conference reports

THANKS TO THE SUMMARIZERS

2006 USENIX ANNUAL TECH

Marc Chiarini
Rik Farrow
Wei Huang
John Jernigan
Scott Michael Koch
Kiran-Kumar Muniswamy-Reddy
Partho Nath
Aameek Singh
Chris Small
Yizhan Sun

SRUTI '06

John Bethencourt
Balanchander Krishnamurthy
Anirudh Ramachandran

EVT '06

Aaron Burstein
Sarah P. Everett
Dan Sandler
Ka-Ping Yee

SUMMARIES

2006 USENIX Annual Technical Conference

Boston, MA
May 30–June 3, 2006

KEYNOTE: PLANETLAB: EVOLUTION VS. INTELLIGENT DESIGN IN PLANETARY-SCALE INFRASTRUCTURE

Larry Peterson, Professor and Chair, Department of Computer Science, Princeton University; Director, PlanetLab Consortium

Summarized by Yizhan Sun

PlanetLab is a global platform for evaluating and deploying network services. It currently includes 670 nodes, spanning 325 sites and 35 countries, and has more than 3 million users. PlanetLab hosts many kinds of services, including file transfer, routing, DNS, multicast, Internet measurement, and email.

Larry Peterson summarized design requirements for the PlanetLab architecture:

1. It must provide a global platform that supports both short-term experiments and long-running service.
2. It must be available now, even though no one knows for sure what "it" is. In other words, we must deploy the existing system and software.
3. We must convince sites to host nodes running code written by unknown researchers from other organizations. This requirement is satisfied by building a relationship between users and service providers through trusted PLC (PlanetLab Consortium).
4. Sustaining growth depends on support for site autonomy and decentralized control.
5. It must scale to support many users with minimum resources available.

Peterson explained that they favor evolution over a clean slate, and design principles over a fixed architecture. Design principles include:

- leverage existing software and interfaces
- keep VM monitor and control plane orthogonal
- exploit virtualization
- give no one root (no more privilege than necessary)
- support federation

VIRTUALIZATION

Summarized by Marc Chiarini

- ***Antfarm: Tracking Processes in a Virtual Machine Environment***

Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

Stephen Jones presented an approach that allows virtual memory managers (VMMs) to track the existence and activities of guest OS processes. Process-aware VMMs are better able to implement traditional OS services such as I/O scheduling, which leads to improved performance over host and guest OS implementations. The main advantages of the authors' approach are threefold: (1) the VMM does not require detailed knowledge of the guest's internal architecture or implementation; (2) no changes to the guest OS are necessary (a big win in the case of legacy or closed-source components); and (3) accurate inferral of process events incurs a very low overhead (2.5% in their worst-case scenario). The team implemented and evaluated their techniques on both x86 and SPARC architectures with the Xen VMM hosting Linux and the Simics full-system simulator hosting Windows.

Jones described the mechanism by which a VMM can detect process creation, destruction, and context switches in the guest. On x86, Antfarm tracks the contents of the privileged CR3 register, which points to the page directory for the process currently running in the guest. When the CR3 changes to any of a particular range of values, it can be inferred that a context switch has occurred. If the CR3 is loaded with a previously unseen value, it can further be inferred (and the VMM can track) that a new process has been created. The VMM makes two more observations to determine whether a process has been destroyed: Windows and Linux systematically clear nonprivileged portions of page table pages before reusing them; the TLB must also be flushed once an address space has been deallocated. If the VMM determines that the number of assigned pages in a process's address space has gone to zero and that the TLB has been flushed (by loading CR3 with a special value), the VMM can rightly infer a process exit. Similar techniques are available for SPARC architectures. Jones concluded with a case study of Antfarm's performance improvements for an anticipatory disk scheduler: By understanding which disk I/O requests come from which guest processes, a scheduler can try to optimize requests across all processes in all guests.

- ***Optimizing Network Virtualization in Xen***

  Aravind Menon, EPFL; Alan L. Cox, Rice University; Willy Zwaenepoel, EPFL

  Awarded Best Paper!

  Aravind Menon presented three modifications to the Xen architecture that significantly improve its network performance. First, the approach implements high-level network offload features for guest domain interfaces, including scatter/gather I/O, TCP/IP checksum, and TCP segmentation offload; second, the performance of the I/O channel between the guest and network driver domains is enhanced; last, the VMM is modified to allow guest OSes to use efficient virtual memory primitives, including superpage and global page mappings.

  After a brief overview of the Xen Network Virtualization Architecture, Menon discussed the team's optimizations in detail. He noted that 60–70% of the processing time for transmit/receive operations is spent in the I/O channel and bridging within the driver domain. To help combat this bottleneck, an offload driver is inserted just before the NIC driver on the path to the physical NIC. This driver implements in software whichever offload features are not already implemented on the NIC. A 4x, 2.1x, and 1.9x reduction in execution cost was achieved in the guest domain, driver domain, and Xen VMM, respectively.

  Menon and his team also attacked the mechanisms used to transfer packets over the I/O channel between the guest and driver domains. They found that the current technique of page remapping for each network packet is not necessary in many cases. Using simple methods, such as data copying, packet header investigation, and MTU-sized socket buffers, the team achieves a 15.7% and 17% improvement in transmission and reception, respectively, across the I/O channel.

  Overall, the optimizations explored in the research improved the transmit throughput in guest domains by a factor of 4.4 and the receive throughput in the driver domain by 35%. The team needs to do further research to determine effective techniques for improving receive performance in the guest domain. In the Q&A, Mike Swift asked about other common network optimizations and how they may be applicable to this work. Menon responded that more offload features may be useful but their benefit has not yet been studied.

- ***High Performance VMM-Bypass I/O in Virtual Machines***

  Jiuxing Liu, IBM T.J. Watson Research Center; Wei Huang, The Ohio State University; Bulent Abali, IBM T.J. Watson Research Center; Dhabaleswar K. Panda, The Ohio State University

  Jiuxing Liu presented a new device virtualization model, VMM-bypass I/O, that allows guest OSes to perform time-critical I/O operations without diverting through the VMM or other specialized I/O VMs. The problems with these techniques are manifest when one considers that every I/O operation involves the VMM, making it a potential bottleneck. Additionally, the second technique results in expensive context switches. The key is to use a "guest module" device driver installed in guest VMs that handle setup and management operations of direct I/O. These modules communicate with "backend modules" within either the VMM or a privileged device driver VM. Co-located with backend modules are the original privileged modules that know how to make requests to intelligent I/O devices.

  Liu went on to describe the InfiniBand architecture and the design and implementation of Xen-IB, their IfiniBand virtualization driver for Xen. Infiniband is a high-speed interconnect to various devices that supports OS-bypass, allowing processes in a host OS to communicate (semi-)directly with the hardware. The prototype built by the

research team supports all privileged InfiniBand operations, including initialization, resource management, memory registration, and event handling. Liu gave a rundown of the InfiniBand cluster used as a testbed, consisting of Xen 3.0 running RedHat AS4 on Intel Xeon machines. Comparisons were presented between native Infiniband and Xen-IB for latency and bandwidth (negligible differences), event/interrupt handling (10- to 25-uS overhead introduced), memory registration (25–35% overhead), IP over InfiniBand (<10% throughput degradation for >16KB-size messages), and MPI bandwidth and latency benchmarks (negligible).

Finally, Liu discussed some remaining challenges. Providing a complete and efficient bypass environment requires addressing some remaining important issues, such as safe device access, QoS among competing VMs, and VM check-pointing and migration. The team is eyeing some directions they think will be fruitful. The prototype can be downloaded at http://xenbits.xensource.com/ext/xen-smartio.hg.

INVITED TALK

Deploying a Sensor Network on an Active Volcano

Matt Welsh, Harvard University

Summarized by John Jernigan

Matt Welsh related his experiences during two deployments of sensor arrays on volcanoes in Ecuador. He explained that the arrays could potentially provide civil authorities with warnings of volcanic activity and help mitigate hazards. The research team spread "motes" (small and inexpensive wireless sensors) in a swath around the volcano and measured seismic and acoustic activity in real time. By comparison, traditional methods of seismic data logging involve manual collection of data from the field site, which can be very remote and difficult to reach. In addition, wireless sensors can cover a larger amount of terrain because of lower costs.

The basic system involved the motes, synchronized by GPS timestamps and sophisticated algorithms, propagating data over an ad hoc mesh network to a radio modem that communicated with a base station many kilometers away. Deploying wireless sensors presents significant technological challenges, however, as high sampling rates generate huge amounts of data, and maintaining accurate timing of captured events is absolutely critical for use by seismologists.

He indicated that node reliability was one of the largest concerns. Ironically, it was not the sensor network that failed often in the deployment, but the base station at the observatory, where a laptop would experience sporadic electrical outages. Logistical issues and bad luck seemed to overshadow the technological acclaim of the sensors as system uptime sank. Seismologists working with the research team lost confidence in the data set as a result of reliability issues. Nevertheless, the data set could be cleaned up and analyzed using external validation techniques, including data from third-party data-logging stations.

Some of the lessons learned from the deployments were as follows: Accurate timing of captured events must be the first priority. The goals of the computer scientists and the seismologists were sometimes disparate, and this affected the usefulness of the gathered data. Nodes should have been collocated with existing data-logging stations for later verification of data. Finally, never take for granted that you can simply find an electrical outlet when you need one!

The next steps for the technology involve nailing down timing issues so that earthquakes can be localized in real time, and utilizing 3D mapping techniques to map the inside of volcanoes.

STORAGE

Summarized by Wei Huang

Provenance-Aware Storage Systems

Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer, Harvard University

Kiran-Kumar Muniswamy-Reddy presented a provenance-aware storage system. In the context of his work, provenance refers to the information that describes data in sufficient detail to facilitate reproduction and enable validation of results. Kiran-Kumar started his talk with several usage cases of provenance-aware storage, such as applications in homeland security, archiving, and business compliance, where accessing the history of files may be critical to end users. However, as Kiran-Kumar pointed out, support for provenance is very limited in file systems. Most of the current solutions are domain-specific, which may cause the data and the provenance to be out of sync. And in many cases the solutions are simply lacking.

Kiran-Kumar argued for the importance of PASS, which keeps the data and the provenance tightly bound and provides transparent management. He then introduced their design of PASS. In their design, the collector records the provenance data or events and passes the records to the file system. The storage layer, which is a stackable file system called PASTA, uses an in-kernel database engine to store the metadata. And the query tool makes the

provenance accessible to users. Kiran-Kumar showed that their implementation had reasonable overhead on applications, both spatially and temporally.

Kiran-Kumar concluded his talk with several research challenges they are experiencing through their prototype study, such as searching suitable security models, pruning of provenance, and addressing the network attached storage. In the Q&A session, Kiran-Kumar was asked whether there are any micro-benchmark evaluations for PASS. He indicated that small file operations micro-benchmarks entail up to 100–200% overhead time. However, since most applications do not access the storage system that often, the overhead is usually acceptable for applications.

- ***Thresher: An Efficient Storage Manager for Copy-on-write Snapshots***

  Liuba Shrira and Hao Xu, Brandeis University

  Thresher targets BITE (Back-In-Time Execution) applications that take snapshots of the past state, inspect the snapshots with BITE, and retain snapshots deemed as interesting for an unlimited time for future analysis. Liuba started her talk with a discussion of why today's snapshot systems are inadequate for BITE applications. She pointed out that it is critical to provide applications with the ability to discriminate among snapshots, so that valuable snapshots can be retained while the less valuable ones can be discarded or moved offline, because although disk space is cheap, administration of storage becomes costly.

  In the second part of the talk, Liuba introduced Thresher, a snapshot storage manager based on new copy-on-write snapshot techniques. Thresher is the first to provide applications with the ability to discriminate among snapshots efficiently. Liuba focused on two important concepts in Thresher: discrimination and segregation. Applications discriminate among snapshots by ranking them according to their importance. The storage manager segregates differently ranked snapshots efficiently, so that higher-ranked snapshots can be accessed faster and lower-ranked snapshots can eventually be discarded without affecting the accessibility of higher-ranked ones and without disk fragmentation.

  Lazy segregation technique allow the rank of snapshots to be specified after the snapshots are taken, enabling BITE-based ranking. Liuba focused on the diff-based segregation technique and the optimizations for low-cost reclamation and faster access to snapshots. Liuba concluded her talk with performance evaluation of Thresher. She showed that lazy segregation and faster snapshots can be implemented with very low performance overhead, allowing a huge reduction in storage requirements for snapshots.

- ***Design Tradeoffs in Applying Content Addressable Storage to Enterprise-scale Systems Based on Virtual Machines***

  Partho Nath, Penn State University; Michael A. Kozuch, Intel Research Pittsburgh; David R. O'Hallaron, Jan Harkes, M. Satyanarayanan, Niraj Tolia, and Matt Toups, Carnegie Mellon University

  Partho Nath presented their experience on applying Content Addressable Storage (CAS) to enterprise-scale systems based on virtual machines. Partho first described the Internet suspend /resume (ISR) client-management system, which is the execution environment at which their work is targeted. ISR is a virtual-machine-based client management system. It stores the user execution environments as parcels, which are the complete VM images, including memory and disk snapshot. Different versions of parcels are stored in a lossless manner.

  Partho then asked two questions, both of which are answered by their evaluations in this paper: Can content-aware storage reduce the (1) storage and (2) network requirements in ISR systems? And, if so, by how much? Their evaluation consisted of three dimensions: the policies, the chunk size, and gzip compression. They evaluated three policies for managing the parcels: the non-CAS baseline policy ("delta"), which stores different versions of parcels for each user as the diff of the previous version; the intra-parcel policy, where each parcel is represented by a separated pool of unique chunks shared by all versions from the same user; and the ALL policy, where all parcels for all users are represented by a single pool of chunks. Gzip can be used to further compress the data.

  Partho showed their evaluation results. He pointed out that adopting CAS into the storage system significantly reduces storage requirements, especially when using relaxed policy (ALL policy). And within CAS policies, using smaller chunks works best in spite of metadata overheads. Another important observation is that CAS policies alone can consume less storage than a non-CAS policy with gzip compression, which avoids the expensive compression operations. In response to a question on the performance overhead of hash calculation for CAS policies. Partho indicated that they had not experienced any noticeable slowdown for hash calculations.

INVITED TALK

Panel: Open Source Software Business Models

Mike Olsen, Oracle, Sleepycat; Brian Aker, MySQL; Miguel de Icaza, Novell, Ximian

Moderator: Stephen Walli, Optaros, Inc.

Summarized by Scott Michael Koch

The discussion began with each panelist sharing his opinions and experiences with OSS. Although the panel agreed that OSS businesses can be very successful, Miguel felt that giving away your company's product for free was a risk, and he does not recommend starting a business of this type. The panel seemed to agree that there are only certain circumstances in which a OSS business can have success. Mike reminded us that it is hard to start any sort of business, and Brian added that selling any sort of software, whether proprietary or open source, today is like "setting up a tip jar" in that you just hope that enough people are willing to pay for your software. Brian felt that, if you want to make money, a service and support model or an ASP model makes the most successful long-term option, instead of trying to sell a binary. It was pointed out that people are becoming very comfortable with the subscription model. Miguel felt that the model of building a proprietary server with free clients was the way to go. Everyone agreed that for the traditional model of selling OSS to be successful, it was key to find a niche in the market where your product was something that everyone needed. Mike summarized this well by saying that using open source software can be successful as a tactic if it supports your overall strategy as a business.

The panelists then went on to talk about the interactions and relations with the communities that surrounded their respective businesses. Mike explained that the community surrounding Bdb consisted mostly of users of the Bdb library, and although they benefited from the many eyeballs examining their code and enforcing high quality, there are no outside contributors. For MySQL, Brian said that ideas for features and quality bug reports are the most important contributions they receive from their community. Miguel explained that his current project, Mono, receives many external code contributions, and he believes that the amount and type of contributions strongly depend on the maturity of the code base. Mike then said that the most important contribution from the community is the adoption of their software, which increases the visibility and popularity of the software in the community. Inspired by a question in the audience, the panel discussed some lessons they had learned from their past experiences with OSS businesses. The only common problem they mentioned was that it can be frustrating trying to deal with the slashdot-type community, and anyone starting an OSS business should be aware of the energy and effort required to constantly nurture that community. Learn to communicate with your audiences appropriately. Marketing to the typical OSS user is best done through attending conferences, setting up blogs, and communicating with them one-on-one.

SHORT PAPERS SESSION I

Summarized by Kiran-Kumar Muniswamy-Reddy

Compare-by-Hash: A Reasoned Analysis

J. Black, University of Colorado, Boulder

John Black presented this paper, a rebuttal to Val Henson's HotOS 2003 paper that criticized the use of hash functions to compare two files to tell whether they are the same. John presented various arguments to make the point that although hash functions may not be strong enough for scenarios where there is an adversary, they are more than sufficient for usage scenarios where there is no adversary. John concluded the talk by stating that the computation power needed to find collisions in a 128-bit hash function in 24 days would cost around $80,000, and for a SHA1 it would take $80,000,000 and 2 years. So other approaches such as social engineering might be more successful. In the Q&A session, John agreed that the current hash functions may not be secure after 20 years.

An Evaluation of Network Stack Parallelization Strategies in Modern Operating Systems

Paul Willmann, Scott Rixner, and Alan L. Cox, Rice University

The paper was presented by Paul Willmann. The paper evaluates three different strategies for parallelizing network stacks: (i) message-based (MsgP), (ii) connection-based using threads for synchronization (ConnP-T), and (iii) connection-based using locks for synchronization (ConnP-L). MsgP is the slowest of the three, as it has a significant amount of locking overhead. ConnP-T has lower locking overhead but experiences significant scheduling overhead. ConnP-L has the best performance, as it mitigates both locking and scheduling overheads. Paul concluded the talk by stating that current programs themselves haven't been written to take advantage of parallelism.

■ ***Disk Drive Level Workload Characterization***

Alma Riska and Erik Riedel, Seagate Research

The paper, presented by Eric Riedel, characterizes workloads in various kinds of devices, including PCs, laptops, and home devices. The authors collected traces by inserting SCSI or IDE analyzers into the I/O bus and intercepting the signals. Some of their findings are as follows. The read/write ratio, the access pattern, and write traffic vary by application. The request size is around 4 kB. I/O bus and disks are underutilized. In the enterprise/desktop environment, requests are spread all over the disk. Videos are highly sequential. Access characteristics depend on the environment: cache management, arrival, and service processes at the disk drive. Characteristics common in environments include idleness and burstiness.

Two key questions were addressed in the Q&A session. (1) How are your results different from the Hewlett-Packard paper? Eric replied that they don't compare, because of the large difference between the devices and environment presented in this paper and those in that paper. (2) Can we get the traces? Eric replied that they may be able to give out the traces.

■ ***Towards a Resilient Operating System for Wireless Sensor Networks***

Hyoseung Kim and Hojung Cha, Yonsei University

Currently, the only way to recover from crashes in sensors is to reset the sensors. Hyoseueng presented RETOS, a resilient, expandable, threaded operating system. RETOS achieves this by introducing dual mode operation and static/dynamic code checking. Dual mode separates out the application and kernel code. RETOS then performs checks on the machine instructions to ensure that applications do not write to or jump to an address outside their logical portion. Some of the instructions can be verified statically at compile time and others need to be verified at run time; for the latter, verification code is injected while compiling the code.

■ ***Transparent Contribution of Memory***

James Cipar, Mark D. Corner, and Emery D. Berger, University of Massachusetts, Amherst

The talk was given by James Cipar. Contributory applications such as condor, SETI@home, and Farsite utilize wasted CPU cycles, idle memory, and free disk space on participating user machines. They can, however, disrupt user activity by forcing user pages to disk. Normal approaches such as scheduling do not help with memory and disk usage. James presented the Transparent Memory Manager (TMM), which controls memory usage by contributory applications, thereby ensuring that it does not impair normal system functionality. TMM works by detecting the imprint of user applications and then limits the memory footprint of contributory applications accordingly. TMM detects the memory imprint by keeping an LRU histogram of memory accesses. When pages need to be allocated but there are no free pages and both normal and contributory apps have exceeded their limit, normal apps are favored. Otherwise, the page is evicted from the class that has exceeded its limit.

INVITED TALK

■ ***Success, Failure, and Alternative Solutions for Network Security***

Peiter Zatko, BBN Technologies

Summarized by John Jernigan

Peiter Zatko, a.k.a. "Mudge," spoke of the current state of affairs in network security, offering his musings and concerns. He began with a summary of his background; he is a former member of l0pht, has worked with the National Security Council, and started his own security company, Intrusic. He is now working for BBN Technologies.

Peiter first addressed some of the pertinent questions in network security today, asking how much progress we have really made, where we have messed up, and where we are spinning our wheels. He points out that the Internet has far outpaced our understanding of security. As the Internet grew and added nodes and users, the threat model increased, but software was still not being designed with any notion of security. Eventually, a distinction between internal and external environments evolved, much like a military compound with a fence and a gateway to swap credentials, but internal resources were not themselves secure. Presently, many networks are watched by intrusion detection systems, which only let in and out certain traffic and flag dubious behavior. However, 0-days still penetrate these defenses and will always be one step ahead of patches by definition. Of even greater concern is that the defenses do little to prevent unauthorized activity within the network itself. Peiter emphasized that our threat model has changed, but our defenses have not grown with the environment.

On the topic of buffer overflows, he suggests that, even if they all went away, we would be left with plenty of threats, such as rootkits, sniffing, and trojaned applications. In addition, overflows of many different types, such as heap-based and pointer overflows, abound. In other areas, it has become too easy for naive developers to create enterprise applications, such as with PHP, and vulnerable software is live and rampant.

Peiter suggested that firewalls, intrusion detection systems, and intrusion prevention systems are not really the answer to our security woes. We should really be looking at what goes on inside a network as well. We should not see drastic changes in the behavior of nodes or out-of-order packets on internal systems with few routers. We need to adhere to RFCs and also to detect when behavior does not match real-world trends on the network.

The thought we are left with is that security is still a cat-and-mouse game, and more intelligent methods of security are strongly needed to keep pace with developing technologies and Internet expansion.

SERVER IMPLEMENTATION

Summarized by Scott Michael Koch

- ***Implementation and Evaluation of Moderate Parallelism in the BIND9 DNS Server***

  Tatuya Jinmei, Toshiba Corporation; Paul Vixie, Internet Systems Consortium

  Tatuya Jinmei presented a paper about improving the performance of ISC's BIND9, a widely used DNS server. The authors found that it had poor performance with threads and did not benefit from having multiple CPUs. Some of the key bottlenecks they found were memory management, operations on reference counters, and thread synchronization. While looking for these bottlenecks they found that 43% of total run time was spent waiting to acquire locks. They were able to eliminate the bottleneck in memory management by enabling a internal memory allocator to provide each thread with a separate pool of memory, and they also separated the workspaces of the threads since the temporary data used by a single thread did not need to be shared. They eliminated the bottleneck on reference counters by using atomic operations without locks instead of using pthread locks. Although this solution is less portable, since it depends on specific hardware architectures, all the same platforms are supported, as before, through an abstract API. They also implemented more efficient reader-writer locks by basing the design on Mellor-Crummey's Algorithm.

  By identifying and eliminating the thread synchronization overhead and these other bottlenecks, they significantly improved BIND9 performance with multiple threads. They confirmed their improvements by testing them on a four-way machine. Their improvements should be available in BIND9 as of version 9.4.0a5. Although they focused on BIND9, they feel the techniques and improvements that they used are applicable to other thread-based applications.

- ***Flux: A Language for Programming High-Performance Servers***

  Brendan Burns, Kevin Grimaldi, Alexander Kostadinov, Emery D. Berger, and Mark D. Corner, University of Massachusetts, Amherst

  Brendan Burns talked about a new programming language with the goal of simplifying the process of building high-performance servers. The authors felt that, when building these types of servers, having to deal with the thread programming makes the code much harder to reuse and adds the possibility of deadlocks in the code. Having to worry about threading is an unnecessary burden on the programmer and can significantly complicate debugging. Flux aims to separate the programming process so that all the concurrency control is taken care of with its simple language, and the logical programming of the server is done in C, C++, or Java. They found that programming with this separated method allowed the programmer to better understand the overall functionality of the different parts of the server without having to worry about the underlying implementation of each of the parts. Using Flux they were able to put together a Web server, image rendering, a BitTorrent peer, and a game server that performed as fast as or faster than their counterparts written entirely in C. More information and a working example of both the HTTP and BitTorrent Server can be found at http://flux.cs.umass.edu/.

- ***Understanding and Addressing Blocking-Induced Network Server Latency***

  Yaoping Ruan, IBM T.J. Watson Research Center; Vivek Pai, Princeton University

  The last paper in this session was way over my head. Even after going though the presentation and attempting to read the paper, any attempt at writing a summary just turned into trying to reword the abstract of the paper. You can find out more about this paper at http://www.cs.princeton.edu/nsg/papers/latency_usenix_06/.

INVITED TALK

Is University Systems Teaching and Research Relevant to Industry?

Moderator: Gernot Heiser, NICTA/UNSW

Panelists: Stephen Geary, HP, head of Linux strategy in R&D; Orran Krieger, IBM, K42, Xen strategy; Margo Seltzer, Harvard, Oracle, Sleepycat; Tim Roscoe, Intel Research Berkeley, OS, distributed systems; Jim Waldo, Sun Labs, Jini, Harvard; Andy Tannenbaum, Vrije University, Minix, 16 textbooks

Summarized by Chris Small

Gernot started by stating the claims of industry: that universities are not producing the kinds of systems people need and are producing irrelevant research. Industry does research, but because it doesn't get published, industry gets no respect from academia. He then asked each of the six panelists to respond to his opening statement.

Andy Tannenbaum: What are universities for? To serve students? Industry? Government? Faculty? The average student's career lasts 40 years; I want to focus on stuff that will be useful for 20 years, emphasizing principles, not facts. Teaching how MS-DOS works might have been very interesting 20 years ago but is less interesting now. I want to teach how to keep the design simple, good software engineering practice, and to expect paradigm shifts. Think in terms of systems. Ignore hype; don't forget the past; ideas get recycled. I think sometimes I'm supposed to teach "bloat-ology."

Jim Waldo: I'm the industry guy and mostly agree with Andy. But he's going to concentrate not on 20 years from now but now. Students never have to maintain a system longer than it takes to write the paper. "I don't fix bugs; I have a Ph.D.—I write new things." When you get your Ph.D. you're not done: You're ready to start. But how can you teach system building and maintenance in a university setting? You can't. Industrial research used to be "short-term"; academics did "longterm" research. But it's no longer true—academics are doing short-term one-off things to get the next grant; industry looks at the longer term. So people coming out of universities are not ready for the adult world, but Jim doesn't expect them to be.

Tim Roscoe: I don't like the division between industry and academic research. Intel sets up lablets of 10–20 researchers, closely attached to the university. They do not pursue patents on joint work. Everything is supposed to be published, open source, etc. Intel can do this because Intel is a manufacturing company, not a software company. It probably doesn't make sense for Microsoft to do this. Intel wanted to strategically influence the way universities work. Can we get universities to do work that's of more value to Intel? Intel provides industrial relevance and resources. PlanetLab is an example—an attempt to change the research culture in distributed systems in academia. There is less emulation and less "we ran this on 17 machines, so clearly it scales up to 100,000 nodes" thinking. Students take their distributed systems textbook and try to implement the ideas on PlanetLab—and it doesn't work. They learn a lot about what really matters by doing it. How do you teach systems principles? It's very hard, unless you're getting experience building real systems.

Margo Seltzer: There are two different topics here that this nice academic-oriented panel are trying to hide from you. Undergrad education and grad education are two very different things. This conference is for graduate types, but industry mostly hires undergrads. Andy is fundamentally wrong: Universities and colleges are there to serve society, not students or faculty. We need to help students figure out what path they want to follow and how to follow it, not to build raw fodder for industry, nor clones of ourselves. They want to develop thinkers and people who can make good decisions, even if they end up being lawyers. Tension exists between giving them tools and giving them a specific skill set. In the long term, the tools are more important.

Orran Krieger agrees with Jim that there has been a longterm/short-term inversion. K42 was developed even though many people in the company thought it was a waste of time, but important skills and knowledge were brought into the company—for example, Linux and pervasive virtualization. What we want from Ph.D.s are people who will come up with radical ideas to change things. Hammond said, "Don't read all the relevant literature—think about the fundamentals and the problem for a month, then go read the literature." Researchers should work on big, irrelevant systems and work in teams. We used to have five-, six-, and seven-year Ph.D.s, and that gave them time to thrash and come up with their own ideas.

Stephen Geary: Hey, I'm a mechanical engineer. I have product responsibilities, making sure that Linux and open-source technologies work on Itanium-based systems. A chunk of code or a piece of research by itself is not interesting, or not as interesting as long-lived supported systems that do things for customers. You get them for four years; I get

them for 40 years. You have to teach people about budgets and schedules.

Andy: The job of the university is to serve society, but they're turning out lawyers.

Jim: What I'm really looking for when I'm hiring is people who know "how," not who know "that"—people who know how to think, not people who know facts (e.g., how to build a particular kind of hash table).

Q. Is academia doing anything right?

Margo: We need to adjust expectations. Andy's students understand how to think about systems, but they don't understand every line of Windows.

Gernot: How is it that academia can churn out mechanical and electrical engineers but not computer systems folks? Why are there so few real systems departments?

Orran: Linux progress is much slower than it should be because they ignore the literature. They did a brilliant job of cloning UNIX. But that's not going to revolutionize the field. The success of Linux has stifled the ability to do the kind of research that will move the field forward. Ten years ago there were more ideas moving things forward.

Andy: It's not the job of universities to produce open source code. But many of the people producing open source code are university graduates.

Q. People build their own tools. They should come out of university with the start of their own personal toolkit.

Tim Roscoe: That's insightful. One thing I've noticed about textbooks, particularly in systems, is that almost all of them are useless at teaching how to think about operating systems, planning to build a system, or how to deal with a large body of code.

Margo: Open source is a fraud—there are a handful of people who commit to the Linux source tree, not tens of thousands.

Orran: We should have a tax on corporations—where their top people come from, money goes.

Margo: Computer science headcount is plummeting. Students think "computer science means programming, and programming will be outsourced."

Orran: One of the best things that is happening to academia is dropping enrollment. People used to get into academia because they were excited; for a while these were people who thought it was a good career move. Now a higher percentage of people are passionate about it.

Margo: It's not that we're only getting the passionate people. In 1992 (at Harvard) we had 30 concentrators; this year we have 12. People who are passionate about technology think, "Oh, I know how to write programs; I don't need to study computer science." Or people in other intellectual disciplines (e.g., physics), who used to have to learn how to program to get a summer job, got seduced, but now they learn these things in high school and ignore computer science in university.

Q. Of 16 graduates, 11 were double majors, and these were mostly in economics.

Gernot: To wrap up, what can we do? Or should we just give up?

Stephen: Gelato Consortium is a good example; it was founded by HP university relations to advance Linux, Itanium, and supercomputing.

Clem Cole: We need to teach people how to collaborate.

SECURITY

Summarized by Yizhan Sun

- ***Reval: A Tool for Real-time Evaluation of DDoS Mitigation Strategies***

  Rangarajan Vasudevan and Z. Morley Mao, University of Michigan; Oliver Spatscheck and Jacobus van der Merwe, AT&T Labs—Research

  An ISP network today faces many DDoS attacks. The defense decision for DDoS attack is often manual and complex. Many defense/mitigation strategies are available, and it is difficult for a network operator to choose the appropriate one in real time. The approach presented here is the Reval simulator framework.

  Reval takes network state, attack info, and mitigation policy as input and goes through initialization, mitigation setup, traffic setup, and evaluation steps. The output of Reval is the optimal solution for a DDoS attack.

  A case study on the Abilene network was illustrated in the talk. Two mitigation mechanisms can be applied in this case: blackholing and scrubbing. The result of using Reval to determine the right mitigation strategy in real time was explained and evaluated.

- ***LADS: Large-scale Automated DDoS Detection System***

  Vyas Sekar, Carnegie Mellon University; Nick Duffield, Oliver Spatscheck, and Jacobus van der Merwe, AT&T Labs—Research; Hui Zhang, Carnegie Mellon University

  Several strategies and their drawbacks for DDoS attacks were introduced:

  - Wait for customer to complain—not effective at all
  - Buy a per-egress detection device—expensive and not scalable
  - Install devices at select locations—gives incomplete coverage and inaccurate limits on sensitivity

- Use existing data feeds (e.g., SNMP and Netflow)
- Use SNMP—entails low overhead and yields few false negatives, but has low diagnostic ability
- Use Netflow—has good diagnostics, yields few false positives, but has higher overhead and does not scale

LADS is a better approach. The mechanism behind LADS is to use time-series anomaly-detection triggers collection of Netflow and do fine-grained analysis afterward. Benefits of LADS include detection of high-impact attacks, efficient data collection and reduced computational cost, and flexibility.

- ***Bump in the Ether: A Framework for Securing Sensitive User Input***

Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter, Carnegie Mellon University

Jonathan McCune first introduced how a user's input (user name and password) can be stolen by a malicious application installed on Windows systems. Then he introduced a threat model and some assumptions of BitE, including a priori knowledge of which software is good. Then he proceeded to explain BitE architecture, setup, and operation.

BitE system architecture is based upon a partially trusted host platform with a BitE Kernel module installed and executed. The BitE kernel module and mobile client participate in key setup and bypass the traditional input path to avoid information being stolen by malicious applications.

BitE can be set up through device association and application registration and operates through several steps, including application request, verification of attestation, user interaction, and establishment of session keys.

INVITED TALK

- ***Architectures and Algorithms for Biomolecular Simulation***

Cliff Young, D.E. Shaw Research, LLC

Summarized by Partho Nath

This talk by Cliff Young was on the need for developing more powerful hardware to get closer to answering challenging questions in modern biology, chemistry, and medicine. A typical means of understanding phenomena in these fields is via molecular dynamics (MD)—simulation of biologically significant molecules at the atomic level. If performing such experiments were accurate and infinitely fast it would be easy to perform arbitrary computational experiments such as determining structures by watching them form, transforming measurements into data for mining later, etc. However, for a goal of, say, simulating about 64,000 atoms at a millisecond scale, with explicit water molecules, one would need a 10,000-fold increase in computational power if a single state-of-the-art processor were used, or a 1,000-fold speedup if a modern parallel cluster were used. The talk considered the pros and cons of several different available architectural options: (a) clusters of commodity processors, (b) general-purpose supercomputers (e.g., Blue-Gene), and (c) special-purpose supercomputing architectures.

The speaker was of the opinion that new specialized, enormously parallel architectures with special-purpose ASICs specially tailored for MD simulations are the answer. Optimizations could include arithmetic specialization, hardware tailored for speed (e.g., hardware tables with parameters) but not too programmable, data flow to exactly where the data is needed, and design for almost never touching off-chip memory. Given that the class of algorithms to be run on these machines is well known, such a machine could be an order of magnitude faster than general-purpose supercomputers. The speaker commented that production of such a machine was already underway and could be expected in 2008. This machine is designed to have 16 segments at the physical level, each consisting of 512 nodes (ASICs) in a 8-cube toroidal mesh (to reflect the physical space being simulated). The speaker detailed the performance of this machine for the NT algorithm (a parallel algorithm for range-limited pairwise interactions of atoms). He noted that this architecture showed asymptotically less inter-processor communication, which translates to better scaling.

Most of the questions to the speaker addressed the machine under production. Regarding soft errors (given that the machine has thousands of nodes), the speaker commented that off-chip memory has ECC, whereas on-chip memory is supposed to be free from such errors. Additionally, the runtime does a checkpoint and reload of the simulation once every hour. Another question was whether writing code for such specialized hardware was going to be a significant bottleneck. The speaker agreed that this might be a significant issue, especially given that programmers were writing code in assembly for a specialized hardware. No compiler was being developed because the development cycle for a compiler would be longer than that of developing the code for the corresponding algorithms in

assembly itself. Given that the architecture is simplified by the absence of both speculation and out-of-order execution, writing efficient code for such an architecture may not be too bad. Answering a query on power demands, the speaker said that housing the machine would be another nontrivial task, both in terms of the physical space required and the cooling costs. On a question on the numeric precision of the machine, the author remarked that no floating-point arithmetic is used. Computations use a fixed-point subset of double-precision, i.e., 32-bit single-precision fixed-point arithmetic. The advantages gained here were that the simulation runs would be more deterministic and that the pipeline design would be simpler. Another question was on whether such a machine is viable at all: Given that the world market may absorb only about five such machines, would it not be cheaper to just build commodity clusters instead of such a specialized cluster? The speaker commented that with commodity clusters a 1,000-fold speedup would not be possible in a five-year timeframe. The speaker conceded that the size of the market justified by such an investment is still an open question.

MANAGEMENT AND ADMINISTRATION

Summarized by Kiran-Kumar Muniswamy-Reddy

- ***Sharing Networked Resources with Brokered Leases***

  David Irwin, Jeff Chase, Laura Grit, Aydan Yumerefendi, and David Becker, Duke University; Kenneth G. Yocum, University of California, San Diego

  David Irwin presented Shirako, a system to coordinate resource allocation between providers and consumers. Shirako introduces brokers, a software entity that maintains inventories of resources offered by providers and matches requests with available resources. Leases are used to bind a set of resource units to a consumer for a lease term. Brokers issue tickets to consumers that are redeemed for leases at the providers. Shirako's design makes resource allocation independent of the application. During the Q&A, Vivek Pai asked how Shirako knows what data the application needs (i.e., what do you do when the applications need disk space but not CPU?) David replied that they tried to allocate better bandwidth and to allocate systems closer to the consumers. Vivek then asked how they dealt with applications that checkpoint their state and restart on a different system. David replied that other groups have been looking at this and they plan to build on that work.

- ***Understanding and Validating Database System Administration***

  Fábio Oliveira, Kiran Nagaraja, Rekha Bachwani, Ricardo Bianchini, Richard P. Martin, and Thu D. Nguyen, Rutgers University

  The talk was given by Fábio Oliveira. The goal of this work is to reduce database downtime. Most of database downtime is caused by mistakes made by database administrators. To this end, the authors conducted a survey of experienced administrators at SAGE to better characterize the source of these errors. They found that one common source of errors is that the deployment environment is different from the test environment. They also found that DBAs of all experience levels are prone to make mistakes.

  They presented three forms of validation to reduce operator errors: trace-based, replica-based, and model-based. In the trace-based approach, they log the requests to and replies from the live system, play the traces onto the system/components to be tested, and compare the two results to detect any errors. Some operations, such as change in schema, cannot be validated by the first two methods, so they propose a model-based approach. In this approach, the operator can specify the expected behavior using their model. The dynamic behavior of the system is then validated with that of the predicted model. In the Q&A, Atul Adya asked whether they closed the loop, that is, did they go back to the DBAs with their results? Fábio replied that they did not. In response to another question by Atul, Fábio replied that they did not deal with triggers.

- ***SMART: An Integrated Multi-Action Advisor for Storage Systems***

  Li Yin, University of California, Berkeley; Sandeep Uttamchandani, Madhukar Korupolu, and Kaladhar Voruganti, IBM Almaden Research Center; Randy Katz, University of California, Berkeley

  The talk was given by Li Yin. The common approach to meeting the service level objective (SLO) for storage systems involves the observe, analyze, and act loop. This approach involves manual interaction and is slow. There are existing tools that help automate the task, but these are again restrictive, as they can correct only one action, such as work throttling, data migration, or addition of new resources. Li presented SMART, a framework that considers multiple corrective actions.

  SMART aims to maximize the system utility for a give optimization window. SMART contains four key components: (1) INPUT modules (containing sensors monitoring system state, SLOs, component modules, workload request rate, etc.), (2) a utility evaluator (which calcu-

lates the overall utility delivered by the system), (3) single action tools (to automate invocation of a single action), and (4) an action advisor that, based on the other three components, generates a schedule for actions to be invoked to improve system utility. The action advisor operates in two different decision modes: normal and unexpected. In normal mode, it proactively generates decisions to forecasted workloads by optimizing local actions to achieve global optima. In unexpected mode, it makes defensive decisions in response to unexpected variations in workloads. The reason for it being defensive is that the unexpected workload may be transient. There were no questions after the talk.

INVITED TALK

Permissive Action Links, Nuclear Weapons, and the History of Public Key Cryptography

Steven M. Bellovin, Columbia University

Summarized by Partho Nath

This talk traced the history of PALs (Permissive Action Links), detailing the motivation for their invention and those responsible for their creation. The speaker ran through a timeline of their use and evolution, highlighting the possible design choices made at those junctures, along with cryptography and key management for the different designs. The talk concluded with possible designs for modern-day PALs and what we might learn from them in designing secure systems. The slides for the talk can be found at http://www.cs.columbia.edu/~smb/talks/pal.pdf. The content for the talk can be found at http://www.cs.columbia.edu/~smb/nsam-160/pal.html.

SHORT PAPERS SESSION II

Summarized by Wei Huang

sMonitor: A Non-Intrusive Client-Perceived End-to-End Performance Monitor of Secured Internet Services

Jianbin Wei and Cheng-Zhong Xu, Wayne State University

Jianbin Wei first described the inadequacies of existing approaches for monitoring end-to-end user-perceived performance of Internet services, especially with increasing deployment of HTTPS services. Jianbin indicated that there is a strong need to deploy a performance monitor, which is nonintrusive, easy to deploy at the server side, and can handle HTTPS services.

Jianbin presented sMonitor, their solution to these goals. sMonitor consists of a package capture to collect live network packets, a packet analyzer to reconstruct the pages of HTTP/HTTPS transactions, and a performance analyzer to derive client-perceived response time of the monitored services. Jianbin focused on their solutions to several key design challenges, such as identifying encrypted HTTP requests from packet size analysis, handling pipelined requests, and parallel downloading.

Jianbin concluded with their evaluation of the accuracy of sMonitor in measuring HTTPS and HTTP services. He showed that errors between the client measurements and the reported performance of sMonitor, which is deployed at the server, are less than 8%.

*Privacy Analysis for Data Sharing in *nix Systems*

Aameek Singh, Ling Liu, and Mustaque Ahamad, Georgia Institute of Technology

The *nix access control model, as Aameek Singh pointed out, must provide good support for both data selectivity (e.g., which data to share with other users) and user selectivity (e.g., with whom to share the data).

However, Aameek's study revealed that, in current *nix systems, the lack of convenience in data-sharing mechanisms often leads to users compromising their security requirements to conveniently fit the specifications of the underlying access-control model. Aameek talked about their studies on two multi-user *nix installations. Simply by scanning readable user directories and guessing executable-only directories, along with email and browser statistics, they were able to "attack" massive amounts of privacy data, which, they believed, were not exposed on purpose. Since the technical sophistication of the attacks is low and there is no quick fix to such vulnerabilities of private data, Aameek raised a major concern about the inadequate protection of privacy in *nix systems.

Aameek concluded the talk with some possible solutions to enhance privacy protection, such as using privacy auditing tools to monitor potential privacy data exposures or virtualizing the file system hierarchy differently for different users. But until that happened, Aameek said, users should pay more attention to monitoring the privacy of their own data.

Securing Web Service by Automatic Robot Detection

KyoungSoo Park and Vivek S. Pai, Princeton University; Kang-Won Lee and Seraphin Calo, IBM T.J. Watson Research Center

KyoungSoo Park presented a automatic robot detection framework to support a secure Web service. KyoungSoo first talked about the widespread existence of malicious bots, including those for password cracking and DDoS attacks. The increasing

abuse of robots motivated an accurate robot detection system.

KyoungSoo described their techniques to separate human browser activities from robot-generated Web traffic. They include browser detection and human activity detection. Browser detection is based on the observation that most robots are not standard browsers; it catches robots if the behavior deviates from that of normal browsers. Human activity detection directly detects humans by observing human activities such as mouse movement or keyboard events behind the browsers. Hardware events are being tracked in dynamically embedded Javascript and the activity is indirectly reported to the server via a fake image request. This technique is based on the fact that current robots are not generating hardware events.

KyoungSoo showed that most human activities can be distinguished within tens of HTTP requests. And the maximum false positive rate is low (2.4%). KyoungSoo also mentioned that with their system deployed on a CoDeeN content distribution network, complaints on robot-related abuse have dropped by a factor of 10. KyoungSoo admitted that serious hackers can still break their detection system and suggested using machine-learning techniques as a remedy.

- ***Cutting through the Confusion: A Measurement Study of Homograph Attacks***

  Tobias Holgers, David E. Watson, and Steven D. Gribble, University of Washington

  David Watson introduced a measurement study of homograph attacks. A homograph is a character or string that is visually confusable with a different character or string. A homograph attack tries to fool a user into visiting a nonauthoritative Web site.

  David presented a study on the nature and quantity of homograph attacks. Using a nine-day trace of Web traffic from the Computer Science Department of the University of Washington, they probed the DNS to find registered names that are confusable with (i.e., a homograph to) the names of visited sites. The results of the study were fourfold: (1) No user visited a nonauthoritative site during the trace; (2) popular Web sites are more likely to have registered confusable names than unpopular sites; (3) registered confusable names tend to consist of substitutions of two or fewer confusable Latin characters, though some IDN (International Domain Name) substitutions were found; and, (4) the intent behind most registered confusable names is benign—predominantly advertisements. David concluded that homograph attacks currently are rare and not severe in nature. However, given the recent increase in phishing incidents, homograph attacks seem like an attractive future method for attackers to lure users to spoofed sites.

- ***Stealth Probing: Efficient Data-Plane Security for IP Routing***

  Ioannis Avramopoulos and Jennifer Rexford, Princeton University

  Ioannis Avramopoulos started his talk by introducing the challenges in secure IP routing. He argued that data-plane monitoring must be part of any complete solution. However, existing proposals for secure forwarding with link-level fault localization capability are heavyweight, requiring cryptographic operations at each hop in a path. Ioannis presented a lightweight data-plane mechanism that monitors the availability of paths in a secure fashion. In intradomain routing, this mechanism also enables the management plane to home in on the location of adversaries by combining the results of probes from different vantage points (called Byzantine tomography). Ioannis discussed advantages of stealth probing, including its incremental deployability, backward compatibility, and incentive compatibility.

  Ioannis presented two deployment scenarios for stealth probing. He described how an ISP can deploy stealth probing to secure its own infrastructure. He also discussed how a pair of edge networks can deploy stealth probing to secure the path through untrusted ASes on the Internet.

INVITED TALK

- ***Gold and Fool's Gold: Successes, Failures, and Futures in Computer Systems Research***

  Butler Lampson, Microsoft Research

  Summarized by Kiran-Kumar Muniswamy-Reddy

  Butler Lampson started off by discussing trends in computer use. He then briefly enumerated things in the history of computer science that worked, things that didn't work and why they didn't work, and a list of things that "maybe" worked. He claimed that the future of computer science lay in applications that dealt with avoiding catastrophes and uncertainties.

  In the context of Moore's law, improvement in hardware simplifies software. Better hardware enables new applications with the complexity going into software. Accordingly, the fields in which computers have been used has been growing. In the 1950s, computers were used for simulation. In the 1980s, they were used for communication and storage (e.g., email, airline tickets, and search engines). By

2010, computers will be embodied in the physical world, that is, interacting nontrivially with the physical world, embedded in factories, cars, robots, etc.

He then gave a list of things that worked: virtual memory, address space, packet nets, objects/subtypes, transactions, RDB and SQL, bitmaps and GUIs, the Web, and algorithms. The list of things that did not work includes capabilities, fancy type systems, formal methods, software engineering (all they did was have interfaces and count the number of lines), RPC (which failed because the idea was to try to mask the fact that the call was remote distributed computing), persistent objects (in which you end up storing a bunch of rubble, because of program bugs), and security (getting worse because there is a lot more software now; also, people don't like security, because security says no but people want to say yes), RISC (Intel retrofit the good ideas of RISC into their chips).

Things that may have worked include parallelism (which now we actually need because we have multi-core systems, but many programmers don't know how to apply the theory, so we probably can't make it work), garbage collection (which was not designed to be used by systems), interface and specifications (with substantial overhead in breaking down the system and specifying the interfaces, they are slightly successful in hardware but not in software), and reusable components (which [1] are expensive to develop, [2] are specific to how resources are allocated and have unique failure models, [3] have been successful in filters and big things [e.g., OSes, DBs, browsers]). Reusable components have not worked for Ole/COM/Web services; however, they have worked for companies such as Amazon, who can afford to have 20% of the things displayed wrong.

Systems research has failed at times, the classic case being that we didn't invent the Web. This is mainly because of the way we think. For example, we felt that the design and the idea of the Web are too simple. The idea of the Web had been around for some time but was never tried. Computer scientists would have tried too hard to come up with an optimal design. Another reason for the failure is that computer scientists tend to deny that things might work. For example, in the case of the Web, they would have just argued that it would never scale.

The future of systems research involves building systems that deal with uncertainty and that avoid catastrophe (e.g., reducing highway traffic deaths to zero). The problem involves computer vision; building world models for roads and vehicles; dealing with uncertainty about sensor inputs, vehicle performance, and a changing environment; and, finally, dependability. Butler defines a dependable system as one that avoids catastrophes. This ensures that the focus is on the really important and provides a way to reduce aspirations for a system. Catastrophe prevention has not always worked; for example, air traffic control specifications state that the downtime should be 3 seconds/year/workstation. But this is not true. The architecture of the system should have a normal mode and a catastrophe mode. The catastrophe mode should have clear, limited goals, implying limited functionality, have <50K lines, and have high assurance. Another issue is dealing with uncertainty. Any "natural" user interface should make assumptions. For example, a speech-understanding program will get some unknown or uncertain input that the computer has to approximate. So one way to deal with this may be to build paradigms where distribution is a standard data type and can be parameterized over a domain (like lists).

Peter Honeyman asked the first question. Is it right to attribute the World Wide Web to physicists? Wasn't Mosiac developed by computer scientists? Butler: Could be I oversimplified. Question: Why is distributed computing a failure? Don't we have the Web? Butler: We don't do distributed computing. We do client-server, where only two machines are talking to each other. Grid? I don't understand it. Margo Seltzer: IBAL is a language that supports probability as a fundamental datatype. I encourage everyone to try it. Margo Seltzer: It looks like catastrophe code is similar to recovery code as it is never run. Butler: Catastrophe code should be a subset of normal code and shouldn't be used only in catastrophes. Marc Chiarini: Is AI a success or a failure? Butler: Yes, it is successful. When it is successful, it's spun off, for example, computer vision. AI continues to be a success and continues to be a mess. Question: Are not large-scale bank computer crashes computer-only catastrophes? Butler: Not true; although it will inconvenience a lot of people, there is enormous redundancy that will get things back to normal. Question: RISC is a success, since most game systems run on it. Butler: There has not been a successful RISC system since then.

PLENARY SESSION

Why Mr. Incredible and Buzz Lightyear Need Better Tools: Pixar and Software Development

Greg Brandeau, Vice President of Technology, Pixar Animation Studios

Summarized by Scott Michael Koch

The talk began by explaining the process involved in creating a movie at Pixar. Using examples from their latest movie, *Cars,* and past movies such as *Monsters Inc.*, he explained the key steps involved in turning an idea for a movie portrayed in storyboard drawings into a detailed, computer-rendered movie. Although all of Pixar's movies are essentially cartoons, the company feels it is important for its movies to contain lifelike effects. Special attention is paid to detail when creating the environments in which their stories take place, by taking into account effects such as weather and fire. When appropriate, Pixar tries to avoid characters appearing to have a plastic texture by giving them fur or other more detailed textures. The next part of the talk began with the showing of a trailer for *Cars* and an explanation of how it compared to some of the movies Pixar had done in the past. Their movies typically take three to five years to complete, and although *Cars* took about the same amount of time to complete as their earlier *Toy Story*, it required 300 times the computing power. He explained the basics of various lighting effects, such as irradiance, ambient occlusion, and reflection, that were used to improve the realistic characteristics of the cars. He also showed a demo of an in-house tool called the Cars Driving System that simulated the movement and interaction of the cars with their environment, so that the animators did not have to worry about underlying cartoon physics such as the car's suspension, steering, and turning.

He talked about some of the challenges they encountered in creating *Cars*, as well as some of Pixar's other movies. Besides the basic process mentioned here, each movie is custom-made. Each has a different director, environment, characters, and technology. Using a technique called Reyes Rendering, the memory space of a 32-bit architecture machine was not enough, so the company were forced to switch to 64-bit machines when rendering *Cars*. In fact, a single car required more than 2 GB of memory. Overall, it required 2.4 CPU millennium to render the movie.

Along with using commercial applications, third-party libraries, and other in-house applications, the majority of Pixar's work is done with a more than 2-million-line in-house application that has been developed over the past 20 years. The application is written in a number of different languages including C++, C, Python, Perl, and sh. The application is constantly being customized to meet the ever-changing needs of the current movie. To take advantage of the best tools at any given time, Pixar feels it is important to keep their software as cross-platform as possible.

A perceived major problem is that Linux/OSS development has not kept up with the innovation of hardware. Having had mixed results with gdb and purify, they felt there needed to be a better debugging utility geared toward larger applications. Using current debugger solutions, a process that usually takes several hours turns into a weekend-long process when run under a debugging environment. They would like OSS developers to design software with large production applications, long run times, OpenGL, and 64-bit technology in mind. They also would like to see a Visual Studio–type IDE for Linux. They also mentioned wanting vendors to provide a sitewide license for software, to make management of licenses for a large number of machines less complicated.

The talk got a mixed response as far as audience questions were concerned. Several attendees from other large companies attested to the fact that the problems and challenges mentioned were not exclusive to Pixar. Others questioned Pixar's contributions back to the open source community. Although they are active in submitting bug reports and patches to projects they use, some thought that they need to be the ones taking the initiative to start solving these problems in the community, and others will join them if they see the project to be worthwhile. There were also suggestions about making all or portions of Pixar code open source in various ways, but the company does not feel that would be appropriate for their type of software. There were also a few suggestion about using Solaris's dtrace, which is something they are considering.

WIDE AREA DISTRIBUTED SYSTEMS

Summarized by Wei Huang

Service Placement in a Shared Wide-Area Platform

David Oppenheimer, University of California, San Diego; Brent Chun, Arched Rock Corporation; David Patterson, University of California, Berkeley; Alex C. Snoeren and Amin Vahdat, University of California, San Diego

David Oppenheimer's talk tried to answer one question: Can intelligent service placement be

useful on a shared wide-area platform such as PlanetLab? At the beginning of his talk, David laid out five perspectives, from which they will analyze the resource characteristics of shared wide-area platforms and try to answer this question: the variability in resource competitions across nodes; the variability in resource demands across slivers (allocated resources on a single node for an application); how random placement behaves; how the quality of initial resource mappings decay over time; and whether resource competition can be predicted.

David presented their studies on these five aspects from a six-month trace of node-, network-, and application-level measurements of PlanetLab. They found out that CPU and network resource usages are highly variable across the nodes. And the resource demands across instances of applications also varied widely. These trends suggested that an intelligent service placement will benefit applications. This was demonstrated by David's simulation results on running OpenDHT, Coral, and CoDeeN, which showed that there were more slivers satisfying application resource requirements by using intelligent service placement. David also pointed out that node placement decisions can be ill-suited after about 30 minutes, which suggested that migration may help applications if the cost is acceptable. David also indicated that a node's CPU and bandwidth usage can be predicted by its utilization of that resource recently, which implies that a migration service need not require high measurement update rate. However, David said that they found no daily or weekly periodicity on resource utilization.

- ***Replay Debugging for Distributed Applications***

  Dennis Geels, Gautam Altekar, Scott Shenker, and Ion Stoica, University of California, Berkeley

  Awarded Best Paper!

  Dennis Geels presented liblog, a debugging tool for distributed applications. Dennis started his talk with challenges of the debugging process in distributed applications. Many errors are due to race conditions and usually are impossible to reproduce locally. Because of this "limited visibility," testing or simulation is usually not sufficient to reproduce and catch the errors. The current state-of-the-art technique for debugging is still to use the print statement. However, once the software is deployed, this technique requires that the developer choose to expose the affected internal state before the fault manifests.

  To address the difficulties of debugging distributed applications, Dennis proposed liblog, which provides lightweight logging and deterministic replay, is transparent to applications, and requires no patch to kernels. It intercepts all libc calls and logs all sending/incoming messages. Each message is associated with a lamport clock so that it can be used later for deterministic replay. Dennis discussed several key challenges and design choices of liblog. He talked about how to deal with concurrent threads, where deterministic replay was harder owing to the lack of kernel support. He also mentioned how to do user-level annotation for TCP traffic and using liblog in a mixed environment of logging and nonlogging processes.

  In the Q&A session, when asked whether there are any success/failure cases,, Dennis briefly mentioned their experience using liblog on I3/Chord and OCALA proxy. He said that liblog helped to find errors caused by broken assumptions about network or coding errors.

- ***Loose Synchronization for Large-Scale Networked Systems***

  Jeannie Albrecht, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat, University of California, San Diego

  Jeannie Albrecht started by addressing the inadequacy of current barrier semantics in large-scale distributed heterogeneous computing environments. She argued that the current barrier semantics is too strict to be effective for emerging applications. For example, network links may be unreliable and machines may become unresponsive. A traditional barrier may lead to the situation where progress is limited by the slowest participant or where one must wait for an indefinite time for failed hosts.

  Jeannie proposed several possible relaxations of strict barrier synchronization (or partial barrier), which are designed to enhance liveness in loosely coupled networked systems. She proposed two partial barrier semantics: early entry, which allows nodes to pass through without waiting for certain slow participants, to prevent a few nodes from slowing down the whole process; and throttle release, which releases the barrier participants within a certain interval, to avoid resource overload by preventing all processes from simultaneously coming into the critical section. Jeannie also talked about several heuristics to dynamically choose the parameters used in partial barriers, such as detecting the knee of the curve (at which point the arrivals are considered to be slow) and finding the optimal capacity of the critical section.

Jeannie presented their experience in adapting wide-area services by using partial barriers for synchronization. She showed promising results. For instance, using a semaphore barrier (a special variation of a throttle barrier) to perform admission control for parallel software installation in Plush enabled an overall completion rate close to the optimal value achievable by manual tuning.

Jeannie was asked about the flexibility of their partial barrier schemes. She answered that the schemes should be very flexible, since applications receive callbacks when the events happen, (i.e., when some nodes are detected to be slow). The applications still have control of the progress and thus have the flexibility to make the best decisions.

INVITED TALK

Routing Without Tears, Bridging Without Danger

Radia Perlman, Sun Microsystems Laboratories

Summarized by Chris Small

Bridges, being at some level simpler than routers (at least, requiring less configuration), are often thought to have come first. Actually, routers came first, bridges came later. And it's a myth that bridges are simpler:

Layer 1 relay = repeater
Layer 2 relay = bridge
Layer 3 relay = router

Wait, this doesn't make sense: layer 2 is defined as neighbor-to-neighbor.

We'll see why this makes sense later. Ethernet is a misnomer: It's not a network, it's a multi-access link. Layer 2 is flat, no topology. If we need to connect two networks of machines connected using a layer 2 protocol, we have no topology information. (If they were connected with layer 3, we could use a router.) So we use bridges. (Switches came from a different direction, but they ended up being the same thing as bridges through parallel evolution.)

The basic idea is that bridges listen promiscuously, learn who is on each side of the bridge, and only forward packets between the two networks as appropriate. But loops (cycles) are a disaster, since layer 2 has neither hop counts nor topology, so you get exponential proliferation of packets. (See Radia's book for the detailed story.)

Let's compute a spanning tree (i.e., subset the graph to make a tree and do not include cycles), only transmit along the links that are in the spanning tree, and save the other links for backups.

On bridges, the spanning tree algorithm turns links on when it thinks the primary links are dead—so if you drop packets, the spanning tree gets turned back into a general graph. On routers, if you drop packets, the link gets shut down. Now that everyone has converged on IP (i.e., everybody is using the same layer 3 protocol), why use bridges at all? Why not routers? Well, bridges are simpler to configure—self-configuring, even.

With link state routing, you discover who you are connected to and broadcast this to your neighbors. Everybody collects this information and forwards it on. Eventually everybody has full information about the entire network.

There is a solution to the bridge/spanning tree problem, called RBridges. They can replace bridges and are safer. Basically they are bridges that gather global link state information. Each RBridge builds its own spanning tree, which is more robust.

Then each RBridge encapsulates packets to tunnel across the network to other RBridges. Add a layer 2 header at each RBridge, with destination address set to the last RBridge.

To fit this into MPLS, we needed to map a 6-byte MAC addr into 19 bits. The trick is to use a nickname (mapping 6-byte MAC addrs to 19-bit nicknames).

Q. What about overflowing maximum packet size?

A. This turns out not to be a problem in practice. The original max packet size was set because the first Ethernet had a very limited amount of RAM, so the max packet size was set where it is. Everybody can handle larger packets these days.

NETWORK AND OPERATING SYSTEM SUPPORT

Summarized by Aameek Singh

System- and Application-level Support for Runtime Hardware Reconfiguration on SoC Platforms

D. Syrivelis and S. Lalis, University of Thessaly, Hellas

Dimitris Syrivelis presented this paper describing an approach to enable programs running on a reconfigurable System-on-Chip (SoC) to modify the underlying Field-Programmable Gate Array (FPGA) behavior at runtime. Accomplishing this requires support from both the underlying system and the application running on it. The reconfiguration is achieved using a quick suspend-resume mechanism, in which the FPGA bitstream corresponding to the new hardware layout is stored in external memory; the system saves its current runtime state and initiates its FPGA reprogramming (i.e., the entire FPGA is reprogrammed from

scratch, as opposed to dynamic partial reconfiguration). After the reconfiguration, the system restarts and manages all effects and potential side effects of the operation. Such reconfigurable systems can offer significant advantages over systems that have soft-core CPUs, drivers, or controllers. Changing the runtime characteristics of underlying units can help applications adapt to different requirements and boost overall performance, though a number of issues such as device addressing need to be resolved. The applications interact with the reconfigurable system through a library that issues device addition/removal requests. The paper also discusses two sample applications of a Mandelbrot calculation and an audio signal monitor.

- ***Resilient Connections for SSH and TLS***

  Teemu Koponen, Helsinki Institute for Information Technology; Pasi Eronen, Nokia Research Center; Mikko Särelä, Helsinki University of Technology

  Teemu Koponen presented this paper, which addresses a common concern of SSH/TLS connections dropping owing to a network outage or travel. The SSH and TLS protocols are extended to provide more resilient connections that can withstand changes in IP addresses and long disconnections. The authors argue that such mobility issues are best handled at a higher session layer as opposed to the data link or network layers, the traditional approaches employed in wireless handover or mobile IP mechanisms. This is especially true in the presence of long disconnection periods and absence of any network infrastructure such as mobile IP home agents. The proposed protocol extensions are made while ensuring that they do not require any network changes or middleware boxes, can be deployed incrementally, and minimize end-point changes. The extensions use in-band signaling (thus easing deployability), negotiations (for an end-point to agree to use the extension—a support that is already present in SSH and TLS), and authentication of reconnection (to prevent hijacking). One unanswered question is the security analysis of these extensions. The authors believe that the extensions do not introduce any new vulnerabilities, but a formal evaluation has yet to be made.

- ***Structured and Unstructured Overlays under the Microscope: A Measurement-based View of Two P2P Systems That People Use***

  Y. Qiao and F. Bustamante, Northwestern University

  Fabián E. Bustamante presented a measurement-based study of two file-sharing peer-to-peer systems based on unstructured (Gnutella-based) and structured (Distributed Hash Table [DHT]–based Overnet system) topologies. The unstructured systems do not dictate the topology of the network, and thus are thought to be more resilient to peer churn (peers joining/leaving the network). In contrast, the structured systems offer guaranteed and scalable O(log *N*) lookup performance (where *N* is the number of peers).

  Based on observations, the authors conclude that both systems are efficient in handing churn; even the Overnet DHT-based system was surprisingly efficient. Both systems had good performance for exact-match (precisely matching an object) queries of popular objects, but Overnet had almost twice the success rate for querying shared objects. Keyword searching was fast in both systems, and load balancing was better handled by Overnet.

INVITED TALK

- ***An Introduction to Software Radio***

  Eric Blossom, Blossom Research; Coordinator and Maintainer of the GNU Radio Project

  Summarized by Rik Farrow

  Software radio means using code to modulate/demodulate radio signals by using as little hardware as possible. Instead of soldering parts, you change the code that is controlling the software radio, providing extreme flexibility, on-the-fly reconfiguration, the ability to act as multiple radios simultaneously, and a much quicker development cycle. Software radio is currently used by the military, SIGINT, research, and cellular companies. Another potential use would be public safety, where interoperability of radios has been a problem (recall the Katrina disaster relief fiasco).

  Blossom introduced some basic concepts required for building any radio transceiver, with the focus on doing this as much in software as possible. Radio waves range from kilohertz into the gigahertz frequencies. To properly digitize any signal, you must sample it according to Nyquist's rule, at at least twice the bandwidth. That sampling is done in hardware using analog to digital converters (ADC), sampling rates as high as 6 GHz, and sample sizes ranging from 8 to 24 bits. Think about that for a moment. If you sample at 6 GHz and 16 bits, we are talking about 12 GB/s. You won't be doing this on your desktop system soon. But researchers have recorded HDTV signals and stored them to disk, requiring a disk storage capacity of 40 MB/s.

  Not all radio requires such high sampling rates (for example, FM radio), and there are projects at GNU Radio (www.gnu.org/software/gnuradio) for an FM

receiver and a 1 Mb/s data transceiver. Blossom said that you can buy hardware today that includes four ADC pairs, an FPLA (for onboard computations, programmed using GNU radio), up to four daughter cards that include analog parts for filtering signals from antennas, and a connection via USB to your desktop system. Using this setup and a 2x2 phased array antenna 1.5 m on a side, Blossom has created software that can track aircraft using the signal from an FM radio station antenna on a mountaintop near his home in Nevada. There are regulatory issues when building your own radio transmitters, but these can be dealt with by using certain frequencies and signal strengths. Blossom called the FCC a bunch of politicians, lawyers, economists, and engineers who regulate bandwidth as if radio were stuck in the 1920s. A single VHF TV channel wastes 6 MHz of bandwidth, for example, and compulsory channels use more than all the bandwidth used by cellular channels. Creative use of software radios would make much better use of bandwidth without causing interference with other forms of radio communication.

GNU Radio uses data flow abstractions, event-based overlay, message queues, and messages, all written in a hybrid of C++ and Python. The software is free and the hardware now costs under $1,000 (www.ettus.com). You can learn more at http://comsec.com/wiki.

CLOSING SESSION

- ***Real Operating Systems for Real-time Motion Control***

  Trevor Blackwell, CTO, Anybots

  Summarized by Marc Chiarini

  Trevor Blackwell gave an interesting and entertaining presentation about his experiences building robots and other devices that are controlled by humans in real time. Some useful areas for these include performing dangerous tasks (bomb squad and underwater salvage), avoiding long-term travel (Mars rover), and, perhaps somewhat controversially, supplying efficient on-demand manual labor (e.g., someone half a world away does your household chores, much to the delight of homebodies and eight-year-olds!).

  The bulk of the talk comprised four parts: fundamentals of robotics and control, software platforms and components, levels of abstraction for controlling robotic devices, and a discussion of his construction of self-balancing motorized vehicles. For the first part, Blackwell quickly took the audience through a primer on a spectrum of computing components and sensors, from large to small, that serve different purposes and are placed on different sections of robots. The joints on humanoid robots are primarily pneumatic and are actuated by software-controlled proportional valves. Human control of the robots he builds, such as those for experimenting with bipedal motion, utilize common sensors in gloves and cameras that provide constant feedback on hand and/or arm position. Several videos comically demonstrated the difficulty of real-time robot control, particularly when lag was involved (even human sensory lag).

  Blackwell moved on to show a breakdown of his component-based heterogeneous infrastructure for robotic experimentation. He starts with a rack of BSD servers responsible for performing complex motion vector and other computations (mostly programmed in Python). These are connected via wired (or wireless) TCP/IP to embedded CPUs running UNIX and mounted at various points on the robots. The CPUs communicate with a set of microcontrollers that drive the actual hardware, valves, etc. By tinkering with several parameters in the embedded FreeBSD kernel, it is possible to achieve millisecond response times when coordinating controllers. Timing is very important to this task, because even a slight lag in actuation can result in, for example, a robot losing balance, running into a wall, or crushing an object. This led naturally into a discussion about the levels of abstraction for motion control: actuator, position control, position control with feedback, high-level, and fully autonomous. A graph gave the audience a good idea of what could be effectively controlled by a human (given human tolerances) or software at each level: Using just actuators and position control, a device at the level of a Roomba vacuum cleaner is achievable; with feedback, unmanned aircraft or an arm on wheels can be controlled; high-level computations might permit reasonable bipedal motion, but a fully humanoid robot would require a high degree of autonomous control. There are, of course, cracks in this picture and not every device falls neatly into a single category.

  The last part of the presentation focused on Blackwell's originating hobby of building self-balancing vehicles such as his Eunicycle and Segway-like scooter. Most of it turns out not to be rocket science, but it still requires a reasonable knowledge of mechanical engineering and classical physics. There were several poignant questions asked during the Q&A: Is building these things an affordable endeavor for hobbyists? Scooters and such are definitely reasonable. Humanoid robotics, especially smaller projects, are quick-

ly becoming an option. Why didn't Blackwell incorporate force-feedback in his projects? Latency is a significant stumbling block, especially for fine control. Why did Blackwell ignore other biologically inspired nonhuman robot designs? The response was that he was most interested in robots that could do tasks designed for people in environments designed for humanoids. See http://anybots.com and http://tlb.org/scooter.html for further details.