores, when compared to their accurate power measurement installation on the Calpella platform. He pointed out that the overall system power was still within tolerable limits but the error grew worse when characterizing individual components. He concluded that: (1) modern systems have lots of hidden state for multiple reasons (sometimes because manufacturers don't want to reveal them); (2) increasing system complexity with some interactions is hard to capture—for example, HDD power modeling error is half that of SSD; and (3) the problem becomes worse

if you add hardware variability to the mix—even identical parts are not necessarily identical. He ended by emphasizing the need for low-cost instrumentation solutions for accurate power characterization, which will be quite essential in future systems.

Assar Westerlund (Permabit) asked whether they had presented their conclusions to the vendors, and Yuvraj said that they were in the process of providing vendors like Intel with all of these numbers, so there was a good cost/benefit argument. Could the measurements be made for a particular system and then be used as workload for others? Yuvraj said that instrumentation was needed on all platforms because data on one need not match the other, even with same model numbers. Amit (MSR) repeated that there were cases where power modeling was needed, e.g., when you had to measure power consumption by two applications separately. Yuvraj agreed but said that was an orthogonal problem. What their research pointed out was that the base system power predicted by some power models itself was erroneous, leading to propagation of these errors when applying other methods to distribute it. Amit argued that some of those techniques are more sophisticated than the ones evaluated by the authors. Yuvraj asked how any power model that has no idea of hardware variability can work. It depends on the workloads and whether it wants something for components or for the total system.

Storage Performance and Migration

Summarized by Mark Spear (mspear@cs.ubc.ca)

Victim Disk First: An Asymmetric Cache to Boost the Performance of Disk Arrays under Faulty Conditions

Shenggang Wan, Qiang Cao, Jianzhong Huang, Siyi Li, Xin Li, Shenghui Zhan, Li Yu, and Changsheng Xie, Huazhong University of Science and Technology; Xubin He, Virginia Commonwealth University

Shenggang Wan presented a new approach to cache management for storage under failure conditions. One of the weaknesses of existing parity-based RAID systems is the dramatically different cost of serving requests under normal and failure scenarios. One disk read is required to deliver a block under normal conditions, but many reads (equal to the number of disks in the parity set minus one) are required to reconstruct the lost data under failure conditions. He proposes that we factor this differential cost into caching metrics. The Requests Generation Ratio (RGR) is the ratio of disk blocks requested to buffer cache blocks requested, and is a better alternative than using naive miss ratios to describe cache effectiveness.

When the disk array fails, Victim Disk First (VDF) caches the blocks from the faulty disks with higher priority. Existing cache policies such as LRU and LFU can be modified to utilize the RGR, weighting blocks in the cache by the true miss penalty instead of assuming a constant replacement cost. This reduces the number of cache misses that will be directed to faulty disks, and therefore to the many surviving disks. Wan's group performed a case study for read requests on a RAID-5 system with a failure. The simulation showed improvements on both reconstruction duration and throughput. A prototype showed results similar to the simulation. Future work involves extending VDF to more general cache algorithms and other RAID levels.

Irfan Ahmad (VMware) asked if this scheme could be extended to handle heterogeneous devices. Instead of having faulty disks, what if you have some fast disks and some really slow disks? Wan said that with some modifications to the miss penalty this technique might be able to offer a more general performance optimization than only for faulty RAID situations, but it would need to be the subject of future work. Dutch Meyer (University of British Columbia) asked if they had considered changing the expiration time for writing back data in cache based on their metrics. Wan deferred to the paper for discussion of write operations. Session chair Ajay Gulati (VMware) asked if their techniques work with other modern cache policies such as ARC, LIRS, and MQ. Wan mentioned that he believed they would, and this would also be the subject of future work.

The Design and Evolution of Live Storage Migration in VMware ESX

Ali Mashtizadeh, Emr  Celebi, Tal Garfinkel, and Min Cai, VMware, Inc.

Ali Mashtizadeh talked about the development of storage migration technologies in VMware ESX over the years. Moving a virtual machine (VM) between two hosts with minimal downtime is important for many reasons. Disk state, on the order of terabytes, is the bulk of the data migrated. Mashtizadeh told about a customer who was upgrading a storage array and had to schedule a month's worth of migrations. The goals of live storage migration include predictable and minimal migration times, low performance impact on the guest, atomicity (for replication and crash consistency), and convergence (barring failures, migration will complete).

The evolution of storage migration begins with a simple approach: making the VMDK base disk image read-only, sending writes to the snapshot, migrating the VMDK, repairing to the destination volume, and migrating the snapshot. Since this approach has a number of disadvantages, including downtime, long migration times, and requiring disk space up to 3x the VM size, they tried something different: A dirty block tracking (DBT) filter marks dirty blocks. The migra-

tion involves copying changes and repeating until the set of changes is small enough, then stopping the VM, copying the final changes, and cutting over. This still had limitations such as questions about convergence (e.g., will it complete under workloads where blocks are dirtied very quickly?). The most recent insight was that storage is not memory: interposing on all writes is practical. I/O mirroring works by synchronously mirroring all writes and making a single pass to transfer the bulk data. This approach has effectively no downtime, low performance penalty, and atomicity. Throttling the source can be used when I/O mirroring to a slow destination. Future work involves leveraging workload analysis (heat maps) for I/O mirroring, sequential write pattern detection, lazy mirroring, and WAN migrations..

Marcos Aguilera (Microsoft Research) asked about the performance hit of using synchronous mirroring across data-centers. Mashtizadeh responded that their WAN vision is asynchronous mirroring with lots of buffering, and throttling if the workload is too fast. Another attendee asked about the race conditions in mirroring. Mashtizadeh had a prepared backup slide for this question, showing there is a locking mechanism. The region currently being copied defers I/Os, but is small enough that it doesn't have bad latency effects in the VM. Assar Westerlund (Permabit) wanted to know about the overhead for the guest, and Mashtizadeh referred to the paper for instantaneous numbers, as well as a graph of the integration of performance penalty over time. Session chair Ajay Gulati (VMware) asked about modeling the cost of vMotion (the migration technology). The current model assumes a constant number of outstanding I/Os and a constant amount of data to transfer. Using heat data, they use a linear scale to weight data across the LBAs of the disk (data at the end of the disk will only be issued at the end of the migration, while data at the beginning will be issued all the time).

Online Migration for Geo-distributed Storage Systems

Nguyen Tran, Marcos K. Aguilera, and Mahesh Balakrishnan, Microsoft Research Silicon Valley

Nguyen Tran shared the observations that many Internet applications are geo-distributed across multiple sites and that the best site to store a user's data may change as a result of user relocation and other events. Current systems for migration include locking and logging: Locking involves blocking writes during the migration, which may take a long time. Logging records writes at the old site during migration, followed by a locking-style migration for the log.

Tran proposed a new approach called distributed data overlays. The intuition behind this approach is read first at the new site, and redirect if the data is not there. This technique allows data to be accessible at all times, and the migration benefits (e.g., locality to user) are realized quickly. When

an overlay is created, a layer is added to the top of the stack. Reads go from the top of the stack down, until valid data is reached. Writes are written only to the top layer. Migration between overlays involves copying data from the source to destination, while reads and writes are going on concurrently. Tran's group developed Nomad, an object storage system using overlays. The distributed data overlays performed better than traditional lock-and-log approaches. Write latencies are instantly lowered, read latencies get better gradually throughout the migration, and there is no point at which the data is inaccessible.

They also studied policies to drive migration for Hotmail using a trace of 50,000 random Hotmail users. If a user travels and their closest site changes, when should we trigger migration of her data? Two simple policies are a "count policy," which is a threshold on the number of accesses at the new site, and a "time policy," which is a threshold on the amount of time at the new site. They found the count policy is better, and the intuition is that you want to migrate data for users making more remote accesses.

Ajay Gulati (VMware) asked whether the latencies are multiplied if reads are being forwarded from overlay to overlay. Tran said that writes should only go to the top overlay, but reads will be forwarded if there is a cache miss. Parallel reads (to multiple overlay layers) could be issued, but it is a tradeoff for bandwidth. Etienne Le Sueur (NICTA and UNSW) asked about the movement of caches in the migration example presented. Tran clarified that both reads and writes could be in the cache, and thus had to be moved accordingly. Emré Celebi (VMware) asked about failure cases pre-migration, during migration, and post-migration. Tran said that a drawback of this approach on its own is the possibility of inaccessible data. However, failures can be handled by replication working together with overlays. Parthasarathy Ranganathan (HP Labs) asked about a user moving from point A to B and back to A. Tran pointed out that upon moving back, a new overlay would be created at A (resulting in two overlays at the same location), and eventually the stacks would be merged.

Joint WebApps and ATC Invited Talk

Summarized by Nadav Amit (namit@cs.technion.ac.il)

Software G Forces: The Effects of Acceleration

Kent Beck, Facebook, Inc.

Kent Beck's keynote was undoubtedly the liveliest in USE-NIX ATC '11. Kent started by reviewing the software development situation today and in the recent past. The software development cycle in the 1990s was mostly yearly, and only a few specific software products, such as bond traders' software, had a deployment cycle that was shorter than a quarter. Today the deployment cycle is considerably shorter, mainly

quarterly, and crazy fringe things start to happen as some software products are deployed on a daily basis. Kent believes this trend will continue, and in the 2030s most of the software development cycles will be shorter than a month. Kent provoked the audience by asking, "What happens if 90% of the software will be deployed on an hourly basis in 20 years?"

Kent then presented the various changes in the software development process that are required for shortening the software deployment interval. Deploying software quarterly instead of yearly requires automated acceptance tests, a continuous software design process, continuous integration, and a subscription-based business model. "What if your design is wrong?" people from the audience asked. "There is certainty that your design is wrong. You will have to adapt it," Kent replied. Irfan Ahmad from VMware expressed his own doubts that the subscription model is relevant in the enterprise software world, since customers do not like to deploy software frequently. Kent replied with a story that showed that sometimes enterprise customers already deploy software more frequently than they think, as patches are released during the integration process. Irfan insisted that the integration process can take up to six months, to which Kent replied that reducing the defect rate will shorten the integration process.

Kent continued by describing the changes required when moving from a quarterly deployment cycle to a monthly one, changes that include developer testing, stand-up meetings, cards on a wall, pay-per-use business-model, and elimination of design documents. The latter raised doubts in the audience, who wondered why social media cannot be used to share knowledge. In response, someone from the audience shouted, "Wiki is where ideas go to die." Moving to weekly deployment cycles requires further changes, such as "two-way data migration" and saving data during data migration, both in the old format and the new format. Moving to daily cycles requires the elimination of staging, an operations team, and stand-up meetings (those that were introduced going to a monthly deployment cycle). Kent concluded by saying he loves daily deployment.

Short Papers

Summarized by Nadav Amit (namit@cs.technion.ac.il)

Slow Down or Sleep, That Is the Question

Etienne Le Sueur and Gernot Heiser, NICTA and The University of New South Wales

Etienne kicked off the short papers session, presenting his work on the tradeoff between dynamic voltage and frequency scaling (DVFS) and sleep states on performance and power-consumption using server, desktop, and embedded processors. DVFS is becoming less effective, due to increased

static power, smaller voltage ranges, and better memory performance, while idle states are becoming more effective. Etienne evaluated real-life “bursty” workloads such as Apache and MPlayer, in contrast to the SPEC CPU workloads which are usually used for evaluation. The results reveal that DVFS is indeed becoming less effective with better C-states. However, lightly loaded systems lose no throughput or latency with reduced frequency and deep C-states, and, due to the CPU not entering the deepest possible “package” C-state, DVFS can still improve energy efficiency. A further interesting result of the research was that desktop-class CPUs (such as the Core i7) deliver better energy per request than embedded-class CPUs (such as the Atom and OMAP).

Low Cost Working Set Size Tracking

Weiming Zhao, Michigan Technological University; Xinxin Jin, Peking University; Zhenlin Wang, Michigan Technological University; Xiaolin Wang, Yingwei Luo, and Xiaoming Li, Peking University

Page-level miss ratio curve (MRC) has various applications, such as working set size estimation and memory resource balancing, yet the overhead of its calculation is high. Weiming presented his innovative solution for reducing overhead by disabling tracking when memory demand is predicted to be stable, and turning it back on when hardware performance counters input indicates a phase-change occurs. Combining this approach with the existing approaches of AVL-tree-based LRU structure and dynamic hot set sizing reduces the overhead to only 2%. Balancing virtual machines memory using these techniques was shown by Weiming to result in a 22% speedup.

FVD: A High-Performance Virtual Machine Image Format for Cloud

Chunqiang Tang, IBM T.J. Watson Research Center

Chunqiang discussed Fast Virtual Disk (FVD), a new virtual machine image format developed by IBM for use in cloud environments that delivers better performance than existing formats. FVD improves performance by using a bitmap for copy-on-write, copy-on-read, and adaptive prefetching, eliminating the need for expensive lookups that virtual disks such as QCOW2 require, and avoiding reading of unmodified data from NAS. FVD was implemented as a block device driver for QEMU, and evaluations showed that its performance is almost as good as raw disk (249% improvement over QCOW2). Chunqiang also presented how the optional use of lookup-table by FVD enables support of a compact image with small metadata size, and that FVD is able to perform efficient snapshots using reference counts.

Okeanos: Wasteless Journaling for Fast and Reliable Multistream Storage

Andromachi Hatzieleftheriou and Stergios V. Anastasiadis, University of Ioannina

Andromachi presented solutions for the bandwidth waste and high disk latencies caused by synchronous subpage writes, solutions that do not sacrifice reliability. The first solution, “wasteless journaling,” is to use journaling for subpage writes, occasionally moving data blocks from memory to their final location. The second, “selective journaling,” is similar to “wasteless journaling,” yet subpage writes are journaled only if they are smaller than a certain threshold, in order to save bandwidth that might be wasted due to duplicate data writes. Experiments showed substantial improvements in write latency, transaction throughput, and storage bandwidth requirements over the ext3 file system. Andromachi noted, in response to Josh Triplett’s question, that further work is required in order to apply these solutions to ext4.

Toward Online Testing of Federated and Heterogeneous Distributed Systems

Marco Canini, Vojin Jovanović, Daniele Venzano, Boris Spasojević, Olivier Crameri, and Dejan Kostić, EPFL

Marco began by joking that his talk would take 40 minutes. Yet, in a mere eight minutes, he managed to clearly present his work on proactive identification of potential faults. As an example, Marco spoke of origin misconfiguration in Pakistan ISPs’ routers that disabled access to YouTube for two hours. His system, named DiCE, continuously and automatically explores the routing system behavior by applying concolic testing to the distributed system nodes while dealing with the exponential number of possible code paths. The solution was applied to the BGP implementation of BIRD, was able to detect origin misconfiguration, and showed negligible impact on live system performance. The main overhead was memory usage (37%), yet they did not optimize the system in respect to memory consumption.

CDE: Using System Call Interposition to Automatically Create Portable Software Packages

Philip J. Guo and Dawson Engler, Stanford University

Philip said that eight minutes would not be enough for presenting the entire paper; he therefore chose to focus on a use case of the system. Trying to install the regular binary package of ML Demos, a GUI for machine learning, will fail if not executed on the Linux distribution and version it is intended for. The CDE system is built to deal with such situations, allowing it to run programs across distributions and versions on any modern x86 computer, by using ptrace to redirect file paths that the target program requests into the package.

Philip's experiments showed that packages can be executed successfully on 2006 Linux distributions, with up to 30% overhead. Josh Triplett wondered whether personal data can be blacklisted, so it would not be included in the package, and Philip said that that customization feature is included in CDE.

Vsys: A Programmable sudo

Sapan Bhatia, Princeton University; Giovanni Di Stasi, University of Napoli; Thom Haddow, Imperial College London; Andy Bavier, Princeton University; Steve Muir, Juniper Networks; Larry Peterson, Princeton University

Sapan presented Vsys, a free and open source tool which is actively used in PlanetLab for restricting access to privileged operations. This tool can be considered a flexible sudo, which is intended to deal with demands that are not conveniently satisfied by the default security model of UNIX. Vsys can be used for performing simple tasks such as opening a raw socket, or complex ones such as creating a private overlay network. Vsys uses FIFO pipes or a UNIX domain socket to allow users to communicate with the extension. As an example, Sapan presented sliceip, an extension that creates service-specific route tables, and pointed out that the vast majority of the extensions supported PhD dissertations and papers in the most prestigious conferences. Vsys was in fact preferred over Linux containers, as the latter presented many support and tool problems and included many bugs.

Internet-scale Visualization and Detection of Performance Events

Jeffrey Pang, Subhabrata Sen, Oliver Spatscheck, and Shobha Venkataraman, AT&T Labs—Research

Monitoring the performance of server farms is the motivation behind BirdsEye, a tool that visualizes the latency of a server to the Internet, presented by Subhabrata Sen. Standard rule-based detection techniques that identify problems cannot, by definition, deal with the “unknown unknowns.” Visualization can significantly aid the rapid discovery of such unknown patterns. The key idea of BirdsEye is to model the latency as a decision tree over the IP address space hierarchy (where each node corresponds to an IP prefix) and cluster together IP addresses with similar performance characteristics. BirdsEye visualizes these adaptive decision trees, distinguishing parts of the Internet that have good performance from those with poor performance. Experiments show that the system is able to identify higher latencies of wireless networks and the abnormal latency of a certain ISP in Utah. Automatically figuring the cause of the anomaly is more difficult and Subhabrata noted that research on this is currently being undertaken.

Polygraph: System for Dynamic Reduction of False Alerts in Large-Scale IT Service Delivery Environments

Sangkyum Kim, University of Illinois at Urbana-Champaign; Winnie Cheng, Shang Guo, Laura Luan, and Daniela Rosu, IBM Research; Abhijit Bose, Google

Sangkyum's talk focused on a challenge that is presented to IT service delivery systems whose monitoring results in huge amount of false alerts. Polygraph, a system that Sangkyum and his colleagues developed, reduces false alerts by using an active-learning approach. Alert policy adjustment schemes based on host and time dimensions enabled the reduction of false alerts significantly.

Storage Deduplication

Summarized by Mark Spear (mspear@cs.ubc.ca)

Building a High-performance Deduplication System

Fanglu Guo and Petros Efstathopoulos, Symantec Research Labs

✎ *Awarded Best Paper!*

Petros Efstathopoulos pointed out that while deduplication systems are maturing, scalability is still an issue. He posits that improving single-node performance is critical, even for multi-node systems. Aiming for perfect deduplication imposes various limitations, so his team made the tradeoff of deduplication efficiency for scalability by using progressive sampled indexing. The sampling rate for segment fingerprints is a function of used storage and available RAM. Another innovation in their deduplication system is the use of Grouped Mark-and-Sweep (GMS), whereby unused segments are garbage-collected in “backup groups.” The workload of this approach is a function of the amount of change, rather than a function of the system capacity, and is therefore a more scalable approach.

They evaluated their system with two data sets: a synthetic data set with no duplicate content, and a data set with multiple versions of a VM image. Their system's raw disk throughput was about 1 GB/s. While a normal file server slowed considerably as concurrency increased, their backup system with no deduplication delivered a constant 1 GB/s by performing write-friendly optimizations. Their cold-cache deduplication raised backup throughput to 6.6 GB/s, and their warm cache backup speed was approximately 11.5 GB/s (with a slight dip at 256 concurrent backups). Even when the system is at 95% capacity, the throughput is approximately the same. The deduplication efficiency was approximately 96–97%.

Assar Westerlund (Permabit) asked about the difference between this system's consistency requirements and how

file systems need to keep track of pointers to data blocks. Efstathopoulos pointed out that after a crash, file systems need either a recovery phase or a log, and his group wanted to avoid a painful recovery process for rebuilding references. Austin Clements (MIT) asked if the fact that 3–4% of duplicates are not detected meant it would cost extra space for a VM that is being backed up over and over again. Efstathopoulos said it would, but this is a reasonable tradeoff for the scalability that the system provides. Ajay Gulati (VMware) asked about comparisons with existing approaches (e.g., Data Domain) in terms of performance and cost. The response was that this is a prototype system. The paper has a survey of what is out there, but it is costly to acquire professional products, and there is no standard for comparison. There is high-level information in the paper, but no performance numbers.

Geoff Kuenning (Harvey Mudd) inquired how the synthetic workload generator works. Efstathopoulos replied that 4K blocks are generated by hashing sequential numbers with the hope that it doesn't generate collisions. Assar Westerlund asked how sparse the VM images used were. They were sparse, preallocated images. More data was added to successive images, and the same sampling rate was used in all workloads. Philip Shilane (EMC) asked about the mark-and-sweep algorithm: in steady state as files are added and deleted, how much space is wasted for dead segments? Efstathopoulos observed partial liveness/deadness fragmentation in containers and said that it is something that they need to deal with, but it is future work.

SiLo: A Similarity-Locality based Near-Exact Deduplication Scheme with Low RAM Overhead and High Throughput

Wen Xia, Huazhong University of Science and Technology; Hong Jiang, University of Nebraska–Lincoln; Dan Feng, Huazhong University of Science and Technology; Yu Hua, Huazhong University of Science and Technology and University of Nebraska–Lincoln

Wen Xia presented SiLo, a deduplication system that exploits both similarity and locality. There are existing approaches to deduplication that exploit either only locality (which can have poor throughput for some data sets) or only similarity (which can have poor deduplication efficiency). Their system takes a backup stream and divides it into similarly sized segments, splitting large files and grouping correlated small files, which has a positive impact on similarity detection. Contiguous segments are grouped into blocks to preserve locality layout on disk. The locality algorithm can subsequently help detect duplicates that the similarity algorithm missed.

Realizing that the deduplication server is the performance bottleneck, they made that the focus of their paper. Their

similarity and locality hash tables use significantly less RAM than some competing systems. They compared their system to ChunkStash (locality-based) and Extreme Binning (similarity-based) under four workloads. SiLo achieved better deduplication efficiency than Extreme Binning and has much lower RAM usage than ChunkStash. SiLo beat both other systems in terms of deduplication throughput, besting ChunkStash by a factor of 3 and Extreme Binning by a factor of 1.5.

Assar Westerlund (Permabit) asked how the segment size and block size should be tuned. Xia replied that using segment sizes of 2 MB or 4 MB and block sizes of about 200 MB worked well in their evaluation, but you could envision choosing different sizes based on RAM availability for indexing purposes. Westerlund also pointed out that the tradeoffs made by this system seem good, but asked if there are data sets for which this doesn't work. Xia said small-scale data sets and low locality wouldn't be workloads suitable for SiLo.

Debugging and Diagnosis

Summarized by Etienne Le Sueur (elesueur@cse.unsw.edu.au)

G²: A Graph Processing System for Diagnosing Distributed Systems

Zhenyu Guo, Microsoft Research Asia; Dong Zhou, Tsinghua University; Haoxiang Lin and Mao Yang, Microsoft Research Asia; Fan Long, Tsinghua University; Chaoqiang Deng, Harbin Institute of Technology; Changshu Liu and Lidong Zhou, Microsoft Research Asia

Haoxiang Lin began by noting that bugs are often easily noticed: a system malfunctions and a user complains. Although the final fixes may seem simple, finding the root causes of such bugs, especially in highly distributed systems, is a complex and painful process. Often, distributed systems are executing on many machines, detailed logs are recorded on each machine, and correlations between logs are not well preserved. This makes using those logs to diagnose problems difficult. G², a graph-based system for diagnosing bugs, uses an execution flow graph model to trace the root causes of malfunctions in a distributed system.

Diagnostic practices are abstracted into two primitives: “slicing” and “hierarchical aggregation.” Slicing can be used to find and filter an execution graph into portions in which important correlated events are present. Hierarchical aggregation helps to summarize whole or part of an execution graph into a high-level view which facilitates better understanding of how the target system works.

Slicing and hierarchical aggregation are implemented as traversing the execution graphs. G² builds a distributed graph engine and applies several special optimizations such as

batched asynchronous graph traversal, partition-level interface, caching, and prefetching, which makes traversal on huge graphs very efficient. Haoxiang presented an evaluation using the SCOPE/Dryad distributed programming system.

Someone asked what types of bugs G² worked well with and what types it did not. Haoxiang responded that G² is useful for bugs having complex internal interactions, which are difficult for “simple” tools to detect. Using G², they can perform several runs of slicing and aggregation to find correlations and, hence, the root causes of such complicated bugs.

Context-based Online Configuration-Error Detection

Ding Yuan, University of Illinois at Urbana-Champaign and University of California, San Diego; Yinglian Xie and Rina Panigrahy, Microsoft Research Silicon Valley; Junfeng Yang, Columbia University; Chad Verbowski and Arunvijay Kumar, Microsoft Corporation

Ding Yuan presented a paper which deals with erroneous configuration errors on Windows systems. Configuration errors are a major root cause of system failures; previous work shows that 25–50% of system outages are caused by configuration errors. Early detection of such configuration errors is important to minimize such outages. The goal of this work is to automatically detect configuration errors and report these to system administrators, who can then proactively fix them.

The basic premise is to monitor changes to the Windows registry. However, this key-value store of configuration parameters is huge, and monitoring and reporting all changes would place an unnecessary burden on users. Therefore, only those configuration parameters that are subsequently read are tested for erroneous values. The proposed system (named CODE) monitors the sequence of events that led to a configuration change. If future event sequences deviate from the known good sequences, then registry changes resulting from such deviated event sequences are reported.

The evaluation shows that the system has very little CPU overhead and generates few false positives. They show that 41 out of 42 production-run error cases reported by real users can be detected using CODE. In addition, CODE can detect 97% of the randomly injected errors. Although there is a 7% memory overhead associated with the system, a single CODE instance can monitor many other machines.

The first questioner pointed out that the paper’s results show that CODE does not achieve full coverage. Ding replied that the missing case is because CODE had learned a rule that was too long. During the detection phase, the learned rule was shorter than what was observed, hence the error was not detected. Can this be addressed? Ding replied that they could change their algorithm to detect an over-fitted context.

Someone asked whether upgrading a system using a service pack requires that the learning phase be repeated. Ding answered that the learning phase is continuous, and the rules previously learned will be used. Can a regular user determine whether the warning is true or false? Ding replied that they imagine administrators as the target users.

OFRewind: Enabling Record and Replay Troubleshooting for Networks

Andreas Wundsam and Dan Levin, Deutsche Telekom Laboratories/TU Berlin; Srini Seetharaman, Deutsche Telekom Inc. R&D Lab USA; Anja Feldmann, Deutsche Telekom Laboratories/TU Berlin

Andreas Wundsam first introduced OpenFlow as an API that allows an external server to change the TCAM table in a network switch, allowing greater flexibility in monitoring network topology changes. OFRewind is a tool which allows network configuration changes to be recorded and replayed to allow troubleshooting and problem diagnosis. Essentially, an OFRecord server is placed between the OpenFlow enabled switch and the OpenFlow controller. Configuration directives are recorded on this server or sent to another storage server somewhere else on the network.

For example, high CPU utilization is recorded on an OpenFlow switch, and this is not correlated with the arrival of any measurable parameters. The switch is a “black box,” and the existing interfaces through which statistics can be gathered (such as SNMP or the CLI) are too coarsely grained to allow the problem to be diagnosed. Using OFRewind to record and replay the network traffic, they found that the problem was caused by STATS_REQ messages, even though the arrival time of these messages was not correlated to the elevated CPU utilization. The OFRecord server does not place any overhead on the flow rate of OpenFlow messages compared to the other OpenFlow controllers.

The first questioner asked about issues with scaling this to large networks. Andreas replied that scaling for OpenFlow is still very much under investigation. Maybe a distributed controller would work. In principle, you can use the same sort of synchronization mechanisms to sync controllers. Someone else pointed out that they have data stores that are distributed and wondered if there could be a central data store. Andreas said that’s certainly possible. You could have the controller route data to a separate controller. You have to be sure not to overload your links for store traffic, though. It would be much better if the Open Flow controller had a separate interface for monitoring (OFRecord) traffic.

ORDER: Object centRic DEterministic Replay for Java

Zhemin Yang, Min Yang, Lvcai Xu, Haibo Chen, and Binyu Zang, Fudan University

Zhemin Yang introduced the difficult types of bugs in traditional multithreaded programs. What makes these bugs difficult to pinpoint is that often you cannot reproduce them, as they are not executing in a deterministic way. He presented a method and a tool (ORDER) to record and replay events in Java applications, which allows a developer to deterministically replay a sequence of events that led up to a concurrency bug. This is achieved by recording, in the JVM, all accesses to objects and associating a timestamp with these accesses. The object accesses can then be replayed deterministically, allowing concurrency bugs to be observed and fixed.

The initial performance of the system was lackluster, so several optimizations were done to improve performance. First, an observation that thread-local objects are often not involved in concurrency bugs led to an optimization where these objects are not recorded. Furthermore, recording accesses to objects which are only assigned once is also not necessary. They used SPECjvm2008, SPECjbb2005, and JRuby to evaluate the system; however, only bugs from JRuby were presented in the talk. They were able to reproduce several concurrency bugs from the open source JRuby implementation.

Had they found any bugs they could not replay? If they were looking at object granularity, they might miss a multi-variable bug. Zhemin Yang pointed out that they record and replay all objects. In the replay phase, the behavior of memory accesses is preserved.

Security and Privacy

Summarized by Raluca Ada Popa (ralucap@mit.edu)

Enabling Security in Cloud Storage SLAs with CloudProof

Raluca Ada Popa, MIT; Jacob R. Lorch, David Molnar, Helen J. Wang, and Li Zhuang, Microsoft Research

Raluca Ada Popa presented CloudProof's model. An owner stores his/her data on a cloud (that could be compromised in any way) and clients access this data based on their permissions. CloudProof considers four security properties (defined in the paper): confidentiality, integrity, write-serializability, and freshness. CloudProof not only enables clients to detect whether the cloud violated integrity, write-serializability, or freshness, but, importantly, enables clients to prove these violations to any external party. This proof-based system is critical to enabling security guarantees in SLAs, wherein clients pay for a desired level of security and are assured

they will receive some compensation in the event of cloud misbehavior. CloudProof's design uses a mechanism called attestations, which are pieces of data exchanged by the cloud and the clients for every client request; clients verify these attestations to detect integrity violations, and the owner audits them probabilistically to detect violations of write-serializability and freshness. When a violation is detected, they can construct a proof using the attestations. CloudProof aims to scale to large enterprises, so it provides a scalable access control scheme based on broadcast encryption and block families. The evaluation on real traces shows that the security mechanisms have a reasonable cost and that the owner's workload is lightweight.

During Q&A, an attendee asked what happens if there are multiple server replicas performing a put in parallel. Raluca answered that the replicas can perform the put in parallel, but the construction and signing of the attestation needs to be performed at only one of them. Another attendee asked whether the lazy revocation scheme used for access control (not covered in the talk, but explained in the paper) allows revoked users to gain access to data they should not see. Raluca answered that revoked users do not get to see the data changed after they were revoked; however, they can still see the data that had not changed since the user was revoked. The key observation here was that, before the user was revoked, that user could have read or downloaded the data he/she had access to. The final question was whether there is a tradeoff between the number of attestations the owner receives and the amount of security enforcement achieved by the owner. Raluca answered that the absolute number of attestations is not necessarily significant for how much security the owner enforces, because some applications may have customers making more frequent requests and hence generating attestations. The owner has to check all attestations to the blocks to be audited generated in a certain time interval to provide security guarantees for the duration of that interval.

jVPFS: Adding Robustness to a Secure Stacked File System with Untrusted Local Storage Components

Carsten Weinhold and Hermann Härtig, Technische Universität Dresden

Carsten Weinhold gave the talk on jVPFS, a system for protecting confidentiality and integrity of application data. jVPFS improves on its predecessor VPFS (Virtual Private File System) by minimizing threats related to unclean shutdown of the system. To protect integrity and consistency, a Merkle hash tree spans the file system, and to log updates to the file system efficiently, jVPFS uses a journaling scheme. The journal itself is protected by continuously hashing all appended records, each hash being a hash of the entire previ-

ous history. jVPFS includes a novel cooperation scheme in which trusted and untrusted components cooperate and some metadata information is revealed to untrusted code to facilitate such cooperation.

The implementation is split into an isolated part that implements security-critical functionality and cryptographic protection, and an untrusted Linux file system, which the secure part reuses for non-critical functionality. The size and complexity of the secure part should be minimized to make it more trustworthy. The overhead of jVPFS as compared to its less safe predecessor VPFS is less than 40%, often much less.

During Q&A, an attendee asked whether jVPFS prevents confidentiality loss from file sizes. Carsten answered that confidentiality protection is only at the block size, and one can indeed infer information from file sizes, although the content itself is protected. The second question was how to enforce order on write. Carsten answered that, for most file systems, one has to call `fsync()` on the journal, which guarantees that all hash-sum updates in the journal are flushed from the Linux-based part of the file system stack to the storage device. This ensures that block writes to encrypted files performed later will be written to the storage device after the journal update. For some workloads, one may also safely relax this ordering requirement. Someone asked, given that the controller may break up the writes into multiple writes, is there a possibility to order those writes? Carsten replied that, if one uses `fsync()`, this should not be a problem, because of the order guarantees of the file system.

Distributed Systems

Summarized by Carsten Weinhold (weinhold@os.inf.tu-dresden.de)

Semantics of Caching with SPOCA: A Stateless, Proportional, Optimally-Consistent Addressing Algorithm

Ashish Chawla, Benjamin Reed, Karl Juhnke, and Ghousuddin Syed, Yahoo! Inc.

📌 *Awarded Best Paper!*

Ashish Chawla presented SPOCA, a new content-addressing algorithm for Yahoo!'s video platform. The platform serves 20 million video assets to users, who make 30 million requests per day from all continents. An early off-the-shelf solution had the problem that front-end servers in the various data centers handling user requests could not optimally cache video files in memory and on disk. For example, a user who accesses an unpopular video cached on a server on another continent may observe sub-standard playback performance. The platform was also prone to redundant caching (rarely

accessed files consume cache space on many servers), and improving the cache promotion algorithm alone was not sufficient. The soft-state-only SPOCA and Zebra algorithms that are deployed at Yahoo! at large scale solve these issues.

SPOCA implements load balancing, fault tolerance, and popular content handling with efficient cache utilization, whereas the Zebra algorithm takes care of handling geographic locality. Zebra uses a sequence of bloom filters to determine popularity. New content IDs are added to the filter representing the current time interval, while filters representing older intervals are removed eventually (and new empty filters added) to implement aging. Zebra determines when to cache files locally and ultimately directs requests to a certain cluster, in which the SPOCA algorithm then selects a front-end server for the requested video. SPOCA uses a sparse hash space to map content to servers in a deterministic way. By hashing the name of a video file twice or more in case of lookup errors, the algorithm can deal with failing servers. A simple extension also based on hashing multiple times allows requests for very popular content to be routed to multiple servers in order to balance load. The evaluation results presented in the talk show that caching performance improved significantly, resulting in more streams being able to be served with less storage.

Asked about how much money Yahoo! saved because of the improvements, Ashish replied that it was “a ton.” How specific is the system for video? It is particularly well suited for big video files that are difficult to cache efficiently.

TidyFS: A Simple and Small Distributed File System

Dennis Fetterly, Maya Haridasan, and Michael Isard, Microsoft Research, Silicon Valley; Swaminathan Sundararaman, University of Wisconsin, Madison

Dennis Fetterly presented TidyFS, a distributed file system that is targeted at data-intensive scalable computing (DISC) workloads, which are built on top of data-parallel frameworks such as MapReduce, Hadoop, or Dryad. The key property of such workloads is that they store data striped across many machines in a cluster, thus enabling a high degree of parallelism. TidyFS exploits the simple access patterns of these frameworks (e.g., data becomes visible only after being fully written and committed, no overwriting) in order to make its design very simple, too. Furthermore, as the frameworks simply re-execute subcomputations in case a machine or disk fails, TidyFS does not need to provide complicated fault-tolerance mechanisms and can instead restrict itself to the simpler concept of lazy replication.

TidyFS represents files as so-called streams, which consist of a sequence of parts stored on compute nodes. A central

reliable (i.e., replicated) metadata server is responsible for mapping streams to parts. Clients request this mapping information from the server and can then access parts directly. Parts can be local NTFS files, accessible via CIFS, or even a SQL database. The storage service, a daemon process running on each storage machine in the TidyFS cluster, manages parts, replicates them (in a lazy way, as instructed by the metadata server), and takes care of garbage-collecting obsolete parts. Evaluation results collected over 18 months of active use of a 256-node research cluster with TidyFS were discussed at the end of the talk.

Someone asked about the use of local ACLs on each storage machine and how these ACLs are updated. Dennis answered that there is a delay, but updates are usually propagated within a minute or so as part of regular communication between the metadata server and storage nodes. If this is too long, the metadata server could theoretically also not provide part locations until the storage nodes were updated. Asked about how to rename a directory, Dennis replied that there are no physical directories, but clients can specify hierarchically formed pathnames nonetheless. The last question was about how to prevent data loss in spite of lazy replication, when a certain subcomputation cannot be easily reproduced after the only copy is lost. A solution to this problem would be to have the application or framework not start working on new parts until they have been replicated.

Personal Devices

Summarized by Carsten Weinhold (weinhold@os.inf.tu-dresden.de)

Eyo: Device-Transparent Personal Storage

Jacob Strauss, Quanta Research Cambridge; Justin Mazzola Paluska and Chris Lesniewski-Laas, Massachusetts Institute of Technology; Bryan Ford, Yale University; Robert Morris and Frans Kaashoek, Massachusetts Institute of Technology

Jacob Strauss began his talk on Eyo with an overview of how users handled personal devices in the past. When they only had a PC and a camera (or a cell phone), point-to-point synchronization using fast and inexpensive local connections was good enough. However, as people started to own more and more devices (e.g., external disks and laptops), a central hub such as a PC became necessary to manage their entire media collection. Unfortunately, this hub must be reachable by all devices, which is often a problem. Moving the hub to the cloud provides more flexibility, but reachability, large data volumes, and slow connections remain an issue. On the other hand, ad hoc management (use local storage, upload later, copy only some songs, etc.) requires the user to track all the objects manually in order to keep devices in sync.

To solve these connectivity and management problems, Eyo provides a complete view of all objects on all devices despite limited storage on some of them. The system is implemented as a daemon that runs on all devices. It separates metadata (which is small and can quickly be replicated on all devices) and the actual content (parts of which may exist only on some devices that have sufficient storage capacity). Eyo automatically performs peer-to-peer synchronization when devices can connect to each other and tries to resolve conflicts automatically. Conflicts can occur when a user changes (the metadata of) the same object on two disconnected devices. It also provides an API that makes version history and placement policy explicit to applications using it. This allows for sophisticated application-specific policies for conflict resolution (e.g., handling play-count of a song from two devices). The evaluation results presented at the end of the talk showed that existing applications can be adapted with reasonable effort (often because they already separate metadata and data in some way).

An attendee asked how the semantics of differently defined metadata on various devices can be handled. Jacob pointed out that there are many agreed-upon standards for JPEG or MP3 files, so a common subset often exists or user-defined fields can be used. Further, it was pointed out that Eyo extracts its own copy of metadata from EXIF fields and similar in-file metadata. Re-export for sharing is currently not supported. Another question related to video, which might be differently encoded for a smartphone and a PC. Eyo supports these by allowing different files of the same object that the metadata refers to. The mechanism is also useful for thumbnail images, which are treated as content objects, too. Deletions by accident on one device do not kill objects immediately, as they are communicated as log entries describing metadata changes first.

Autonomous Storage Management for Personal Devices with PodBase

Ansley Post, MPI-SWS; Juan Navarro, TU Munich; Petr Kuznetsov, TU Berlin/Deutsche Telekom Laboratories; Peter Druschel, MPI-SWS

At the beginning of his talk on PodBase, Ansley Post contrasted professionally managed data storage in an enterprise environment (redundant storage, offline and offsite backup) with the way ordinary users manage their data. On one hand, users own heterogeneous devices that have different use cases and connectivity. On the other hand, users are usually inexperienced and reluctant administrators who struggle with keeping data available and stored safely. PodBase intends to provide a solution for them by easily and automatically replicating files on the user's devices for increased availability and durability. Its design emphasizes minimal

user attention to accomplish this task and it uses a linear programming approach to adapt to changing conditions.

The idea is to introduce devices to PodBase once and then let the system figure out how to replicate files. An off-the-shelf LP solver is used to generate a multi-step plan (e.g., copy file A to B, copy C to A) from a set of actions, goals, and reconciled metadata (including connection history and version vectors). One particular feature detailed in the talk was an “automatic sneaker net,” where a laptop that frequently moves between, for example, a PC at home and a desktop at work is used to carry replicas of files from one machine to the other. PodBase works across a wide range of devices and supports plugins, which can implement specific strategies for synchronizing data or backup. The presented evaluation results included a 30-day user study involving 10 households with 25 devices. It showed that PodBase can indeed use otherwise unused storage to increase the replication count for files (without user interaction).

Can one prevent certain files from showing up on other devices? The system is flexible in this regard, as plugins or backlists can be used for that. Making sure that files are encrypted before being pushed to cloud storage is possible, too. An attendee asked if bandwidth can be capped. Yes, users asked for this. Asked about whether users continue to use the system after the study ended, Ansley replied that they did for some time, but then stopped because support was no longer available. Why was an LP solver necessary? They first used a greedy approach, but automatic sneakernet was not possible when the available space in a device filled up—the LP solver was more capable in resolving such situations.