conference reports

- This issue's reports focus on the USENIX Annual Technical Conference (USENIX '04), held in Boston, Massachusetts, June 27–July 2, 2004.
- Our thanks to the scribe coordinator:

  Rik Farrow
- Our thanks to the summarizers:

  Bill Bogstad
  Ming Chow
  Brian Cornell
  Richard S. Cox
  Todd Deshane
  Patty Jablonski
  Rob Martin
  Martin Michlmay
  Adam S. Moskowitz
  Peter Nilsson
  G. Jason Peng
  Calicrates Policroniades
  David Reveman
  Matt Salter
  Swaroop Sridhar
  Sudarshan Srinivasan
  Matus Telgarsky
  Wanghong Yuan
  Ningning Zhu

USENIX ANNUAL TECHNICAL CONFERENCE (USENIX '04)

Boston, Massachusetts
June 27–July 2, 2004

PLENARY SESSION

Summarized by Richard S. Cox

Open Source and Proprietary Software: A Blending of Cultures

Alan Nugent, Novell

Alan Nugent opened the USENIX Annual Technical Conference with his plenary session addressing the integration of open source software and procedures at Novell.

Many people believe that open source will destroy the software industry; that it is developed by hackers without discipline; that it is a fad; or that there is no money in open source. Seeking to debunk these myths, Alan first suggested that, rather than wrecking the industry, open source has increased diversity and thus has created opportunities. Second, open source software can be of very high quality, since a majority of open source contributors are professional developers working on projects that interest them. The community is growing daily, and contributors are quick to realize important initiatives. While open source software is free, there is a market for selling the support and maintenance contracts that large customers require before they are willing to build mission-critical systems using a package.

The adoption of open source has allowed Novell to work with customers to build solutions that more closely match their needs and infrastructure. Novell does this by providing complementary packages (open or closed) that interact with those developed by the open source community. By focusing on complementing existing projects, rather than providing substitutes, they avoid competing with open source developers, an arrangement that benefits all involved.

At Novell, this has required reworking the legal framework under which licenses are sold, expending significant effort in convincing customers to accept solutions combining proprietary and open components, and changing the focus of the organization.

Greg Mitchell asked how the sociology of the company changed as more open source developers were brought in. Alan responded that, while some employees were upset and a few even left, the acquisition of open source teams has been very successful and brought more energy throughout the company. Novell was able to do very well retaining employees from acquired companies.

GENERAL SESSION PAPERS: INSTRUMENTATION AND DEBUGGING

Summarized by Swaroop Sridhar

Making the "Box" Transparent: System Call Performance as a First-Class Result

Yaoping Ruan and Vivek Pai, Princeton University

Mr. Yaoping Ruan presented the "DeBox"ing technique for debugging OS-intensive applications. He began the talk with a motivating example of monitoring system call performance on a server running the SpecWeb99 benchmark. He pointed out that system call profile as measured from user space sometimes indicated anomalous kernel behavior. He identified the trade-off between speed, completeness, and accuracy among various profiling tools. Later, Ruan presented the

design of the DeBox system. The key idea is to make the system call performance a first-class result and return it in-band (like errno). Proposing a split between the measurement policy and mechanism, Ruan said that the applications should be able to interactively profile interesting events.

Later, Ruan gave details about the implementation of DeBox. He gave the details of profiling primitives added to the kernel and the interface available to the applications. He also provided details about the various kinds of information that the system offered, the amount of change that had to be done to the kernel and applications, and so on. Ruan went on to present a case study on Flash Web server performance. He presented various optimizations with a step-by-step performance analysis.

Ruan concluded by stating that DeBox is very effective on OS-intensive applications and complex workloads. He also claimed that the results showed that the system was portable. During the Q&A session, Ruan said that they were investigating the use of DeBox on other OS-intensive applications such as database systems, but the results were not yet available. More information about DeBox can be found at http://www.cs.princeton.edu/~yruan/debox, or by contacting {yruan, vivek}@cs.princeton.edu.

Dynamic Instrumentation of Production Systems

Bryan M. Cantrill, Michael W. Shapiro, and Adam H. Leventhal, Sun Microsystems

In introducing Bryan Cantrill, session chair Val Henson—also from Sun Microsystems—said that she could definitely confirm Sun's use of DTrace in production. Cantrill began his power-packed speech by stating that all of today's tools were targeted at development and not production. As a result, the systems are incapable of dealing with systemic problems. Cantrill asserted that for a tool to be used in production, the necessary constraints are that there should be *zero* probe effect when disabled, and the system must be absolutely safe. To have systemic scope, both kernel and applications must be instrumentable, and the system must be able to prune and coalesce the enormous amount of data into useful information.

Later, Cantrill introduced the various concepts and features of DTrace: dynamic-only instrumentation, unified instrumentation, arbitrary context kernel instrumentation, high-level control language, predicate and arbitrary action specification, data-integrity constraints, facility for user-defined variables, data aggregation, speculative tracing, scripting capacity, boot-time tracing, virtualized consumers, etc. Next, Cantrill elaborated on the D language: syntax and use, D intermediate form, probes, providers and actions, aggregations and scalability of the architecture. Cantrill also shared some experiences with DTrace and gave some examples of D scripts and analyzed their results. Finally, using the example of a bug in gtik2 applet2—a stock ticker for GNOME desktop—he showed how a small programmer error could cause widespread damage in a production system such as SunRay server. Cantrill challenged the idea that no other existing tool could trace this problem to its root cause, and that a trace was possible only by the extensive use of aggregation functions and thread local variables provided by DTrace.

During the Q&A session, Jonathan Shapiro said that he believed that the gtik2 applet2 problem should be attributed to the fundamental problems in monolithic kernel design, and asked the speaker to comment on the use of DTrace for debugging kernel bugs. Cantrill did not totally agree with Shapiro's views, but only asserted that DTrace was effective in tracing kernel-level bugs. Answering another question, Cantrill said that there was no extra effort required to use this tool with third-party kernel-level modules. When asked whether there were any plans to port their system to Linux or any other operating system, Cantrill answered in the negative and quipped, "Use the best OS available!" The authors can be contacted at dtrace-core@kiowa.eng.sum.com.

Flashback: A Lightweight Extension for Rollback and Deterministic Replay for Software Debugging

Sudarshan M. Srinivasan, Srikanth Kandula, Christopher R. Andrews, and Yuanyuan Zhou, University of Illinois, Urbana-Champaign

With the increase in volume and complexity of software development, there is a proportional increase in software bugs, their effects, and the difficulty in tracing or even reproducing them. Various checkpointing and logging mechanisms and their applications have received a lot of research attention in the last decade. Mr. Sudarshan Srinivasan presented Flashback, a lightweight OS extension to facilitate rollback and replay, as applied to software debugging.

After providing a brief general background and motivation for lightweight checkpointing, Srinivasan went straight into the main idea of Flashback. Flashback achieves checkpointing by forking a shadow process, thus replicating the in-memory state of the process. The processes' interactions with the system are logged so that, during replay from a checkpoint, the (shadow) process gets an execution environment similar to the original run. Srinivasan presented some challenges posed due to multi-threading, memory-mapped files, and shared memory and signals. He also presented the approaches adopted in Flashback toward solving these problems.

Srinivasan went on to present some details of the present Linux imple-

mentation regarding modifications to the kernel, changes to GDB, etc. Srinivasan identified incorporating replay support for multi-threaded applications as an area for future work. Later, responding to Val Henson's question regarding possible applications of Flashback other than debugging, Srinivasan said they were investigating uses of Flashback in other avenues, such as lightweight transaction models. The source code for Flashback can be obtained at http://carmen.cs.uiuc.edu/.

ADVANCED SYSTEM ADMINISTRATION SIG: AUTOMATING SYSTEM AND STORAGE CONFIGURATION

Summarized by Rob Martin

The CHAMPS System: A Schedule-Optimized Change Manager

Alexander Keller, IBM T.J. Watson Research Center

Dr. Alexander Keller began by describing CHAMPS (Change Management with Planning and Scheduling) as "a schedule optimized change management system" that is not yet a product. "It's a research prototype [providing] change management with planning and scheduling." Its end product is the schedule: "all the things that are going to be carried out on which machines [and] concrete systems that are going to carry out these tasks." Keller described this as "a change plan."

Keller described CHAMPS within the larger context of change management as "trying to assess the impact of a change and figure out what the dependencies between different tasks are and creating a change plan. . . . We are specifically not concerned with actually implementing or rolling out a change, because there are deployment systems that can do this." Later in the talk, Keller gave examples of such systems: cfengine and Tivoli Intelligent Orchestrator for Service Optimization.

The CHAMPS system consists of two subcomponents: the Task Graph Builder and the Planner and Scheduler. The end product of the system is a change plan depicted in a standard workflow language (BPEL4WS). This, in turn, is fed into an "off-the-shelf" deployment system which "rolls out the changes and provides feedback status information back into the [CHAMPS] system for summary planning." The workflow engine executes the plan and monitors whether each activity has completed or failed.

A key goal of CHAMPS is optimizing the schedule based on dependencies to carry out tasks in parallel wherever possible. The information used to figure out which tasks are going to be carried out in sequence and which in parallel are "product dependency descriptions." "The availability of authoritative dependency information [from package developers] is very important." Once the dependencies are put into the Task Graph Builder, the system generates the Task Graph.

"The Task Graph tells us which tasks are going to be carried out, in what order . . . , and whether they must be in sequence or can be in parallel." The Task Graph is used as input to the Planner and Scheduler. "The Planning system may make decisions such as 'we must take away a machine from customer X and give it to customer Y'; in order to do that [the system] must be aware of the service level agreements and policies that the data center has. . . . It is up to the planning system to bind the existing Task Graph to the complete system to generate concrete system names, times, and dates.

"We put in declarative information about the relationships between tasks, [and the CHAMPS system] automatically generates this schedule and allows the administrator to apply modifications to the schedule." Estimating individual task duration is crucial, and CHAMPS uses "past deployments" to calculate future durations for individual tasks.

Multiple task graphs, each representing a single change, are input into the Planner and Scheduler, which then binds the changes to services and resources and optimizes a schedule for all of the changes. "We are treating this problem as an optimization problem." The optimization is done by "fifty pages of Java . . . not visible to the administrator. We support a very general level of objective functions [for] minimizing penalties, maximizing profits. The administrator selects from push-button options that provide choices like 'maximize profits,' 'minimize downtime,' 'maximize throughput,' 'minimize costs.' By selecting one [or a combination] of these choices the optimization parameters are automatically set." The CHAMPS system then calculates the optimum schedule, if necessary deciding that certain changes cannot be accomplished given the overall set of changes requested.

Keller concluded by listing the areas that require more work in the future, including "tooling for deployment descriptors," reusing change plans (storing them in an XML library, for example), knowing when a plan is running behind schedule, carrying configuration information along with the workflows, and identifying parameters that flow out of one task and are required for other downstream tasks.

During the Q&A session, there was a lively exchange on the "sad state of dependencies in software packages." Is there a standard for describing dependencies? Work done by the Grid Forum on defining a standard, and the use of dependency sniffing tools were mentioned.

Autonomics in System Configuration

Paul Anderson, University of Edinburgh

What is system configuration? Paul Anderson, professor and researcher at the University of Edinburgh, starts out with some background on the general subject. When you want to build a new site, you start off with three things: the hardware (empty disks and bare metal), the software, and specifications and policies about how you want the final system to run. The core of the configuration problem is to take those three things and put them together to get some sort of computer system that performs to the specification. Anderson refers to the final site as a "fabric," a term he borrows from the recent work in grid computing.

Anderson points out that the "main thing to notice is the big distinction between the software and the configuration policies. The pile of software you start off with has no configuration and in theory can all be put on all the machines that you've got. It's the specifications and the configuration policies that differentiate the individual machines."

Configuration starts with the base layer of internal services inside your fabric at a lower level than the applications you want to end up with. DNS, NFS, DHCP, and like services form a base layer you have to get going before you build anything on top of it.

The idea of autonomics is to "take some of the low-level decision making away from the system administrator and have a lot of things happen automatically, so the system administrator can move up a level and think of higher-level policies and planning." As with a compiler, you trust the autonomic system to place low-level data and decide which bits go where.

After the initial configuration, as change occurs due to load balancing, software or hardware failure, and day-to-day use, the autonomic system adjusts the fabric and configuration of the system so that it comes back into alignment with the original specifications and policies. The feedback from the autonomic system does not make changes to the original specification; rather, it brings the fabric back to providing the original services and policies specified.

"Autonomics is not new. Cfengine and lcfg are examples of tools that provide this sort of automatic fixing up of configuration files when something goes wrong at the host level. There are inter-host autonomic systems like fault-tolerance systems, RAID, and load balancing that will adjust systems. What is new is trying to think of this in a uniform way and integrating it into the configuration system."

Anderson described the major issues under consideration in researching autonomic solutions. He described "a declarative specification of what the system behavior should look like. Some kind of logical statement that is true about the system rather than a recipe about how to get there. If you don't have a good declarative statement to start with, then you don't know what to do. . . . The language you need to describe the configuration is not a programming language: We are not talking about a process, we are talking about the description of the configuration data and the way that that system actually is." Anderson gives an example of a declarative statement. "'Host X uses host M as a mail server.' In most configuration systems you don't see statements like these. Rather, you see lower-level details, like a script setting parameters for sendmail.cf."

The goal is to use the declarative language to describe the system and "let the autonomic system juggle the details" to make sure the specifications remain true.

An example declaration: "Make sure we have two DHCP servers on each network segment." This expresses a high-level policy rather than details like "make this machine configured as a DHCP server." The final goal of an autonomic system is to take these declarative statements and generate the details. "System administrators will specify those criteria that are important for the job without specifying the details. The important point is not to specify too much detail, because you need to give the autonomic system room to move." If something breaks, the autonomic system needs flexibility in order to fix the problem.

Autonomic systems require a lot of trust in the system. The system automatically makes some serious decisions for you. "System administrators are not normally happy giving that kind of freedom to the system. You want the system to decide things for you but you want to be able to review them and adjust them to make sure they are right." Autonomic systems will need to provide feedback as to why something has happened. "You should be able to ask the system, 'Why have you put that there?'" Some mechanism for reviewing system actions and tuning the policy implementation for future actions needs to be provided.

What Anderson is seeking is a compromise between the two extremes of, on the one hand, a complete expert system that can be given high-level policy goals and perform all reasoning and logic decisions, generating all individual assignments for all machines and services, and a solution that is based on hand-crafting (or scripting) the low-level specifications for machines and services required to deliver the specified policy. What we want is some autonomics but not a completely unpredictable system.

"The autonomic system has to be able to change all aspects of a system configuration dynamically. UNIX was never designed to be re-

configured on the fly." UNIX has all sorts of config files in all sorts of formats; services may need to be stopped and re-started in order to make certain changes. "This is a big problem in incorporating autonomics into system configuration."

Anderson concluded by reviewing the lcfg system, analyzing where it has useful autonomic capabilities and where it falls short. He pointed to the http://www.lcfg.org Web site and the LISA '03 Gridweaver paper for those who want to explore the complete details.

GENERAL SESSION PAPERS: SWIMMING IN A SEA OF DATA

Summarized by G. Jason Peng and Wanghong Yuan

Email Prioritization: Reducing Delays on Legitimate Mail Caused by Junk Mail

Dan Twining, Matthew M. Williamson, Miranda J.F. Mowbray, and Maher Rahmouni, Hewlett-Packard Labs

Matthew Williamson discussed the motivation for this paper. In particular, he described the delay problem caused by junk emails, the distribution of junk mails, and the source of junk mails. Dan Twining then presented the proposed approach, which combines pre-acceptance (header scanning) and post-acceptance (content scanning) to predict the next message type based on sending history. The pre-acceptance method maintains the number of good and total messages and tells if a server is good based on the ratio. The system is implemented in a lightweight manner and shows good results on a real system.

Redundancy Elimination Within Large Collections of Files

Purushottam Kulkarni, University of Massachusetts; Fred Douglis, Jason LaVoie, and John M. Tracey, IBM T.J. Watson Research Center

Storage needs keep growing as per-byte cost gets cheaper. The goal in storage is to increase efficiency by reducing redundancy. Current techniques (compression, duplicate block-and-chunk suppression, and resemblance detection) have shortcomings. Purushottam Kulkarni proposed a technique called Redundancy Elimination at Block Level (REBL), which first detects duplicate chunks and encodes blocks using the resemblance technique. This paper also evaluates five techniques to quantify the effectiveness of REBL.

Alternatives for Detecting Redundancy in Storage Systems Data

Calicrates Policroniades and Ian Pratt, Cambridge University

Calicrates Policroniades introduced the benefits of redundancy elimination and previous techniques for redundancy elimination, and then compared three frequently used techniques: whole-file content hashing (WF), fixed-size blocking (FSB), and content-defined chunks (CDC). The results show that in terms of compression ratio, CDC is the best, FSB is almost as good, and WF is the worst. But when compression, processing overhead, and storage overhead are considered, however, no one solution wins.

ADVANCED SYSTEM ADMINISTRATION SIG: SYSTEM ADMINISTRATION: THE BIG PICTURE

Summarized by Rob Martin

The Technical Big Picture

Alva Couch, Tufts University

Each fall at Tufts University, Professor Alva Couch presents a talk to his students on the Big Picture in system administration. In Couch's words, "Where are we going? What are we going to do? How is it going to work? What is going to be the benefit?" This year at USENIX Tech '04, Professor Couch let us in on a preview of his "technical briefing on next year's big picture" talk for his students.

According to Couch, the future of system administration is about "cost models" and "supporting the enterprise mission." Couch summarized it this way: "Based upon a cost model, we can re-define good system administrating. That idea is rather cosmic, because what we are doing right now, what we consider 'good' right now, I would claim does not make sense with respect to any cost model. . . . Looking at things from a broader perspective of lifecycle costs, we get a better idea of whether we are doing our jobs."

Professor Couch refers to traditional SA thinking and practice as "micro-scale reasoning" and "the bottom-up approach." He includes "adhering to practices, process maturity, six nines at the server, closing tickets quickly, reducing troubleshooting costs" as examples of micro-scale thinking. The opposite of this is the "top-down approach": "In a top-down approach we start at the organization and mission and work down. It turns out that starting at that point and thinking out the whole nature of the profession leads to different conclusions and that's the subject of this talk today."

Professor Couch lists some observations drawn from "macro-scale thinking": "System administration enables particular things. It enables missions. It supports plans. It manages resources. It enforces policies. There is a very high level at which, Burgess says, 'the system administrator manages human computer ecologies.'" Professor Couch says this macro-scale thinking will lead to some sacrilegious ideas. "Six nines for the mission does not require six nines at the servers. . . . There is a fundamental idea that we build six nine infrastructures upon six nine servers and six nine foundations. That is actually not true. Meta-stability is enough. Perceived stability is enough. Security that

compromises mission can be inappropriate and counter-productive. Security is not an end unto itself. It is part of a larger mission picture and availability and security have to be balanced."

Three recent published papers that "got a lot of flak in the last LISA and LISAs before" got Professor Couch thinking this way. The papers were on the cost of downtime (Patterson); the timing of security patches (Beattie et al.); and. resource management without quotas for specific users (Burgess).

Patterson says downtime can be quantified. Professor Couch "cannot believe the resistance that this idea got. Nobody ever talked about cost models before. And maybe because of that this was a very controversial idea."

Couch then talked about the paper that "caused a riot." "Beattie et al. showed that if uptime was important because downtime was expensive, then waiting a couple of weeks to apply a new security patch was probably optimal. . . . The thinking was about a cost model; it was not about simple religion, about just applying things because you are supposed to, but about understanding that patches of a problem in the very first case . . . have problems themselves and that waiting a couple of weeks to apply all of them was a better enterprise strategy.

"Finally, another very bizarre idea: protect useful work instead of limiting people. Burgess actually proposes a game theoretical approach for quotas. The idea of the game is extremely simple. You have a very simple strategy for deleting pigs and you counter every strategy the user could use to defeat you. It uses random scheduling for cleanup to beat the wily user.

"These three examples have common attributes. They consider the broader picture of enabling work and mission rather than fixing systems. They consider costs and values rather than just considering the cost of implementing a change. They also include the cost of not doing things, as well as the cost of doing things."

Couch proposes that we have to change the way we are doing SA: "In pursuing micro-scale perfection we are pursuing a private game, and nobody except us cares. We have to play a different game. . . . Micro-scale system administration as we know it is doomed. The idea of pursuing six nines at the server will be a solved problem in 10 years." He referred to the previous USENIX Tech session on autonomics and said, "We are beginning to understand how to automate installs, automate deployments, and automate monitoring and recovery. In the near future much of this will be automated. Unfortunately, system administration is subject to outsourcing. . . . But we can't . . . automate macro-scale thinking. That's the future of system administration. Being able to take these boxes with six nines and make them talk to each other to support an enterprise mission.

"The future will be about understanding cost and value and how they relate. It's going to be about interacting with middleware. It's not about supporting users; it's about supporting missions."

Professor Couch says the new challenge is to "take the human mission and turn it into something the machine can understand." To do this we will need to research new areas, such as economic models to describe "making day-to-day cost-value decisions." Professor Couch suggests we need to ask, "How does one best quantify the value of mission support? How does SA work impede or aid mission? Is anything you are doing getting in the way of mission and how can you stop?"

He concluded by reminding us what can't be automated and outsourced in SA: "We are here at this conference because one can't outsource community. Outsourcing affects and limits many things. One cannot outsource the value of people being together in one place and thinking about a common problem. That will remain a factor in system administration I think for as long as we live."

GENERAL SESSION PAPERS: NETWORK PERFORMANCE

Summarized by Sudarshan Srinivasan

Monkey See, Monkey Do: A Tool for TCP Tracing and Replaying

Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker, University of California, San Diego; Urs Hölzle and Neal Cardwell, Google

The authors describe Monkey See, Monkey Do (MS-MD), a tool that generates realistic client requests in order to test changes to the back end of Google search engines. Currently, changes to servers are tested by either using synthetic (and consequently unrealistic) workloads or real users, making it risky and effort-consuming. The motivation for developing MS-MD is to overcome the shortcomings of existing approaches of testing.

The tool has two phases of operation—the Monkey See phase, where it observes real network connections, measuring network traffic parameters along the way, and the Monkey Do phase, where it generates realistic workloads based on the previously observed metrics. The recorded parameters include the HTTP header, query parameters, delay ACK policy, and other measurable quantities (such as response time). All tracing is done in front of the server farm, and the authors assume that congestion—i.e., queueing of requests—happens, if at all, only along the data path. They also assume that the Web servers themselves are well provisioned and that there is no congestion in the intranet. Caching

behavior is also recorded and replayed by MS-MD.

They evaluate the tool with respect to two questions: how accurately it reproduces the workload, and how accurately it predicts server performance with changes effected. Results show that the measured times without changes to the kernel match up more or less between the original run and the simulation using MS-MD. The tool is more accurate when the RTTs are small; they ascribe this behavior to the fact that the client emulators are on Linux systems, which have a more aggressive ACK policy than traditional Windows clients. Experiments also show that the tool accurately predicts changes in network behavior when services are changed. The tool works for Google, and the authors contend that it will also be usable in other domains.

A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths

Ming Zhang, Junwen Lai, Larry Peterson, and Randolph Wang, Princeton University; Arvind Krishnamurthy, Yale University

Ming described mTCP, a transport-level network protocol developed by the authors for aggregating the bandwidth of multiple heterogeneous paths between two hosts. Bandwidth aggregation provides the benefits of improved performance compared to individual network connections, while also improving the resilience of the aggregate connection. The main challenges for providing effective bandwidth aggregation are congestion control, congestion sharing, recovery from failed paths, and selecting which paths to use for packets dynamically.

mTCP uses a single send/receive buffer for all connections, along with per-path congestion control. Packets get striped across the various possible links. This leads to a greater chance of packets getting reordered, though along each channel packets are still in order. This generates too many DUP ACKS. The problem of packet reordering is solved by using SACK TCP. mTCP uses an extended scoreboard algorithm to figure out which packets have been received and which are outstanding. Packets are sent in the order in which they are queued, and the choice of channel is based on proportional scheduling.

For handling shared congestion, mTCP drops one or more of the shared connections in the presence of congestion, so that single-channel connections do not suffer at the expense of aggregated connections. Shared connections are detected by studying the correlations between the different fast retransmissions—closely related fast retransmissions between two links point to a shared connection. For path selection, overlay networks are used to create candidate paths from which a subset of paths is selected greedily with the minimum common links between them. The greedy algorithm chooses paths that are most disjoint so that there is minimum interference between the paths in terms of performance impact and the effect of failed links.

Performance measurements show that the throughput of mTCP is more or less cumulative of the individual network throughputs, as it should ideally be. Separate per-path congestion control provides better throughput than combined control. The failure-detection and recovery mechanisms adopted by mTCP work effectively, allowing the network to recover seamlessly from the failure of one or more of the links. Finally, the throughput of the mTCP system is significantly better than individual paths in the presence of congestion.

Multihoming Performance Benefits: An Experimental Evaluation of Practical Enterprise Strategies

Aditya Akella and Srinivasan Seshan, Carnegie Mellon University; Anees Shaikh, IBM T.J. Watson Research Center

Multihoming is a frequently adopted strategy in enterprise networks for improving performance as well as availability of the network. Route controllers are used to provide the required performance and availability characteristics. Aditya presented the results of experiments conducted by the authors to evaluate the performance benefits achievable from commercially available multihoming network solutions. This work extends an earlier study, which showed potential performance improvements of up to 40% compared to no route control for a three-ISP network under ideal route control. Aditya also presented a simple near-optimal greedy route control algorithm.

The route controller needs to monitor performance on the ISP links and choose the best link to send packets through based on the performance of the ISPs at that time. It also needs a way to direct traffic once this choice is made. The authors use EWMA (exponential weighted moving average) to track average performance of each of the ISPs and decide on the best path. While redirecting outbound traffic along a particular ISP's network is trivial, redirecting incoming traffic is a little more difficult. In-bound traffic from within the enterprise is redirected using NAT, and externally originated traffic is handled by modifying the DNS entries.

While monitoring the ISP links' performance, only the traffic to the top Web servers is observed, since they account for most of the network traffic. The actual monitoring can be either passive or active. In passive measurement, the route controller periodically measures the turnaround time (time between the last HTTP request and the first response); this is an estimate of the RTT of the network link. Route controllers doing active monitoring initiate out-of-band probes to get

network performance metrics. Sliding windows and frequency counts are two methods used to decide the frequency of monitoring.

The performance measurements show that even simple passive monitoring of the network connections offers significant performance gains compared to no route control. Active monitoring outperforms passive monitoring, but only by a small margin. The use of history in EWMA does not offer any performance benefit; the current performance of the network connections is a good indicator of future performance. With regard to the effect of the frequency of sampling on performance, results show that a very small sampling interval is harmful to the performance because of frequent changes to NAT and DNS entries, while a too-large sampling interval may lead to stale values being used for the network metrics.

ADVANCED SYSTEM ADMINISTRATION SIG: LARGE STORAGE

Summarized by Adam S. Moskowitz

Autonomic Policy-Based Storage Management

Kaladhar Voruganti, IBM Almaden Research

Kaladhar Voruganti discussed his work on the SMaestro Network Storage Planner, which is part of IBM's autonomic computing effort. Despite advances in technology, storage resources tend to be poorly utilized, it is difficult to map application needs to storage systems based on their capabilities, and the number of system administrators needed to manage storage has not decreased (despite claims of easier configuration and management). Customers are moving to SAN and NAS solutions, but these problems are not going away. Voruganti believes that there are both process and technological aspects to the problems (and their solutions).

There is currently virtually no automated solution to analyzing a customer's environment (with respect to storage needs), no way to translate high-level business goals into technology requirements and policies, no way to plan a storage infrastructure based on these needs and requirements, and no way to monitor the solution to ensure that it does not violate the stated goals and requirements. Voruganti's work is limited to a tool to helping plan the storage infrastructure. He was also careful to note that this tool is intended to make things easier for the architect, not replace him. There are many daunting tasks facing an architect when planning a storage infrastructure: interoperability concerns, best practices to follow, gathering data from multiple sources (all of which use different reporting mechanisms), performing complicated "what-if" analysis, and validating that everything was done correctly. SMaestro is intended to make at least some of this easier.

Typical application requirements include I/Os per second (both average and peak), bandwidth, percentages of random/sequential reads/writes, MTBF, maximum acceptable outage times, encryption, integrity, wrote-once support, recoverability, retention times, scalability, and (of course) cost. All this must be balanced against typical storage system capabilities such as latency, I/O rates, availability, points of failure, "hot upgrade," reliability, and (here it is again) cost. SMaestro uses templates for both categories as well as for virtualization software, backup/restore software, and SAN file systems. Policies are written in plain English and then translated, using data models, into something that can be used with the templates to suggest a storage architecture and to verify that the proposed architecture meets the requirements.

Experiences with Large Storage Environments

Andrew Hume, AT&T Research

In his talk, subtitled "Inside an inode, no one can hear you scream," Andrew Hume discussed his experiences trying to actually deal with large quantities of data. How large? In one case his project collected between 200GB and 700GB of new data every day. Hume was quick to point out that "20TB just doesn't go as far as it used to." Hume also mentioned improvement in compression algorithms, some of which study the data and then routinely deliver ratios well above 90%. [Summarizer's note: Some of these compression programs were written by Dave Korn; when I asked him if they would be released to the public, he said, "We want to, we're trying to, but we're not there yet."]

To process this data, Hume uses no RAID, no SAN or NAS, no distributed filesystems, and no special hardware save for a high-speed low-latency network and suitable host adapters on each node. Instead, data is broken into chunks, shipped to the next available node in the cluster, and then processed locally. In doing this sort of thing, Hume learned that networks and disk controllers are not as reliable as previously thought. To handle these unreported errors, all data is checksummed before and after each transfer, sometimes more than once. Part of Hume's work is trying to be able to prove that files are "correct" to a standard high enough to be accepted in court. In part this is being driven by legislation such as the Sarbanes-Oxley Act.

Hume claims that backup systems are getting worse; they now have to buy new backup hardware about every three years. His project currently uses AIT-2, which he finds unacceptably slow. By comparison, computers get faster and more reliable every time he buys them.

In the Q&A session, as a comment on a question from an audience member (to Voruganti), Andrew claimed to be from "the Ken Thompson school of thought on expert systems: there's table look-up, fraud, and grand fraud."

PLENARY SESSION

Summarized by Richard S. Cox

Network Complexity: How Do I Manage All of This?

Eliot Lear, Cisco Systems

Eliot presented an appeal for work on network management. While providers were previously concerned with only a few very expensive devices, we now have a huge number and variety of devices ranging from routers and switches to laptops and phones, all of which need to be monitored and managed in order to support critical network applications.

The first step in managing is to know what is connected to the network. Unfortunately, today's discovery mechanisms won't work for the millions of devices on future networks. Instead, devices will need to "call home" or send notification when their status changes. This presents issues from what to name the device to determining where to send the notifications. In some cases, there may be multiple interested parties: for example, an ISP, a VPN provider, a voice service provider, as well as the customer, may all be interested in updates from an IP phone; managing this while respecting privacy and security is a major challenge.

Having found the network components, we next need to determine their current status. As a bright point, standards for monitoring and state retrieval from individual devices, such as SNMPv3 and syslog, are maturing and investment is being made in tools. Even simple devices now support an SNMP interface. However, dealing with the large amounts of data generated by all these devices is still an unsolved problem. Correlating reports from multiple sources to identify a fault or anomaly is important, both to limit the amount of information presented to a human and to avoid burdening the infrastructure with excessive management traffic. Some support from the networking hardware—for example, programmable aggregation in the routers—might be useful here.

Having determined the state of the network, closing the loop requires the ability to control the devices. Here, standards are less advanced, though a certain amount of that can be attributed to the cutting-edge nature of the field. The NETCONF protocol is an effort to provide a common transport protocol and syntax for configuration, but vendor-independent configuration schemas are still unspecified.

GENERAL SESSION PAPERS: OVERLAYS IN PRACTICE

Summarized by Ningning Zhu

Handling Churn in a DHT

Sean Rhea, Dennis Geels, and John Kubiatowicz, University of California, Berkeley; Timothy Roscoe, Intel Research

■ *Awarded Best Paper*

According to statistics from real P2P networks such as Kazaa, churn ("the continuous process of node arrival and departure") is prevalent in real life. Through experiment on ModelNet—an emulated network—the authors show that several important distributed hash table (DHT) variances (e.g., Tapestry, Chord, and Pastry) all failed to handle churn very efficiently.

This work identifies and explores three factors affecting DHT performance under churn: reactive versus periodic failure recovery, message timeout calculation, and proximity neighbor selection. Results show that periodic recovery is better than reactive recovery; TCP-style timeout calculations outperform those based on a virtual coordinate scheme; and simple global sampling is as good as other much more sophisticated schemes in neighbor selection.

They've used the DHT implementation Bamboo, which is derived from Pastry but has been enhanced with the above techniques to handle churn. The code and additional information can be found at http://bamboo-dht.org.

Q: Is TCP-style timeout doing well because of absence of background traffic?

A: To some degree, the answer is probably yes, although the benchmark itself also creates some load imbalance. In any case, background traffic would probably hurt performance of a virtual coordinate scheme as well.

Q: How did you have a good timeout for the return path, and how did you measure latency when the return path was different from the lookup path?

A: There is an ACK for each lookup message on every hop to get latencies, and a conservative timeout is used for the return path. There is an average of five hops in the lookup and only one return hop, so this conservative estimate isn't too far off. We could not have explored the possibilities of using virtual coordinates for only the return hop, or have the return path be along the lookup path.

Q: In this paper, what is the dimension of virtual coordinate space?

A: It is a 2.5 dimensions space with x and y in a plane and z above the plane. The distance between ($x1$, $y1$, $z1$) and ($x2$, $y2$, $z2$) is Z1 plus Z2 plus the square root of $((x1–x2)^2 + (y1–y2)^2)$. The latest Vivaldi work seems to indicate that this is a good metric.

Q: Why does Tapestry work fine in simulation but not so well in a more realistic network emulation?

A: Because Tapestry has no leaf set; Tapestry really needs to recover quickly from routing table neighbor failures. This problem leads it to be built to recover reactively, therefore to suffer from the same problems that Bamboo/Pastry exhibits from reactive recovery.

Q: Have you tried some sort of periodic/reactive hybrid?

A: It is really hard to do without reverting to the policy of increasing traffic under stress. There's probably some middle ground, but we're not sure what it is yet.

A Network Positioning System for the Internet

T.S. Eugene Ng, Rice University; Hui Zhang, Carnegie Mellon University

Knowledge about network distance is essential for the performance optimization of a large distributed system. For an n-node system, directly computing the delay between each pair of nodes takes $O(n^2)$ time. The author proposes to solve the scalability issue by building a network positioning system (NPS) using a Euclidean space model. Each node infers its network coordinates by measuring their distance to several reference points, and network distance between any pair can then be calculated by their network coordinates.

In building such a network positioning system, there are many practical issues, including system bootstrap, how to support a large number of hosts, how to select reference points, how to maintain position consistency, how to adapt to Internet dynamics, and how to maintain position stability. The system is evaluated on PlanetLab with 127 nodes using an 8D Euclidean model; results showed that position accuracy was fully maintained through the 20-hour testing period.

Q: Is there any technique from NTP that NPS can also benefit from, considering that NTP and NPS have some common issues to deal with?

A: I'm not very familiar with NTP and therefore have no clear answer.

Q: Why did you choose to use 8D Euclidean space? Is there anything particularly important about the number of dimensions?

A: From my experience, 6 to 8 dimensions would all be fine. A too-low number would not yield the desired accuracy, and a too-high number increases the calculation complexity.

Q: Has the system been tested on a practical Internet domain other than PlanetLab, which has a more artificial environment? If yes, what was the result?

A: No. But I expect the scheme to work well on practical Internet domains, too.

Q: Why did you choose Euclidean space instead of another model?

A: Several other geometry models were explored; there wasn't much difference in the result.

Early Experience with an Internet Broadcast System Based on Overlay Multicast

Yang-hua Chu, Aditya Ganjam, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan, and Hui Zhang, Carnegie Mellon University; T.S. Eugene Ng, Rice University

Internet multicast has been studied for many years; the protocol design and evaluation were mostly based on static analysis and simulation. ESM (End System Multicast) is the first mature and deployed system to use Overlay Multicast for broadcasting video and audio streams. The system has had four publishers and 4000 users, providing unique experiences on Internet broadcasting.

ESM requires no change to network infrastructure; the end hosts in an ESM system are programmable to support application-specific customizations. ESM is a distributed and self-improving protocol optimized for end-to-end bandwidth. It supports heterogeneous receivers, NATs, and firewalls, and has user-friendly managing tools. The results indicate that the overlay tree built by ESM is quite optimized and well structured, and 90% of the users get 90% of the bandwidth. In this paper, Kay Sripanidkulchai presented a concept called "Resource Index" to quantify the bandwidth utilization in the system. When Resource Index indicates the system resource utilization is high, some users experience degradation of video quality. One important lesson learned from ESM is that there are a lot of NATs in the system (70%), which ties up resources and causes poor performance.

The ESM system can be accessed on line at http://esm.cs.cmu.edu.

Q: There have been several recent proposals, such as CoopNet and SplitStream, to use multiple trees for streaming. From your experience do you think multiple tree approaches are necessary?

A: From what we've seen, single trees seem to do fairly well, though for larger-scale groups, multiple trees may become useful.

Q: How do you deal with the data proxy server in your system?

A: It is counted as NAT behind firewall.

Q: How do you avoid "free rides," i.e., when a user lies about the resource that he or she is going to contribute to the system?

A: ESM uses actual measurement instead of trusting a user's claim.

GENERAL SESSION PAPERS: SECURE SERVICE

Summarized by Wanghong Yuan

Reliability and Security in the CoDeeN Content Distribution Network

Limin Wang, KyoungSoo Park, Ruoming Pang, Vivek Pai, and Larry Peterson, Princeton University

KyoungSoo Park first introduced CoDeeN, an academic content-distribution network on PlanetLab, and its security problems. The root of these problems is that CoDeeN has no end-to-end authentication. KyoungSoo then described their approach to security, which includes multi-level rate limiting and privilege separation. They achieve reliability by using active local and peer monitoring. In addition, he discussed DNS problems solved via mapping objects in the same proxy and CoDNS. Finally, he summarized the lessons and future work, including robot detection, CoDeploy and CoDNS. More information is available at http://codeen.cs.princeton.edu.

Q: What causes the DNS problems?

A: Local DNS server overload.

Q: What are other solutions for accessing local content?

A: More efficient approaches, with more information; currently the privilege separation is simple.

Building Secure High-Performance Web Services with OKWS

Max Krohn, MIT

The motivation story is SparkMatch version 1, which crashed with 500,000 signups. Version 2 solved some problems in the database, but there were too many connections. Version 3 in 2002 distributed database but the development cycle was too long. Max summarized the desired Web service features: thin fast server, smart gzip support, small number of database connections, memory reclamation, and an easy and safe way to run C/C++ code, and mentioned that the major problem is dynamic content. Currently, Apache/PHP does not work well, due to the poor isolation.

Max then described their system, called the OK Web server (OKWS). Its design is that of a multi-service Web site, and its isolation strategy is the least-privilege principle. Max also gave an example of how to build a Web service with OKWS and illustrated how the isolation works in OKWS in detail. OKWS is implemented in C++ with SFS libraries, database translation libraries, and Perl-like tools. The key point is one process and one thread for one service, without synchronization. SparkMatchv4 is built using OKWS, which compared favorably to Apache, Haboob, and Flash. The source code is available at http://www.okws.org.

Q: Is there an advantage for not maintaining the database pool?

A: Yes, based on observation of Apache.

Q: Why not replace script? Why develop a new Web server?

A: Security problems in Apache.

Q: How do you support two requests sharing the same service?

A: It's possible to do this; the paper has more detail on how.

REX: Secure, Extensible Remote Execution

Michael Kaminsky, Eric Peterson, M. Frans Kaashoek, and Kevin Fu, MIT; Daniel B. Giffin, David Mazières, New York University

The motivation is that remote execution is important but there are features not widely available in current tools. The major problem addressed in REX is locating the simplest abstraction that can support all of these features. Michael described how to establish a session, run a program, pass file descriptors, use X Window system forwarding, connect through NAT and dynamic IP address, and forward restricted credentials. The evaluation tries to answer two questions: Is REX reliable and are there any architecture benefits? More information is available at http://www.fs.net.

Q: Are there problems in file access?

A: Only in some access, such as write and read.

Q: What is the relationship between TCP and channel?

A: There is one TCP and there are multiple channels.

USEBSD SIG

Summarized by Adam S. Moskovitz

The NetBSD Update System

Alistair Crooks, The NetBSD Project

Alistair Crooks described a system for downloading and installing binary patches, similar in many ways to Microsoft's Windows Update Facility, but for use on any number of platforms. Crooks' system runs on a variety of platforms, including the *BSD variants, Mac OS X, Linux, and Solaris. Like the Microsoft system, NetBSD Update is easy to use and gives the user three options for automatic behavior: inform the user; inform and download appropriate packages; inform, download, and install the packages/patches. Crooks uses a file on the update server to list packages for which vulnerabilities exist and a program that runs on the target system to say which of those packages is present and should be updated.

NetBSD Update includes other important features, the most significant (in my mind) being the ability to digitally sign update packages and the user's ability to accept or reject those updates based on the validity of the signature(s). Unfortunately, like so many other systems, the lack of a widely accepted public key infrastructure means this feature is still a bit more cum-

bersome than it ought to be (which, of course, should not be considered a shortcoming of Crooks' work).

Another feature is the ability to undo the effects of an update. NetBSD Update automatically preserves all files that will be overwritten and stores them for the user in case the update causes more problems than it solves, or if worse vulnerabilities are found in the update than existed in the unpatched system. [Summarizer's note: Of course, this sort of thing has never happened—the phrase "the patch for the latest jumbo patch" is merely a joke among system administrators.]

A Software Approach to Distributing Requests for DNS Service Using GNU Zebra, ISC BIND 9, and FreeBSD

Joe Abley, Internet Systems Consortium, Inc.

[Summarizer's note: This talk/system deals with certain aspects of routing that go beyond my general knowledge of the subject; if something you read doesn't make sense, the error is almost certainly mine and not the speaker's.]

Joe Abley described a system for distributing DNS requests across multiple hosts without the use of dedicated load balancers, by using a "service address" (sometimes called a "virtual IP address"), anycast, and the Equal Cost Multi-Path (ECMP) feature of the OSPF routing protocol. This system is currently in use for the F root name server (run by ISC—the Internet Systems Consortium), which also provides slave service for 30–40 ccTLDs. Currently, 24 nodes across California and in New York, Tokyo, and Stockholm make up what appears to be a single root name server.

Individual nodes in the cluster are configured with unique unicast IP addresses, and with the service address on the loopback interface. Hosts inform the routers of their readiness to answer requests via OSPF Link State Advertisements; simple wrapper informs the router when an instance of "named" fails (internal assertion failures are set to dump core and exit).

For UDP-based DNS queries, it doesn't matter which node in a cluster provides the answer. For TCP-based operations like zone transfers, all packets in the transaction must be routed to a single node; Abley uses a combination of "flow hashing" (via Cisco Express Forwarding or the "load-balance per-packet" feature on Juniper routers), avoiding ECMP routes for stateful transactions, and using BGP.

GENERAL SESSION PAPERS: THE NETWORK-APPLICATION INTERFACE

Summarized by Calicrates Policroniades

Network Subsystems Reloaded: A High-Performance, Defensible Network Subsystem

Anshumal Sinha, Sandeep Sarat, and Jonathan S. Shapiro, Johns Hopkins University

Anshumal Sinha introduced his talk with several issues observed in in-kernel monolithic network systems: security (they represent a single point of failure), maintainability (robustness-critical code is large and difficult to maintain and debug), and flexibility (lack of support for the simultaneous existence of multiple protocols, complexity to do application-specific optimizations). He stressed that monolithic network systems are only used because of their performance benefits. The author mentioned that previous user-level implementations have failed to deliver sufficient throughput, noting, however, their hypothesis that earlier systems failed to provide an appropriate solution to a key problem: performance degradation resulting from data copying from one address space to another in order to provide protection domain boundaries efficiently. Failing to properly manipulate data from different protection domains degrades the performance of the system.

The author then presented the methodology to evaluate their solution. Based on the EROS micro-kernel to support domain factoring, they built two network subsystems to evaluate the costs due to the user-level implementation, the domain factoring, and the microkernel performance. In particular, they built an EROS-based monolithic network subsystem and an EROS-based domain factored network subsystem. Anshumal explained in detail the implementation of both systems and their distinctive features. When presenting experimental results and evaluation, he included a conventional Linux in-kernel network stack as a reference baseline for performance comparison. A detailed explanation of their results in terms of latency and throughput showed that the performance exhibited by the domain factored network subsystem was comparable or close to the other two strategies (EROS monolithic approach and conventional Linux network stack).

Anshumal concluded his talk by remarking that domain factoring is more feasible than previously assumed, that the instruction cache plays a significant role in the performance of the system, and that factoring provides the basis for defensible systems.

accept()able Strategies for Improving Web Server Performance

Tim Brecht, David Pariag, and Louay Gammo, University of Waterloo

Tim Brecht discussed how particular strategies to handle connection requests affect the performance of Web servers. He began by mentioning how to improve Web servers' performance by modifying their corresponding accept strategies. He mentioned that an adequate solu-

tion should not only improve Web servers' peak performance but also be able to maintain it even under overload conditions with a large number of connections. He presented throughput results for three architecturally different Web servers: the event-driven, user-mode micro-server (39–71% improvement); the multi-threaded, user-mode Knot (0–32%); and the kernel-mode TUX (19–36%).

Tim stressed that current multi-accept servers overemphasize acceptance of new connections and ignore the processing of existing connections. Second, he mentioned that his work aimed to reduce the gap in performance typically seen between kernel-mode and user-mode servers. Next, he described in detail the three architectures that were analyzed in the paper and explained how the accept-limit, which defines an upper limit on the number of connections accepted consecutively, affects each technique's performance.

The presenter made a careful analysis of the impact of varying the accept-limit on each of the servers based on their experimental results. In his experiments they used two workloads: a one-packet workload and a SpecWeb99-like workload that uses httperf to generate overload conditions. The experimental results presented by the author were organized by server performance, queue drop rates, and latencies observed under different accept-limit policies. Tim mentioned that it is necessary to ensure that Web servers accept connections at sufficiently high rates so that a balance between accepting and working times can be adequately established. Finally, he said they were able to demonstrate that performance improvements can be obtained by modifying the accept strategies used in Web servers.

Lazy Asynchronous I/O for Event-Driven Servers

Khaled Elmeleegy, Anupam Chanda, and Alan L. Cox, Rice University; Willy Zwaenepoel, EPFL, Lausanne
Presenter: Khaled Elmeleegy

Khaled Elmeleegy began his presentation explaining why event-driven architectures are used and the difficulties that programmers experience when developing high-performance servers using existing I/O libraries. He remarked that current I/O libraries have an incomplete coverage or leave to applications the burden of state maintenance. In contrast, Lazy Asynchronous I/O (LAIO) is a good option to develop high-performance, event-driven servers with less programming effort, because it covers all possible blocking I/O calls, creates a continuation only when an operation actually blocks, and notifies applications only when a blocking operation has been wholly completed. Next, Khaled explained the way in which event-driven servers generally work and the role that event-handlers play in their operation; he also mentioned how blocking operations degrade server throughput. He presented LAIO as a solution to the typical blocking problems seen in event-driven servers and proceeded to describe the LAIO API, their functions, and implementation.

After introducing the two different workloads used in their experiments, Khaled compared LAIO's throughput and performance against other I/O libraries. With this purpose, the authors modified the networking and disk I/O strategies of the Flash Web server and measured the results obtained for different versions of the server. In situations where performance was comparable, the complexity of the programs decreased because it is not necessary to write handlers or to maintain state as in conventional non-blocking I/O, which finally leads to a reduction of the amount of code that needs to be written.

Khaled finished his talk by highlighting LAIO's generality (covering all the I/O calls) and simplicity (requiring fewer lines of code without handlers or the need to maintain state). In terms of throughput, LAIO meets or exceeds the performance of other methods. During Q&A, the audience mainly focused on comparing LAIO with multi-threaded servers; Khaled mentioned, however, they had decided to focus the study on event-driven strategies.

USEBSD SIG

Summarized by Matus Telgarsky

Building a Secure Digital Cinema Server Using FreeBSD

Nate Lawson, Cryptography Research

Nate's dense, practical, and often anecdotal talk went beyond general crypto issues and concepts to also discuss the troubles and travails afflicting construction of a digital cinema server and problems with its wide acceptance.

After providing a quick overview of Cryptography Research Inc., Nate presented an extremely cogent diagram that proved one of the strongest bromides of the security circuit—you are only as strong as your weakest link. A graph pitted probability of compromise against effort/cost of attack. The perfect scenario features a deep curve predicting massive effort for even the slightest crack; however, usual circumstances produce an almost inverted curve (symmetric against $y=x$), where bribing employees, using common scripted attacks, and abusing operating system holes is often easy and constitutes the majority of compromises, rather than a crack of an encryption key, a shield many often deem fully sufficient.

Nate continued to describe three fundamental underpinnings of any security paradigm: strength (which encryption provides),

assurance (pragmatic assessment of attack methods, especially easier entrances), and renewability (post-mortem reconstruction).

Traditional analogue cinema (obviously) uses film cameras, is transferred to a digital format for post-processing (effects, editing, etc.), is printed thousands of times (at $3,000 a copy, which degrades after a week of use anyway), and is played on $30,000 projectors. Digital cinema is directly captured to hard drives and obviously omits any tedious conversion; however, distribution and projection standards do not exist—costs are prohibitive, and the market is in flux. Not only does refitting cost around $100,000, but there are even questions whether the theater is responsible for said expenditure. In 2003, only 90 theaters in the U.S. were digital (30 in 2000).

Digi-Flicks enlisted Cryptography Research to design a new security system and, hopefully, extend digital cinema acceptance. Planned design goals were transport independence, thorough use of strong crypto algorithms, multi-factor authentication (i.e., simultaneously utilized smart card, pass code, and key file), flexible authorization policies, reliable playback over imperfect media, and, of course, a rapid development cycle. Amusingly enough, for the 300 target theaters, shipping hard drives was found to be the most cost-effective distribution method.

The sample extant hardware was rather unpleasant—a simple UNIX-like OS over a 33MHz PowerPC with 64MB RAM, too many ASICs, and no documentation. Serial and even Ethernet proved too slow for data transmission, so the SCSI interface was selected to actually transfer the films (FreeBSD was selected partially due to the ease with which the disc access code could be modified to play nice with this chaotic configuration). The encrypted drive would be decoded with a smart card and by dialing in to an auth server. Nate spent a moment covering common pitfalls in encryption selection, including an amusing pair of images presenting a still identifiable pattern in an encrypted image due to naively small block selection.

A recurring suggestion was careful thread-model analysis in order to realistically and sensibly determine circumstances and identify weak points, and then assign design parameters accordingly. Nate detailed many possible scenarios (one-time read-only access, repeated read-only access, one-time read-write access, and repeated read-write access) and the specific dangers within each. Similarly, a good security structure depends on clear top-down study and careful perusal of all possibilities.

Panel: The State of the BSD Projects

Chair: Marshall Kirk McKusick

The FreeBSD Project: Robert Watson, Core Team Member, The FreeBSD Project

The NetBSD Project: Christos Zoulas, President, NetBSD Foundation

The DragonFly BSD Project: Matt Dillon, Project Leader, The DragonFly BSD Project

All attending BSD parties were able to give impromptu presentations detailing past work, current revelations, and future plans. FreeBSD, NetBSD, and DragonFly BSD were represented; rumblings abounded regarding Darwin and OpenBSD, supporters of which were unfortunately concurrently occupied with other tasks.

Robert Watson's speedy flight through FreeBSD 4 (stable) and 5 (development) built a good measure of excitement and confidence in the impressive list of features stabilizing in the new code. 4.10 has survived healthily with minor security updates and has enjoyed hardened security. 5.X has been in continuous development for five years, with hard work solidifying a new SMP model and threading core, among numerous other improvements. 5.3 features include gcc 3.4, PCI and ACPI work, X.org X server, more fine-grained locking work (including heavy "giant" lock removal in many subsystems), SMP thread scheduler tweaks, and the flexible pf packet filter, among others. SMPng is satisfactorily transitioning from bug and correctness checking to performance tuning.

Next came Alistair Crooks presenting NetBSD, alive since 1993 and eagerly awaiting a 2.0 release. Nascent features include SMP work, scheduler activations, kqueues, wireless drivers, and many other features. A primary goal of NetBSD is to function on many architectures—the netbsd.org sidebar presents an impressive list of 54 disparate devices. Time was also spent describing the package distribution system, pkgsrc, which is easily configurable, consistent, supports multiple versions of installed programs, and currently boasts over 4500 packages.

Matt Dillon discussed his ambitious DragonFly BSD project, which is steadily advancing upon a 1.0 release. Already a veteran hacker (contributor to Linux and FreeBSD, among many other projects), Matt's aggressively progressive plan for restructuring BSD revolves around a message-passing core with a lightweight IPC model. Much of the talk and current focus is extremely low level—for instance, much focus is going into restructuring to optimize cache usage in all subsystems, from the ground up. One corollary goal is minimizing use of fine-grained mutexes. DragonFly is a continuation of FreeBSD 4.X, though a pleasant rapport exists with the parent project, and indeed updates still filter through.

This enthusiastic one and a half hour panel showed the BSD projects to be in excellent condition,

with a dedicated group of hackers backing each—in fact, one of the few times I witnessed the BSD hackers leaving the laptop room (and their diligent hackery) was to attend this informative panel. FreeBSD and NetBSD pointed out their foundations, facilitating donations through cheerily tax-deductible exchanges. DragonFly has not yet formed a foundation, though Matt Dillon would certainly enjoy frequent surreptitious anonymous gifts.

PLENARY SESSION

Summarized by Swaroop Sridhar

Thinking Sensibly About Security in an Uncertain World

Bruce Schneier, Counterpane Internet Security, Inc.

Mr. Bruce Schneier delivered his thought-provoking and entertaining talk without any kind of visual aid. He began by saying that we are living in an interesting era called "silly security season." Introducing the notion of all of us being "security consumers," he said that we need to step back and analyze whether this security is really worth it. Is it worth the billions of dollars and the loss of convenience, anonymity, performance, or freedom? Elaborating on security trade-offs, he said that it is well known that trade-offs are ubiquitous, but there is a fundamental security trade-off paradox: We, who claim to be the "most intelligent" species on the face of this earth, always make the wrong security trade-offs. Much of Schneier's talk was based on why this is so and how it could be fixed.

Schneier said that the way to do good security trade-offs is to slow down and have a basis for rational discussion, rather than making quick decisions based on emotions. However, he warned that this could be quite complicated because the meaning of factors such as inconvenience, risk, and privacy are often subjective.

Next, Schneier brought up the topic of risk assessment. He warned that people are always bothered about "spectacular" risks (e.g., risks while flying in a plane) and downplay "pedestrian" or "under control" risks (e.g., risks while driving), which matter much more in our lives. Schneier identified technology and media as two main culprits causing this problem. News, by definition, means that which does not happen every day. The media only show the uncommon happenings, replaying them over and over to create a feeling that they are very common. Technology contributes its bit, obscuring risks by hiding operational details from users.

Again, bringing up the topic of why extreme trade-offs (such as National ID cards) are taken for little gain, Schneier said that security decisions are usually made for non-security reasons. This leads to a notion of an "agenda" among all the "players" of the bigger system, of which security is a part. For example, closing national highways is good according to a police agenda, but bad according to a public agenda. Schneier proposed a model of "Security Utilitarianism"—which leads to the greatest security for the greatest number of people.

Schneier stated that one of the fundamental problems is that we often have no control over the security policies that are implemented. The right kind of security should be worked out by means of negotiations and deliberations. He cautioned, however, that the negotiations should be held at the right time, with the right people. Arguing with a security guard at an airport gate, for example, would be a bad idea. Schneier identified four factors that can effect a change in security norms—government rules and laws, market forces (e.g., refusing to use an insecure OS), technology, and social norms. He said that by turning the above four knobs, we must be able to work out the right kind of security.

Later, Schneier said that we need to accept the risks as real, and try to reduce them. He also proposed that one possible solution is to put the person who can best mitigate the risk in charge of the risk. He illustrated this point with the example of a supermarket cash register, where the customer is "used" to guard against cashier malpractices. Schneier concluded by saying that as individuals we have very little power, but as an aggregate we can achieve a lot toward our collective good.

GENERAL SESSION PAPERS: UNPLUGGED

Summarized by Matus Telgarsky

Energy Efficient Prefetching and Caching

Athanasios E. Papathanasiou and Michael L. Scott, University of Rochester

■ *Awarded Best Paper*

Modern operating system design prescribes a plethora of heuristics for caching and prefetching to aid in disk performance, with nary a word about energy savings. Studies attribute 9–32% of laptop energy expenditure to hard disk use; hence, heavy savings in that realm will result in drastic overall improvements.

Traditional prefetching techniques aim to reduce disk access latency by attempting to maintain the working set of an application's disk data cached by replacing unused cache elements with simplistically determined prefetch targets. Unfortunately, this does not preclude the possibility of an application reading and writing at arbitrary times, obviating the possibility of simply tacking on any sort of basic power management scheme. Indeed, many of the tests on a stock Linux kernel

showed 100% of the disk idle times to fall beneath one second, not nearly long enough to enter a disk's power-saving state without incurring a net power efficiency loss owing to energy required for transition.

The presentation detailed the design and implementation of Bursty, a mechanism providing highly speculative prefetching, a kernel interface to hint disk access patterns, and a daemon both to monitor and manage the system. Applications may hint improperly, or lack hints entirely, so the monitor must both generate and judge extant hints. The prefetching is also self-aware—the success rate of the algorithm is constantly measured to determine whether further adjustments are required. Additionally, per-application idle times are irrelevant if they are not in phase between applications; hence, the daemon also attempts to organize these patterns to allow for consistent disk avoidance between all applications. Once the predicted idle period is estimated to beyond the intersection of regular drive use and idle use combined with transition expenditure, the drive is powered down into an appropriate low-power mode. A variety of tests with different applications using a variety of workloads and disk access patterns (and memory configurations—Bursty is hungry!) found 60–80% energy savings, with negligible losses in efficiency.

Time-Based Fairness Improves Performance in Multi-Rate WLANs

Godfrey Tan and John Guttag, MIT

Modern Wireless networks theoretically maintain decent throughput when congested, though in practice the common utilization of rate diversity as an automatic signal strengthening scheme causes standard throughput-based fairness schemes to result in unexpectedly poor performance. This paper presents an overview of the problem, design and implementation of a solution—termed "time-based fairness"—and experimental verification.

802.11b was used as the test case. Though traditionally known to sport 11Mbps, the standard also defines three other rates: 1, 2, and 5.5. Vendors use these speeds when packet transmission failure becomes a problem, eventually bumping the speed to the slowest rate, which features the highest signal resilience. Current channel proportioning and access point downlink scheduling techniques result in throughput-based fairness, meaning a slower rate receives a larger portion of channel time, ostensibly aiding the feeble companion but causing the hare to be tied to the turtle.

Time-based fairness apportions channel use equally by time, resulting in much higher possible throughput. The average time for network tasks to complete is also reduced, obviously a benefit to many mobile users and definitely to anyone who would rather have things to do while a laggy transmission completes. The implementation is flexible enough to function properly on extant access points and does not need extensive modification on clients; adoption is easy and backwards-compatible.

EmStar: A Software Environment for Developing and Deploying Wireless Sensor Networks

Lewis Girod, Jeremy Elson, Alberto Cerpa, Thanos Stathopoulos, Nithya Ramanathan, and Deborah Estrin, UCLA

The burgeoning study, experimentation, and deployment of wireless sensor network applications is generating a need for full-fledged development suites. EmStar provides just that for 32-bit embedded MicroServer platforms: tools and libraries providing simulation, visualization, and emulation. Other functionality aids in development and testing of IPC and network communication.

Due to time constraints, a large portion of the talk focused on FUSD (Framework for User-Space Devices), which is a kernel module proxy to device file events. FUSD, though new, is already used by numerous applications to simplify communication with device nodes. It is essentially a micro-kernel extension to Linux. At the moment, performance is sufficient but unsatisfactory; read throughput, for instance, can be 3 to 17 times slower than analogous read performance without the FUSD proxy overhead.

EmSim and EmCee are simulation tools; these in turn are modular to allow for easy extension and minimized footprint. EmRun starts up, maintains, and shuts down an EmStar system according to a policy in a configuration file; it features process respawn, in-memory logging, fast startup, and graceful shutdown. All components (more than are listed here) are written with modularity in mind, and code is heavily reused. It has already proved useful in numerous projects at the CENS labs working with a variegated set of hardware.

SECURITY SIG

Summarized by Ming Chow

Panel: The Politicization of Security

Moderator: Avi Rubin, Johns Hopkins University

Panelists: Ed Felten, Princeton University; Jeff Grove, ACM; Gary McGraw, Cigital

The common theme of this panel was how politicized security, especially that relating to technology, has become. Professor Avi Rubin spoke of his experiences working on the issue of electronic voting (eVoting). He spoke about dealing with policy issues, and about how eVoting has become a partisan, politically charged issue and, as

such, is targeted for abuse. An example is that companies producing eVoting technologies and equipment have strong political ties. The goal from each political party is to "not have the other guy win." Professor Rubin has been on major news sources (e.g., CNN) speaking about technical issues of eVoting, and has received numerous telephone calls from both Democrats and Republicans. Professor Rubin recounted being called to testify in front of Congress about eVoting, and recalled the amount of fighting and bickering on both political sides dealing with the issue. He summed up the current state of politics by saying that "partisanship has never been worse."

Gary McGraw spoke of the long history of politicization of scientific research and development and the degree to which current scientific research and development are influenced by politics (like Galileo and Darwin centuries ago). He stated that security and terrorism are sensitive subjects, and that "we should understand the problem, having worked in an asymmetric situation for years in computer security." McGraw also said that too often "individual rights can be trumped in the name of security" (e.g., DMCA and the Patriot Act).

Jeff Grove has worked with the government on Capitol Hill, and expressed his dissatisfaction on the number of bad laws being implemented, including the DMCA, and the regulation of P2P networks. Grove outlined how the Senate can address and jump on issues and make dumb laws. The problem persists because of bad conclusions, bad assumptions, and lack of basic understanding about technologies. In addition, there is a small handful of powerful players who are effective in influencing the government to create laws fitting their agendas. Bad laws expose developers to liabilities, even when there's no infringement, and provide civil enforcement by encouraging legal actions by the entertainment industry.

Professor Ed Felten spoke about the Digital Millennium Copyright Act (DMCA) and his work, which made national headlines several years ago. Professor Felten stated that the DMCA was created by negotiations in which computer scientists were not involved. His work with advisee John Halderman was discussed—the weak DRM technology created by SunnComm could be bypassed on Windows computers by holding down the shift key. The government was cracking down on Professor Felten's research and he was threatened by the RIAA under the DMCA. Professor Felten settled with both Princeton University and the government by creating educational packets for the government on security research. Professor Felten recalled testifying in Congress about a bill to limit developing tools on decoding technologies, and summarized the atmosphere in one word: "theater." Finally, he gave out his Web site: http://www.freedom-to-tinker.com.

The theme from all of the panel speakers was clear: "We (the computing and scientific communities) need to step up to the plate and educate people on technological issues." The goal can be accomplished by being more involved, by being partisan, and by talking to anyone who is curious. Openness and debate are encouraged and are healthy. It is critical to tell the truth and to convince people about what's really going on. Gary McGraw also said that attacking systems is a necessary part of security and that outlawing attacks makes little sense. Finally, media and politics are great investments: the "Slashdot effect" helps ridicule bad laws, and working even with your local government is a 10–15-year investment.

FREENIX OPENING REMARKS AND AWARDS

Summarized by Martin Michlmayr

Bart Massey, Portland State University; Keith Packard, Hewlett-Packard Cambridge Research Lab

Bart Massey and Keith Packard opened the FREENIX track, a forum devoted to free and open source software, by giving a brief summary of papers that were submitted this year. Out of 61 papers submitted, 15 were accepted. The organizers were happy to see that among the accepted papers, seven were from students, and seven were non-US papers. They said that the quality of all submitted papers was very high and that the review process was more formal than in the last few years, adding three external reviewers to the program committee. They also thanked DoCoMo for sponsoring student travel for the conference.

In this opening speech, two awards to papers in the FREENIX track were given. The Best Paper award went to "Wayback: A User-level Versioning File System for Linux," and the Best Student Paper was "Design and Implementation of Netdude, a Framework for Packet Trace Manipulation."

There will be another FREENIX track at USENIX '05 in Anaheim, California. Since future USENIX conferences will take place around April, the deadline for FREENIX submissions is October 22, 2004, rather than in December. More information on the next FREENIX track and Call for Papers can be found at http://www.usenix.org/events/usenix05/cfp/freenix.html.

FREENIX INVITED TALK

Summarized by Martin Michlmayr

The Technical Changes in Qt Version 4

Matthias Ettrich, Trolltech
Linux/Open Source

Matthias Ettrich, founder of the KDE project and a main developer on Qt, gave an overview of the next generation of Qt, a cross-platform C++ GUI toolkit. Qt supports X11, Microsoft Windows, Mac OS X, and embedded Linux, and offers native look and feel on each of these platforms. Qt provides single-source compatibility: one source code compiles on all target platforms. While Qt mainly offered GUI functions in the past, it is much more than a GUI library these days: It also supports I/O, printing, networking, SQL, process handling, and threading. One aim of Qt is to provide an excellent programming experience.

Qt introduced the signals-and-slot concept in order to allow different GUI components to communicate. You can connect any signal to any number of slots in any module, and communication is done at run time. The sender and receiver don't need to know each other. In version 4, connections can be either synchronous or asynchronous ("equal connections"); this will allow thread communication. Arthur is Qt's paint subsystem, and version 4 will offer several new features: linear gradient brushes, alpha-blended drawing, anti-aliased lines, painter paths, and an OpenGL backend.

Interview is a model/view framework for tree views, lists views, and tables. In the Model-View-Controller (MVC) paradigm, all these components are separated from each other. The model contains data, the view renders data, and the controller transforms interaction with the view into actions to be performed on the model. Qt 4 will introduce semi-transparent windows and flicker-free painting, and it will also implicitly provide double-buffering for all built-in and custom widgets: this is transparent, and no code needs to be rewritten. In addition, it will allow large windows (even modern window systems limit a widget's coordinate system to 16 bit, but Qt 4 won't have this limitation), and there will be improvements in size and performance. Qt 3 was originally designed for desktop computers (with fast CPUs with FPUs, lots of RAM and disk space). On the other hand, Qtopia was designed for embedded systems. Qt 4 aims at merging the benefits of both product lines into one.

In summary, Qt 4 will provide a number of new features that will offer new possibilities for cross-platform development. Ettrich hopes that a first technology preview will be made really soon, with another one following in Q3 2004. A beta of Qt 4 should be released in Q4 2004, with the final version following in Q1 2005.

SECURITY SIG

Summarized by Ming Chow

Panel: Wireless Devices and Consumer Privacy

Organizers: Ari Juels, RSA Laboratories; Richard Smith, Consultant

Panelists: Markus Jakobsson, RSA Laboratories; Frank Schroth, uLocate; Matthew Gray, Newbury Networks

The panel talked about wireless technologies, including GPS and RFID, and privacy issues concerning them. Frank Schroth discussed uLocate's wireless technology, which enables small business users to view the location of phones in their account, including maps and routes. uLocate's technology is based on GPS on the Nextel Network. The interface on cellular phones is a Java-based application that transmits data to server via UDP. Permissioned users can view information on their phone or via Web. The benefits of the technologies to small business include convenience, efficiency, and safety. The security of the uLocate service consists of two layers: a carrier level and an application level. Mr. Schroth also discussed privacy concerns about the technology, namely, addressable IP addresses, privacy, and leadership and ownership of risks.

Matthew Gray of Newbury Networks discussed his concerns about location-tracking technologies. The goal at Newbury Networks is to see what people are accessing without interfering with larger networks (e.g., Starbucks) and to eliminate such false positives to enhance security and privacy. Mr. Gray noted that security and privacy are in opposition to each other. He said that consumers and regulators must understand the risks of location-tracking technologies.

Marcus Jakobsson of RSA Laboratories listed three ways in which location privacy can be violated: active attacks (keep asking a device), passive attacks (listen to communication from other devices), and remote attacks (infer location from public information). He said that legislation for location privacy is necessary and meaningful. However, such law will be difficult to enforce because detecting abuse by institutions is hard, and it is even harder to detect abuse by individuals. He noted that countermeasures have been proposed but not deployed. In conclusion, Mr. Jakobsson listed several things that must be done immediately: the threats of location privacy must be studied and understood, legislation must be enacted, and countermeasures must be implemented.

Finally, Ari Juels and Richard Smith discussed radio frequency identification (RFID) tags and privacy concerns about the technology. Mr. Juels presented a brief tutorial of the RFID technology: an RFID tag

uses a chip (IC) antenna slightly larger than a quarter. Currently, many people have tools and gadgets that have RFID tags, such as E-ZPass, Mobil SpeedPass, and physical-access cards. RFID tags are seen as next-generation barcodes; Mr. Juels listed the benefits of RFID tags over barcodes (they're fast, efficient, mobile, can uniquely specify objects, and require little computational power). What this means is that the world will consist of billions of $0.05 computers. The major privacy problem concerning RFID technology is that it can be used to profile a person incredibly easily and quickly, providing detailed information, for example, on artificial body parts and other details of a person. Mr. Juels noted that approximately 42% of Goggle hits on a search for RFID contain the word "privacy." The solution to the privacy problem is to kill RFID tags. However, RFID tags are too useful. Mr. Juels concluded his talk by saying that there is serious danger to privacy if the technology is deployed naively, but the danger can be mitigated to strike a technical balance with society.

At the end of the talk, the panel discussed what must be done now to mitigate privacy concerns. One question to the panel was whether policy legislation hurts or helps technical development. A member of the panel suggested that a policy of saving data for 90 days would be sufficient. There was also a discussion about disclosure of information to consumers. Mr. Schroth responded that it has been startling to him how people do not care about disclosing information about themselves: people are willing to give lots of information to companies, including passwords.

FREENIX SESSION: SERVER

Summarized by Matus Telgarsky

Migrating an MVS Mainframe Application to a PC

Glenn S. Fowler, Andrew G. Hume, David G. Korn, and Kiem-Phong Vo, AT&T Labs

Rotting at the hearts of many old institutions' organizational frameworks are mainframes and their respective applications. Though the software may be relatively dependable, operational costs are prohibitive (the task emulated within this paper is estimated at $20,000 per month just for mainframe use), and the code consists of thousands, even millions of lines of ancient COBOL and JCL, on a system without a hierarchical file system. Emulating the process is a feasible and cost-effective alternative.

The MVS was to handle a mammoth of data, so a variety of tools were written to efficiently compress it prior to transmission. The OpenCOBOL compiler was extended to handle a few language extensions and different character sets, parse compressed data directly, and also receive a few performance enhancements. An extended sort program was built to enable MVS features, and a flexible JCL interpreter was built with handy features such as ksh script generation. An unsophisticated scheduler was developed to emulate MVS handling of processes.

David Korn took a moment to quip that a 25-year-old tip indicated that sort is optimally performed on a UNIX machine by transferring it to tape, performing it on a mainframe, and transporting it back—yet today the situation is reversed. Two 2.8GHz Pentium 4 machines were used to emulate the mainframes, at under $4,000 total. The 60-hour MVS task took 19 hours on the shiny new silicon. Data transmission ballooned surprisingly to nearly 24 hours due to tapes acting—predictably—unpredictably fussily.

C-JDBC: Flexible Database Clustering Middleware

Emmanuel Cecchet, INRIA; Julie Marguerite, ObjectWeb; Willy Zwaenepoel, EPFL

The general trend in modern high-power computing is toward clusters of commodity machines, achieving a significantly superior price-power ratio over more traditional and expensive many-CPU SMP machines. Though many tiers of server applications have extended to utilize this trend, in general RDBMS installations have lagged behind. Limited support has come from Oracle and IBM, but open source databases either relied on simplistic master-slave replication services or other similar compromises.

Clustered JDBC (C-JDBC) is an open source database middleware which abstracts pools of databases into a virtual database, complete with load balancing, query caching, logging, checkpointing, scheduling, authentication, and other features. JDBC is used to connect to virtually any database (as they basically all provide JDBC drivers), and allows for seamless integration of heterogeneous database farms into single resources. Performance has also been considered deeply: For most workloads, increasing nodes results in linear benchmark improvements, meaning superbly minor overhead and excellent scalability.

Fault tolerance and redundancy are not only accounted for with a flexible load balancer, but C-JDBC controllers themselves can be stacked horizontally to virtualize the same databases and seamlessly provide redundancy. Arbitrary trees may be constructed by attaching C-JDBC controllers as client databases to other C-JDBC controllers. Though only 10 months have passed since its initial beta release, C-JDBC has already been downloaded more than 15,000 times.

Wayback: A User-Level Versioning File System for Linux

Brian Cornell, Peter A. Dinda, and Fabian E. Bustamente, Northwestern University

■ ***Awarded Best Paper***

Modern file systems and operating systems, though very tolerant of naughty applications thanks to caching, journaling, prefetching, locking, and a whole pile of other nuances, are still rather unhelpful to the common and simple user errors of accidental file deletion and overwrites. Some efforts have been undertaken to provide generalized undelete operations, and code management repositories provide versioning, but for the most part these sorts of mistakes must be protected against at the application level, if at all.

Wayback provides a versioning history through a hidden undo log for any directory remounted with it. It depends on the underlying file system to provide the storage, hence alleviating complexities arising from partitioning and supporting the disparate menagerie of extant file systems. The undo log is precisely that—it records data, allowing it to return to the previous state of a file. Any write operation causes a new entry to be placed in the log. File-system calls are observed by using the FUSE proxy, greatly facilitating development but unfortunately hampering speed slightly. Even so, Wayback is quite quick, usually not too far regressed from the file system inside FUSE.

Future plans potentially include compression, redo logs to provide bi-directional revision traversal, and even hierarchical version storage. The system is absolutely transparent and provides a simple usage paradigm to return to old versions. Since all changes are stored, a foggy memory and a dim spark of energy are enough to recover practically anything.

SECURITY SIG

Summarized by Ming Chow

Debate: Is an Operating System Monoculture a Threat to Security?

Dan Geer, Verdasys, Inc.; Scott Charney, Microsoft

Moderated by Avi Rubin, Johns Hopkins University

In one of the most anticipated events of the conference, Dan Geer debated Scott Charney on whether an operating system monoculture is a threat to society. Dan Geer—an instrumental contributor in the MIT Athena Project, former CTO at @Stake, Inc., and the former president of USENIX—argued that the problem is avoidable and mitigable, but difficult. He compared the current situation to the natural world and biology, and how a diverse gene set mitigates predation and disease. In a computing context, a diverse series of operating systems mitigates the onslaught of attacks and security breaches. Dr. Geer said that "if there is a monoculture, then the bigger the species, the juicier and more attractive." He was critical of the current state of computing and Microsoft's dominance: There are too many gadgets in Windows, leading to confusion as the operating system goes beyond its threshold. In addition, he stated that there are more serious security problems that are not publicly known, and virus writers are two steps ahead of antivirus writers. Dr. Geer concluded his arguments by reflecting on history, that "lessons learned in the real world apply to the computing world: All monocultures live on borrowed time like cotton and potatoes, and we are subject to the laws of nature."

Scott Charney spent years working in the public sector, most notably combating cybercrimes as chief of the Computer Crime and Intellectual Property Section (CCIPS) in the Criminal Division of the US Department of Justice. Mr. Charney stated that diversity of operating systems is not the issue, nor does it matter; the real issue is "how to catch the bad guys." He explained that there really isn't a monoculture of operating systems—not all Windows systems are alike. In fact, Mr. Charney argued that monocultures may be beneficial. He gave the example of Southwest Airlines and how all the planes are Boeing 737s. Southwest Airlines has accepted the risks that all their planes are Boeing 737s, and the major benefit is low-cost maintenance (e.g., pilots can operate any plane without having to learn each one individually). Mr. Charney explained that a computer hacker's goals are to compromise confidentiality and the integrity and configuration of accounts and systems. Finally, Mr. Charney compared the digital age to the Industrial Revolution: makers do not want security because the benefits of technology outweigh security. He believes that it is unfortunate that our current situation reflects that of the Industrial Revolution, and we need to look at security holistically.

During the Q&A session, Dr. Geer was asked about recommendations regarding the fact that there are only a small handful of operating systems available. He responded by saying that "we do not have enough alternatives" and "standards that matter must be platform independent." Mr. Charney was asked about the insecurity of Microsoft Internet Explorer. He responded that Microsoft is releasing better APIs for its products as part of its antitrust settlement. He added that "the issue [security] is in large applications."

Both experts gave their closing remarks after all questions. Mr. Charney concluded by saying that "we have dug ourselves into a deep hole and we need to understand computer security issues holistically to dig out of the hole." Dr. Geer concluded by ascribing the public sickness to bowing to one operating system, and pointed out

that virus writes attack one culture. He reminded the audience of nature's lessons, and that those with the most to lose are those who are the most interdependent.

PLENARY SESSION

Summarized by Patty Jablonski and Todd Deshane

Cheap Hardware + Fault Tolerance = Web Site

Rob Pike, Google, Inc.

As we were waiting for the presentation to begin, a continuous stream of words and phases in many different languages scrolled down the open terminal window that showed on the projector screen. This stream of words and phrases contained people's names, Web sites, and places. Rob Pike opened the presentation by saying that this is an example of unfiltered search queries that people submit to the Google search engine and that this is what Google looks like "from Google's perspective."

To demonstrate Google's global presence, Rob showed a map of the world that depicted queries/square degree/hour for a selected day. Spots of light represented the Google queries received from a given area of the world at that time. He noted that there was a place in Tokyo, Japan, that never goes dark. He chose this particular day of data because on that day, back in August, the northeastern United States had experienced a power outage. This area was seen as a dark area on the map that would otherwise have been lighted. He jokingly said, "Where there's electricity, there are Google users."

So, what is Google.com made out of? The search engine consists of a crawler, an indexer and a query handler. The crawler collects documents and makes "copies of the Internet," which now contains over 4 billion Web documents and over 880 million images. The indexer processes and represents the data, and the query handler processes user queries. These three components require Web servers to take the queries, index servers to store the names of the documents, document servers to store all of the Web documents, and ad servers to determine what advertisements to show based on auction money.

There are four main sources of failure that could occur in this type of Web server system. The hardware (e.g., disks), software, the network, or power could fail at any given time. Google deals with each of these types of failures through redundancy and replication.

At Google they expect hardware, especially disks, to fail on a daily basis. Rob Pike stated that the there is a "mean time to failure of three years for one machine, so for 1,000 computers, expect to lose one per day!" So, at Google, they expect to lose many more than one computer per day. With such a high loss rate and the need of multiple computers for storing the 10s of terabytes of the Internet, Google needs a lot of machines and disk space. It makes sense that to save money on the large number of computer purchases, Google chose to buy relatively cheap hardware. In the beginning of Google, when it was located at google.stanford.edu, it consisted of a few cheap PCs in a Stanford University computer lab. Shortly after, it moved to someone's garage, until it finally reached its current location in more sophisticated computer racks contained within multiple data centers across the world. Google's success story is unlike many others during the dot-com boom, during which time many startup companies spent all of their money on expensive hardware with "gleaming racks," while Google held their entire systems together with Velcro!

The PCs at Google are unreliable, cheap, and fast. In order to make these computers reliable, Google must use fault-tolerant software. Reliability, scalability, and load balancing are achieved by breaking resources into pieces and replicating everything. The software is aware of the redundancy structure and spreads things around to avoid a single point of failure. Google's PageRank system (a ranking that is based on the number of links to a given Web page) is used for ordering document names in the index. The index gets broken into "shards" based on its PageRank score. Higher-ranked pages will be replicated more than lower-ranked pages, so that the higher-ranked pages are less likely to be lost in a time of failure. Replication is done at the index level, the document level, and across data centers. When a query is sent, DNS resolves to 1 of *n* data centers (presumably the one closest to the sender). The load balancer then chooses one of the Web servers, which then sends the query to one replica of each index shard. Since the index is read-only and replicated across index servers, the search is done in parallel. This entire process takes approximately a quarter of a second. Google's underlying file system is abbreviated GFS. This file system is large and distributed and contains chunks of files on chunk servers (there are multiple chunks per chunk server). The chunks are replicated, often three times but more often if they are heavily used. The chunk servers act as a master and support automated fail-over. (For more information on GFS, see the GoogleFS paper in SOSP '03.) As seen here, reliable software is more important than expensive hardware in Google's case.

As mentioned, network and power outages can also occur. Rob noted that, unlike their reliance on cheap hardware, Google cannot operate effectively with cheap networking equipment, such as switches and routers. It is important to note that network or power failures only reduce capacity (the query request will merely time out and just needs to be reissued). When an entire

rack of servers was accidentally wiped clean, no data was actually lost—just the terabytes of storage capacity. The same automated software-recovery mechanisms are in place if there is an unreachable network or a power failure.

Throughout Google's existence, they have learned many valuable lessons. The best lesson is "failures will happen, plan for it and survive." When things break, they break too often for humans to fix, so there needs to be an automated, "self-healing" system in place. Another important lesson when dealing with commodity (cheap) hardware and components is that you need to use better software and be careful not to cut corners too much (i.e., two machines per power supply may seem good on paper, but "it is a false economy," since system restoration requires you to power down both computers when only one needs to be off). It is also necessary to make sure that there is adequate cooling for all of the computers. And, finally, Google needs to continue to improve by adapting their fault-tolerant techniques, software, and algorithms to newer and faster hardware. They continue to look for new architectures for redundancy, improve automated failure and recovery, and develop their new services (e.g., Gmail) around these principles.

FREENIX SESSION: FREE DESKTOP

Summarized by Brian Cornell

Glitz: Hardware Accelerated Image Compositing Using OpenGL

Peter Nilsson and David Reveman, Umeå University

Desktop computer users have many more demands today than they used to. Users expect features such as translucency, shadows, and transformations to be prevalent. Hardware manufactures have met these demands and provide graphics processors capable of performing these tasks at high speeds. Peter Nilsson and David Reveman introduced Glitz, which brings this hardware power to the user by providing an interface between OpenGL and the graphics library Cairo, where it can be used easily by developers.

The goals of Glitz's design are to create a system with efficiency, quality, and consistency. Glitz implements hardware features, including native off-screen drawing, image transformation, polygon rendering, clipping, gradients, and convolution filtering. Glitz also allows for seamless integration between 2D and 3D environments.

Both accuracy and performance in Glitz were compared against other rendering engines. Glitz was found to operate anywhere from 3 to 200 times faster than the XRender extension to X servers. Using Glitz, Peter and David have shown that Cairo can be very fast. Glitz is at http://glitz.freedesktop.org.

High Performance X Servers in the Kdrive Architecture

Eric Anholt, LinuxFund

Eric began by introducing Kdrive, a small X server written by Keith Packard. Since Kdrive is a smaller server, it is easier to change. Eric then went into some background about the capabilities of modern hardware and of current extensions to the X server. Finally, he introduced the Kdrive Acceleration Architecture (KAA), an acceleration extension to the Kdrive X server.

The KAA offers a few improvements, including compositing, blending, and an improved off-screen memory manager. Whereas other implementations are limited to off-screen memory of the same color depth and image width as the screen, the memory manager in KAA allows off-screen memory to be of any type and size. This memory manager also determines what should be kept in which type of memory using a system that keeps track of scores based on the use of buffers.

In his initial implementation, Eric was able to render anti-aliased text five times faster than previously. However, text using composite alpha for subpixel anti-aliasing was five times slower. Eric would like to implement X video, support for GLX and a fix for the composite alpha speeds in future versions. His work can be found at http://pdx.freedesktop.org/~anholt/freenix2004.

How Xlib Is Implemented (and What We're Doing About It)

Jamey Sharp, Portland State University

Jamey began his presentation by explaining how Xlib works. He introduced an abstraction between three layers in Xlib: transport, protocol, and utilities. The transport layer handles communication between the client and server, the protocol layer constructs requests for the server, and the utilities layer does everything else. The transport and protocol layers, though very important, are not a big part of the Xlib implementation. Xlib was designed long ago for the systems that were available then and has been added onto many times since, creating a system that is not well designed.

Jamey introduced a solution to this: XCB. XCB is an implementation of an X client library that is simpler and smaller than Xlib. XCB focuses mainly on the transport and protocol layers. The problem with XCB is that most X programs are written to use Xlib, and rewriting them to use XCB would be a major task. Thus Jamey needed a migration path to make this process easier.

Jamey first tried to implement this migration by reimplementing the Xlib API using XCB. The problem with that is that the Xlib API is enormous. After making little progress into the immense repository of Xlib, Jamey started over, this time going from bottom up. He began with a complete Xlib imple-

mentation, and gradually replaced the more crucial components with XCB equivalents. The result is that current Xlib programs can easily by migrated to use these smaller client libraries, and there is no noticeable change in performance. Jamey's work can be found at http://xcb.freedesktop.org.

USELINUX SIG

Summarized by Patty Jablonski and Todd Deshane

The FlightGear Flight Simulator

Alexander R. Perry, P.A. Murray

The FlightGear Flight Simulator is a graphics project that simulates flying aircraft in reality. Perry says, "It is not a game." This is an open source project that has been released under the GNU General Public License (GPL) for Mac, Win32, IRIX, and Linux 32 and 64 bit. The simulator is portable, modular, platform neutral, and uses advanced algorithms: "It uses models, not just guesses."

The FlightGear project was started in 1996 by David Murr. Today its worldwide developer community, which includes Perry, consists of 89 people and is still growing. This group is inclusive (goes beyond just software engineers) and is multi-disciplinary (includes both technical and nontechnical people). Perry states that beginners are welcome to join in the project's development as well. For more information on the version releases of FlightGear, visit http://www.flightgear.org/version.html.

FlightGear has many features that make the simulation as realistic as possible. The flight simulator has 3D aircraft and scenery as seen from the pilot's perspective in the cockpit. The lighting changes are realistic and the aircraft is complex. Various things affect the overall flight experience, such as an open window on the aircraft, weather conditions (clouds and smog), and temperature. Intentional impairments have been employed selectively, which makes the simulator more difficult to use but accurately depicts behavior and views. Real-life problems and subtle interactions that can distract or confuse the pilot have been accurately modeled. Many of these features, including instrumentation problems and forces of nature, are often reported as bugs, since people who use Microsoft Flight Simulator are not as familiar with the situations that real pilots face as seen in FlightGear.

The FlightGear implementation is modular and quite complex: "It takes a lot of code to make things behave badly." FlightGear uses networking for remote access and allows a flight instructor to adjust the pilot's settings without the pilot knowing. It uses XML to allow changes to the flight environment. Additionally, there is a property database that stores all of the scenery for the entire planet. A large amount of storage space is needed for this. The 3D graphics and audio are made with OpenGC and OpenAL. Third-party extensions to the flight simulator are generally done in Python.

To download the latest version of FlightGear, see http://www.flightgear.org. Related projects and links:FlightGear Aviation Training Device: http://fgatd.sourceforge.net
OpenAL: http://www.openal.org
OpenGC: http://www.opengc.org
PLIB: http://plib.sourceforge.net

Making RCU Safe for Deep Sub-Millisecond Response Real-Time Applications

Dipankar Sarma and Paul E. McKenney, IBM

RCU, or Read-Copy Update, is a reader-writer synchronization mechanism for the 2.6 Linux kernel. RCU is best for "read-mostly" data structures. Although this works well for most situations, real-time applications in the Linux environment are becoming more popular and are in need of quicker response times. RCU callbacks at the end of grace periods (between context switches) cause too much latency in these real-time applications.

First, it is important to distinguish between readers and writers. Readers can access old versions of files independently of subsequent writers. In this case, garbage collection is needed to remove old or invalid copies of the files. Writers, on the other hand, create new files and delete old ones atomically. Because of this, readers have little to no overhead, while writers have a substantial amount of overhead.

Real-time latencies are 800 microseconds measured under load. Measurements were taken with Andrew Morton's "amlat" tool. This 800 microsecond latency is too long for such applications as engine controls, where there is a need to have three degrees of control when measuring revolutions per minute (rpm). There are three ways in which Sarma and McKenney are trying to solve this latency problem.

One possible solution for the latency problem described here is "Per-CPU Daemon." The primary advantage of this approach is that it is transparent to users of "call_rcu( )." Disadvantages include proliferation of kernel daemons and tuning parameters.

Another potential solution proposed by Sarma and McKenney is called "Direct Invocation of RCU Callbacks." Advantages to this option are that there are no kernel daemons and no tuning parameters, and it eliminates "softirqs" for callbacks. Disadvantages are that it is not transparent to the user and it can cause problems if used incorrectly by the user.

Finally, the third option presented by Sarma and McKenney on dealing with real-time latency is "Throttling of RCU Callbacks." The main advantage of this option is that, like "Per-CPU Daemon,"

it is transparent to users of "call_rcu( )." Disadvantages of this method are tuning parameters and that the current implementation is based on iterations and not time.

The initial performance results show that all three approaches have similar, significant performance increases and little complexity. Sarma and McKenney conclude with their argument that RCU can be made safe for real time. Finally, they found that up to this point, "Throttling of RCU Callbacks" is the less intrusive of the two transparent choices. They note that other performance issues exist for real-time applications and that tools to identify problems other than latency are currently under development.

Making Hardware Just Work

Robert Love, Ximian

The problem with the Linux desktop is that Linux lags behind its competition in terms of hardware management. Instead of having to su to root and mount drives for a simple plug-and-play device, we want our desktop to be able to figure out what to do for us. "Think MacOS X simplicity," Robert Love says.

Robert Love is one of the main developers of "Project Utopia," an umbrella project focused on using the 2.6 Linux kernel's new features, sysfs and hotplug, along with HAL (hardware abstraction layer), udev (a user-level device management system), D-BUS (a message bus), and the GNOME Volume Manager. The goal of this project is to be clean and elegant without the use of kernel hacks, like "supermount." It is very important to "do things right."

The 2.6 Linux kernel does not provide central management as is. It also does not have a platform-agnostic daemon to take advantage of sysfs' device database and hotplug's ability to provide notification of when a new device is added. The additional components of Project Utopia are needed to provide this support.

HAL is a central repository of device information. HAL has a platform-independent interface and persistent key/value pairs, provides asynchronous notifications of changes to devices, and handles Universal Resource Identifiers (URIs) to access devices. The application programmer's interface (API) to HAL is "libhal." Using HAL instead of legacy code reduces the amount of code needed to handle devices significantly (thousands of lines of photo application code can be reduced to less than 10 lines of code with HAL).

On most current Linux distributions, the contents of /dev contains about 18,000 device nodes. Realistically, you only want to see a list of the devices that you currently have. The clean, elegant user-space solution to this is called "udev." /udev only lists devices that you actually have on your system and can be renamed for your convenience.

In order to tie all of this together, Project Utopia needed a message-passing system called D-BUS. The kernel can send out D-BUS messages of new devices. The kernel/D-BUS layer is used by the GNOME Volume Manager.

The GNOME Volume Manager, a manager of disk and other media volumes, allows you to automatically manage volumes, automounting or autoplaying new media/devices, like automatically playing a CD or DVD. It can also create desktop icons based on the type of device or media attached to your system.

Linux is made up of a lot of projects, which often lack integration. The Utopia Project developers hope to bring some unification and integration to the Linux desktop.

FREENIX SESSION: SECURITY

Summarized by Matt Salter

Design and Implementation of Netdude, a Framework for Packet Trace Manipulation

Christian Kreibich, University of Cambridge, UK

■ *Awarded Best Student Paper*

Solving a problem that involves manipulating network traffic often requires complex filtering, fine-grained and large-scale editing, and visualization. Finding well-maintained tools with the desired functionality is often a hassle and not always possible. Another approach is to write your own solution. While tools that allow editing of captured network traffic have been created, they are often not reusable at the API level, since their functionality is only available in standalone executables.

The network dump data displayer and editor, Netdude, is a framework for packet inspection and manipulation. Netdude has GUI and command line usage paradigms. It allows for scaling of trace sizes and is reusable at all levels, as well as being extensible.

A bottom-up view of Netdude's layered architecture is as follows: libpcap handles elementary trace file operations. libpcacpnav is a wrapper around libpcap that allows one to jump to arbitrary points in the trace file, identified by timestamps or offsets. libpcacpnav uses heuristics to get in sync with the packet stream. Above libpcacpnav is libnetdude, the core of the framework which makes the editing of large traces transparent. libnetdude is extensible through two kinds of plugins: feature and protocol. It provides per-packet TCP dump output, as well as an observer/observee API to inform the user of updates. The GUI is GTK-based and extensible through the same kinds of plugins as libnetdude, and

updates itself through libnetdude's observer API.

Handling of big trace files always involves limiting the number of packets in memory. Since it is not possible to simply use mmap() for inserting and deleting packets, trace files are edited at the granularity of trace areas, which are bounded by timestamps or fractional offsets. Modified trace areas become trace parts, which are flattened onto the original trace file when the file is saved.

Netdude has served as a mechanism to conveniently access suspicious network activity, create traces for network performance evaluation, edit honeypot traffic, and generate IDS signatures. There is much work left to do, including packet resizing and support of scripting environments. Help is welcome!

Trusted Path Execution for the Linux 2.6 Kernel as a Linux Security Module

Niki A. Rahimi, IBM

Trusted Path Execution (TPE) was originally a kernel patch to OpenBSD 2.4 created by Mike Schiffman. It was later modified for OpenBSD 2.8 and 2.9 by the Stephanie project.

Currently, TPE is implemented for Linux 2.5/2.6 as a Linux Security Module (LSM). TPE's notion of a trusted path is not to be confused with the more common concept of trusted path in a network context. In TPE, a trusted path is root owned and neither group nor world writable. A trusted user, root by default, is any user on the access control list (ACL) determined by the system administrator.

The TPE LSM performs a check upon execution of a file by utilizing the `tpe_bprm_set_security` hook in the LSM framework. Upon execution of a file, the module verifies whether the user and the path are trusted. The TPE ACL is modified via a sysfs pseudo file-system approach. A directory called tpefs is created containing two files, add and del. Users are added and deleted by writing their UID to the add and del files, respectively. TPE enhances security by preventing execution of untrusted code on the system. The check of path and user occurs exactly before execution is allowed, and if the user and the path are both untrusted, execution is denied.

In addition to trusting code in root owned directories, TPE LSM trusts code in directories of trusted users. TPE is part of the LSM patch as of 2.5.70. It is open to improvements as it is released under a dual BSD/GPL license. LSM, accepted as the current method to introduce security to the Linux kernel by the kernel community, is a small project with lots of potential, for which many more modules are needed.

Modular Construction of DTE Policies

Serge E. Hallyn, IBM Linux Technology Center; Phil Kearns, College of William and Mary

Domain Type Enforcement (DTE) is a mandatory access control system introduced in the 1970s by Honeywell, TIS. It assigns types to files and domains to processes. A domain is structured as a list of sets. One of these is the entry type set, which specifies through which types the domain may be entered. Another is the type access set, which specifies which types the domain can access. The signal access set specifies which domains the domain can signal, and the transitions set specifies which domains the domain can transition to. There are two types of domain transitions, auto and exec. When a process under some domain executes a file which is an entry point to another domain, either it must switch to the new domain (auto) or exercise the default option of keeping its domain (exec).

A DTE policy contains lots of domains, types, and defaults. The policy module files presented in this paper are a collection of domains, types, and group definitions, as well as the access rules pertaining to them. Type definitions consist of both path assignments and access grants. Both type and domain definitions may contain assert statements that are used for maintenance of policy constraints, which are interpreted and enforced by a policy consistency class. Since domains and types can declare conflicting access rules, priorities for the access rules are defined. These priorities are determined by placing specific rules over general rules, inbound access rules over outbound access rules, and use of the "absolute" keyword. Group definitions facilitate more generic modules. These are achieved through either the keyword "all," binding a group name to a set of domains or types, or namespace globbing. Groups are expanded only at time of reference and can be dynamically extended.

Modules interact with the system by obtaining system-specific data. They can also be moved between systems and shipped with software. Modules are loaded into the DTE LSM module through a configuration file which is generated by a script that takes a list of module files as input. Work related to this project includes DTEX by Chuck Fox, Fedora, the IBM research project Goyko, and Tresys. This project still needs to be applied to SELinux, which would require object classes and fine-grained permissions, and the modules need to be distributed. Future work also includes possible improvement of the priority specification.

USELINUX SIG

Summarized by Martin Michlmayr

Building and Maintaining an International Volunteer Linux Community

Jenn Vesperman, author and consultant; Val Henson, Sun Microsystems

Val Henson shared her insights about creating a volunteer community based on her experience with the LinuxChix project. LinuxChix is an international community whose focus is to create a friendly, predominantly female Linux community. The project was founded in 1999 by Deb Richardson, who created a Web site, mailing lists, and a logo. By 2001, Deb was burning out as the LinuxChix project struggled to remain active. At that time, Jenn Vesperman was chosen as the new coordinator.

Henson summarized the lessons she learned about building and running an international volunteer community, grouping them into three categories: First, the social category, where she suggests that you have to build a sense of community; second, there are organizational aspects, which boils down to delegation; the third category is the technical one, and Henson emphasized the importance of using technology that distributes well.

Henson suggested that building a sense of community was fairly easy for LinuxChix, because building community was the goal of the project. Women are quite excited to find other women who share their interests, and they create bonds fairly easily. One interesting observation Henson made was that being friendly and nice does not attract incompetent people. While LinuxChix' explicit goal is being friendly to its members, there is some hostility in other projects to new members, possibly in order to have a high barrier to joining in order to keep incompetent contributors away. According to Henson, this strategy does not work; being friendly is the better approach. To help create community, LinuxChix uses specialized mailing lists, some of them focused and technical but others allowing completely off-topic discussions. A member-only posting policy on some lists increases the sense of community.

Henson emphasized the importance of delegation as part of organization. She suggested that the first coordinator of the project burned out because she took on too many tasks herself; Henson therefore jokingly said that the number one rule is to do nothing yourself. Instead, delegate to other people; once a task has been delegated, let go and don't interfere. It is also important to give credit. One task of the main coordinator is to monitor the health of other volunteers, and to act accordingly—for example, by sending an overworked volunteer on vacation and by finding more volunteers to help out. Finally, she also suggested that rules should be kept to a minimum—rules will drive good people away, and trolls won't abide by them anyway.

Indexing Arbitrary Data with SWISH-E

Josh Rabinowitz, SkateboardDirectory.com

Josh Rabinowitz introduced SWISH-E, a simple Web-indexing system for humans. He summarized the tool as being a fast, powerful, flexible, free, and easy-to-use system for indexing collections of Web pages, and suggested that the definition would not be complete if any of these adjectives were removed. SWISH-E is based on Kevin Hughes' SWISH project from 1994 and is now maintained by Bill Moseley. The tool is written in C and creates binary indexes. There are C, Perl, and PHP interfaces.

There are several alternatives to SWISH-E, such as htdig and MySQL, but Rabinowitz argued that SWISH-E has several attractive features. It is fast and robust, and the tool undergoes steady development and bug fixes. There is good and extensive documentation, as well as a lively and informative mailing list. Finally, SWISH-E offers a "bulk insert" method which doesn't presume to know how or what you want to index. SWISH-E currently handles XML, HTML, and text, but there are two methods for dealing with arbitrary files. First, an external program can be written that converts a file to a format SWISH-E understands. Second, a FileFilter for each given file type can be created. This approach is more modular, but it's slower, since it invokes a child process for each file.

Rabinowitz showed in some examples the ease of creating indexes with SWISH-E and introduced SMAN (http://www.joshr.com/src/sman/), a tool to search UNIX man pages that is based on SWISH-E. Finally, Rabinowitz summarized some future development. The 2GB size limit should be removed soon, and UTF-8 support is a major feature that is being worked on. The ranking system should be rewritten, and the main developer of SWISH-E is interested in working together with a graduate student who would like to pursue such a project.

FREENIX SESSION

Demonstration: Croquet, a Networked Collaborative 3D Immersive Environment

Dave Reed, HP Labs

This demonstration of a 3D networked, collaborative Croquet environment is considered a "research project, not a product," says Dave Reed, project developer. It demonstrates a peer-to-peer networked application that supports collaborative computing and scalable computation.

The demonstration used two PCs networked together on the same switch (the network is meant to

provide different views of the same world), but Reed claimed that the environment can support many more peers. There were some networking problems during the demonstration which made one of the views update too slowly or inaccurately show the world. Because of this, Reed presented the environment in one perspective only.

The environment showed clouds on a 3D plane with windows or "portals" on it. These portals acted as hyperlinks or mirrors to other worlds. Dave went from portal to portal showing us what each had inside. One portal showed a recursive pyramid and another had water that rippled when the mouse was moved over it. Still other portals showed images of people, a chess board, and a flag with a spring and mast to show how the flag changes when moved with the mouse.

He entered into one portal that brought us to an underwater world. He explained that with this scenario, you can "change the laws of physics" by making heavy objects float or have whispers only be heard by certain people across the room. In this underwater world, Reed was represented as a fish. He used a Paint program to draw a new character on the fly and then added this character to the world. The funny part was that there were large help signs in the water explaining "how to make a fish" or "how to make your fish swim" for the beginners. He then left this world and showed a vast world with waterfalls and trees in it that he called a "traditional video game world."

This 3D Croquet environment was implemented using OpenAL and OpenGL for its audiovisual features. It works with communicating objects whose messages are replicated or "cloned"; most messages do not go over the network (almost all computation is local). Objects are governed by mouse movements and positions so that the appropriate action is taken. This system is considered real time, and it is therefore very important that each machine in the network has the correct synchronized time.

In the future, Reed would like to have this Croquet environment implemented for small devices, such as cell phones. He also hopes to develop a security model for the system. There is an important unanswered question: What should people be allowed/not allowed to do in this environment (what is considered cheating, what should be allowed to be read/written, how much should you be allowed to see?)?

This Croquet project is approximately nine months old and is being developed in partnership with the University of Minnesota and the University of Wisconsin. The project is scheduled to be released as open source in an open and public forum.

USELINUX SIG

Summarized by Adam S. Moskovitz

Linux and Genomics: The Two Revolutions

Martin Krzywinski and Yaron Butterfield, Genome Sciences Centre

The session started off with Martin Krzywinski, from Canada's Michael Smith Genome Sciences Centre, talking about Linux and genomics, their near-parallel rapid advancements, how Linux is used in genomics, and how genomics has independently adopted many of the same "ideals" as the Linux community.

Martin started by discussing what I believe are the most significant parallels between the Linux and genomics communities, namely, openness and innovation. Just as the Linux community encourages people to build useful things and give them away, the genomics community does that not only with software tools but with the data itself. The most well-known example of this is the human genome project, where the completed genome was uploaded to the public databases pretty much as soon as it was ready. Most public genomic sequencing centers submit new data to Genbank (a public repository of sequence information) pretty much as soon as it is collected.

The genomics community, like the Linux community, actively contributes software to the public on a regular basis. A genome browser from USCS and the Ensembl browser/data miner/visualizer are both freely available. Jim Kent's genomic assembler, which was instrumental in the public effort to complete the human genome, is also freely available. Finally, Lincoln Stein has contributed numerous CPAN modules, both for genomics and such commonly used modules as the Perl interface to Tom Boutell's libgd and Stein's own CGI.pm.

Using these and other public tools, the Smith Centre was the first to publicly release the full sequence of the coronavirus believed to be responsible for SARS (Sudden Acute Respiratory Syndrome). This was accomplished in just five days, using an eight-way Linux system. Krzywinski shared some of the feedback the center received; much of it was positive but some wasn't. One person wrote:

Subject: You have to be NUTS!

My daughter doesn't think its such a good idea to have the gene sequencing for the new coronavirus on the internet. I don't either! There should have been a better way! You must be crazy!

[Summarizer's note: I suppose some people feel the same way about making the Linux source code public.]

By the way, Martin Krzywinski gets my award for most interesting slides: black drawings on a red

background and the funkiest font I've seen in a long time.

Thin Client Linux, a Case Presentation of Implementation

Martin Echt, Capital Cardiology Associates; Jordan Rosen, Lille Corp.

In this presentation, Martin and Jordan described how Martin's medical practice decided to install a Linux-based thin client instead of Windows PCs and how that decision has worked out.

Martin started by describing their practice (with more than 200 employees total, 40+ doctors, seven offices, seven hospitals, in New York and western Massachusetts), their work load (128,000 patient visits, 800 open-heart surgeries, 380,000 services billed per year for $22 million in revenue), and their MIS needs (billing, storing test results, financial planning and analysis, payroll, medical imaging, calendar, email, word processing, and more).

Martin then proceeded to give a fairly detailed cost-benefit analysis of "thick" Windows versus thick Linux versus thin Linux. While their initial outlay was about $15,000 more for thin Linux, subsequent savings more than made up for that difference. Specifically, for Martin's practice, their initial outlay per workstation would have been about $3,000 for Windows compared to about $2,100 for thin Linux. Their first-year operating costs showed a similar savings: $2,800 vs. $1,300. With 200 workstations, the savings from choosing a thin Linux client were clear. The last savings was in significantly reduced support costs for remote sites: because almost everything was done at the central office, and because hardware maintenance was mostly reduced to swapping out bad machines (which could be done by an employee with no special skills), their remote maintenance costs were reduced to nearly nothing.

Martin also pointed out several other benefits of thin Linux, chiefly, that employees were more productive. With Windows, too many applications could be customized and employees spent too much time doing this with no real gain in production; with Linux it was easier to disallow such customizations. Another benefit was that applications could be customized to prevent user-caused "outages." The most obvious example was preventing users from closing an application without properly quitting; they simply removed the "X" button from the menu bar! The last of these savings Martin mentioned was preventing employees from using the computers for personal use (things like playing solitaire, downloading music files, and setting fancy screen savers). He estimated that at 15 minutes per day per employee, such wasted time cost his company over $110,000 each year.

Jordan then took over and presented the technical side of things. The first thing he mentioned was that some applications could not be made to work under Linux; for these the practice kept 10 "thick" Windows systems and set up a single Windows 2000 Server system for data storage; Samba was used for logons and drive mappings. Next, Jordan discussed the high and low points of the software used on the thin Linux client: OpenOffice worked quite well, but other applications (Evolution and Mozilla) had a few problems, such as tending to crash or not handling certain required functions (some Web sites, calendaring). There were some problems with low-level things such as file permissions and lack of file locking in OpenOffice.

On the whole, user acceptance of the thin Linux client was high, and the practice has been running for 300 days without a single server crash, the network has never gone down except from human error, the remote desktop (via VPN) has yet to fail, and no viruses or worms have affected their network.

Towards Carrier Grade Linux Platforms

Ibrahim Haddad, Ericsson Research

The third talk of this session was what appears to be a refereed paper, written and presented by Ibrahim Haddad from Ericsson Research on "Carrier Grade Linux"—that is, a Linux operating system capable of being used in servers and switches in a public telecommunications network. Typically, such servers require 99.999% reliability (less than five minutes of downtime per year), and switches require 99.9999% (less than 30 seconds downtime). Obviously, no version of Linux is there yet, but Haddad's talk summarized what is needed to get there, as well as what features will be required by carriers before Linux could be used to replace existing, proprietary systems.

Haddad presented an overview of the groups (committees, working groups, associations) working on this problem: The PCI Industrial Computer Manufacturers Group (highly available hardware), the Carrier Grade Linux Working Group of the Open Source Development Labs (Linux improvements), and the Service Availability Forum (defining high-availability APIs). Haddad works with the CGL working group, who released their first public draft in May 2004.

The remainder of Haddad's presentation covered three services not found in the stock Linux release that would be required for mission-critical environments. The first service was TIPC (Transparent Inter-Process Communication), an intra- and inter-cluster protocol that provides a framework for supervising and reporting topology changes. TIPC has been used by Ericsson for several years now and has been available as open source since February 2003. The second service, DigSig (Distributed Digital Signature), is part of the larger Dis-

tributed Security Infrastructure (DSI) initiative. This service allows an administrator to embed digital signatures in ELF binaries and adds functionality to the Linux kernel that prevents unsigned or badly signed binaries from executing. DigSig has been available as open source since January 2003. The last service is a package that implements native support for asynchronous events in the Linux kernel. Carrier grade platforms must process huge numbers of events quickly and efficiently, and this package implements the first tier of such services. It was also released as open source in January 2003.

PLENARY SESSION

Summarized by Martin Michlmayr

The State of the Spam

Eric Allman, Sendmail, Inc.

Eric Allman opened his speech in a very funny way when he analyzed the way talks on spam traditionally work. They first summarize what spam is all about, mention that spam is bad, go on to say that the situation is really bad, and finally claim that their product will solve all spam problems. Allman did not go this route, and while he suggested several ways to combat spam, he also made it clear that it will take years to come up with effective solutions and that everyone has to work together.

In the beginning, Allman gave some statistics and summarized claims about spam. Apparently, there are about 90 "world class spammers" who pay US$100,000 per month for bandwidth and servers. According to SpamHaus, 200 spam operations account for 90% of all spam. Spam costs mere microcents per message, which is why spammers continue to operate even though AOL rejects 80% of all incoming mail as spam.

Allman proceeded to summarize existing and new technologies used against spam: realtime blackhole lists (RBLs), content filtering, and challenge-response. RBLs are controversial because their false-positive rate is pretty high. Content filtering comes in different classes: heuristic filters work by observing what spammers are doing and creating means to detect and counter them. Unfortunately, this leaves us in a reactive mode; they send spam, we adapt our tools, and in the meantime we suffer from spam. Fingerprinting and collaboration store a fingerprint of a spam message so other people can test the fingerprint and discard spam. Again, this method is reactive and Allman suggested that it is only effective when the fingerprint database is updated every 15 seconds! Machine learning filters let the computer figure out the interesting stuff. This method needs two piles of "training data": spam and not-spam. While this method works fairly well for individuals, this is less the case on the server level, since legitimate mail varies a lot depending on the user.

Newer methods which are currently being worked on are traffic analysis, identity authentication, and economic schemes. Traffic-based filtering observes typical traffic patterns. For example, a host that normally sends 100 messages in a month and suddenly sends millions in a few minutes is very suspicious. One possible way to use this is to greylist a host and slow down the connection significantly. Identify-based filtering almost always requires authentication. You can use allow-lists and lock-lists in order to reduce the amount of resources spent on more expensive spam checks. There are two philosophies: everything not explicitly illegal gets through (default to accept) or all not explicitly legal gets blocked (default to deny). Sender authentication is not an anti-spam solution in and of itself, but it is essential for identity-based algorithms. We already have SMTP AUTH and TLS, but both are MTA-to-MTA, not end-to-end. Per-user authentication would be possible with PGP or S/MIME. Finally, there are economic schemes which shift costs from recipient to sender. A very small cost doesn't hurt usual senders (perhaps 100/day) but does hurt bulk senders (millions/day). These systems do not necessarily have to be cash-based since the credit can come in a different form.

At the end of the talk, Allman made several predictions. First, he suggested that spam will never go away completely. Authentication will help but won't solve the problem by itself. He thinks that spam will be "manageable" within two to three years, and that legislation will scare away bit players, but not large commercial spammers.

FREENIX SESSION: SOFTWARE ENGINEERING

Summarized by Brian Cornell

Managing Volunteer Activity in Free Software Projects

Martin Michlmayr, University of Melbourne

Martin is a member of the Debian GNU/Linux team and brought his experience with free software projects to the community. The main problem he introduced was that volunteers will sometimes neglect their duties, and it is hard to figure out when they do. For small projects this can mean that the project dies because nobody finishes it. For large projects this means that the quality of the product suffers and there are delays in the release of new versions.

Martin went on to describe how Debian is organized and what they do about this problem. At Debian there isn't a hierarchical management structure, so developers aren't supervised by a manager. Therefore they have to carefully look through hundreds of developers to figure out who is neglecting to do what they should. They find these people

in many ways: for example, when there is a bug in a package that is release critical or when a newer version of the software a package provides is available. Every once in a while they compile a list of people who appear to be neglecting packages.

Once you know who is not doing what they should, you have to do something about it. Kicking someone off of a project unnecessarily is not a good idea, because they could provide a lot of help for your project, and they may not be easy to replace. Debian contacts maintainers asking them if they are still active, and gives them two to three weeks to respond. If they don't respond, they're contacted again and given more time to respond before they are eventually removed. Because these people are volunteers, Debian cannot be overly demanding of them, and therefore a polite system like this is necessary.

Martin also points out that you can try to prevent the dereliction problem by having redundancy throughout the project.

Creating a Portable Programming Language Using Open Source Software

Andreas Bauer, Technische Universität München

Andreas Bauer's talk would have been very welcome in any class on compilers. He gave detailed information about the use of gcc to create new programming languages. Gcc, the Gnu Compiler Collection, has support for many different languages, and independently has support for many architectures. Andreas presented the capabilities of gcc through a simple expression language he called Toy.

Andreas talked about the current design of gcc's programming language interface. Gcc uses trees to express the language, and then generates an intermediate language called RTL based on these trees. Programming language interfaces are responsible for generating these trees during parsing. Unfortunately, these trees are somewhat tailored toward C and don't make concepts such as tail recursion, garbage collection, and scope representation easy. For this reason, a new system called SSA is being developed with generic trees and more optimization.

FREENIX INVITED TALK

Summarized by David Reveman and Peter Nilsson

Current GTK+ Development

Mattias Clasen

GTK+ is a multiplatform toolkit for creating graphical user interfaces with excellent internationalization support. GTK+ was initially developed for and used by the GIMP, the GNU Image Manipulation Program. Today GTK+ is used by a large number of applications and is the toolkit used by the GNU project's GNOME desktop. It can be used with a wide range of programming languages.

Mattias described the different components GTK+ is based on. Glib is a low-level core library that provides data-structure handling for C, portability wrappers, and interfaces for such runtime functionality as an event loop, threads, dynamic loading, and an object system. Pango is a library for layout and rendering of text, with an emphasis on internationalization. The ATK library provides a set of interfaces for accessibility, and the GDK library provides a layer of abstraction that sits between GTK+ and the underlying windowing system.

The talk briefly covered the additions to the 2.4 release, like the new, much improved file chooser and the new combo box. He mentioned that current maintenance of the 2.4 release is mainly directed to bug fixing and performance improvements. It is very complex to fix bugs in such a widely used toolkit without breaking backwards compatibility. Performance improvements have been made primarily in three areas: predictive exposes, reduced flicker by unsetting the background, and reduced signal emission overhead.

Current goals for GTK+ are to provide a full platform, close gaps to higher-level software layers, sanitize the GNOME library stack, keep up with evolving UI needs, and maintain binary compatibility.

The 2.6 release is planned for December 2004 and will contain solidified 2.4 add-ons as well as other smaller additions. The new file chooser will work well in 2.6; improvements include shared settings with Nautilus, automatic shortcuts, recent files, and the ability to choose file formats in the Save dialog. Some missing features will be added to the combo box, including separators, scrolling, and so-called insensitive items. New additions to 2.6 will also be made in areas of command line argument parsing, an icon list widget, a progress cell renderer, and some widgets from libgnomeui. Support for rotated text has been added to Pango, and more work will also be going into Pango.

Mattias talked a lot about what we can expect to see in the future of GTK+. Some of the planned changes are a new rendering model, support for RGBA visuals, an improved theme system, a built-in printing system, and full introspection. A big change that will happen to GTK+ is the introduction of a new rendering model. This will be accomplished by moving to the Cairo library for rendering. Cairo is a modern 2D graphics library with a PostScript-like API. It has capabilities similar to Java 2D, SVG, and PDF 1.4; alpha-compositing is a natural part of Cairo. Cairo has output back ends for X, OpenGL, local image buffers and PostScript. Support for RGBA visuals will be added to GTK+ and will make translucent windows, fade-in effects for menus, and drop shad-

ows well supported. The motivation for a new improved theme system is the desire to remove GTK+ dependencies, fully support Cairo's rendering model, and include layout access in the theme system. A full theme system specification should be made available and will most likely use a standard syntax like XML's CSS. The built-in printing system will include appropriate printing dialogs and will be based on Cairo, with back ends for CUPS, lpr, and GDI. Introspection is useful for language bindings, documentation, and IDEs. GTK+ already supports introspection of type hierarchy, properties, and signals, but not yet of virtual functions in class structs and library functions, which will be added in the future.

EXTREME LINUX SIG

Summarized by Matt Salter

A New Distributed Security Model for Linux Clusters

Makan Pourzandi, Open Systems Lab, Ericsson Research

The target applications for distributed security are large distributed applications with a large software base that provide around-the-clock service and require high availability (99–99.999% uptime). The model presented in this paper specifically targets Linux clustered servers and is intended for servers exposed to the public, providing services to different operators, and running untrusted third-party software.

Distributed security has several requirements. One is security isolation, or compartmentalization. This is needed because exploitable vulnerabilities are probable in a large software base; without compartmentalization, a single vulnerability could expose the entire system. Runtime changes to the security context must be possible and reflected immediately, and application-layer security cannot be relied upon, since administrators must then contend with vulnerabilities in applications over which they have no direct control. The challenges of distributed security include creating a coherent implementation that does not leave any security gaps, integrating different security solutions from different vendors, and managing the system to prevent misconfigurations and inconsistencies.

Because most of the target applications have only a few users with whom everything is done, a security policy based on process is needed. At the node level, such security is achieved through mandatory access control. The model presented in this paper extends mandatory access control to the entire cluster. Processes are assigned a unique security ID (ScID), assembled from the ScID of the binary (stored in the ELF header), the ScID of the parent process, and the node security ID (SnID). To achieve compartmentalization, virtual security zones are set up inside the cluster. Security zones are groups of ScIDs and SnIDs. The distributed security policy allows for access control decisions on the process level based on the IDs of the source and target processes. Network, socket, and transition rules also exist. The architecture of the distributed access control implementation is as follows: each cluster has a single security server and each node has a security manager. The security policy is propagated from the security server to the security managers, which enforce policy at the node level via secure communication channels.

This model is not intended to replace existing security solutions, but, rather, to serve as an add-on to them. Challenges include creating a comprehensible and acceptable security policy and explicitly defining security zones in the distributed security policy.

Implementing Clusters for High Availability

James E.J. Bottomley, SteelEye Technology

A "highly available" (HA) system is any system that takes action to increase availability beyond what would ordinarily be possible. HA clusters consist of multiple networked local machines with some type of shared storage. There are three types of HA clusters. The simplest type is a two-node-only cluster, which cannot be scaled. A second type is the quorate cluster, which is centrally controlled and will not work without a membership service. A quorate cluster is defined such that no other cluster may be formed from excluded nodes, which means it cannot be split into two clusters. If the cluster is split, the majority of the nodes survive. The final type is the resource-driven cluster, in which resources are grouped by which services they belong to. In a resource-driven cluster, a node must simply establish ownership of a group to export the service. Resource-driven clusters also allow independent subclusters to form. The simplest of these cluster types is two-node-only, followed by resource-driven, and the far more complex quorate cluster. Recovery is much faster in resource-driven clusters than in quorate clusters.

Determining availability is difficult because you need to know what the system's uptime and downtime are in your environment. While duplication of nodes allows you to determine downtime, it does not allow you to determine uptime. Uptime can only be controlled through careful implementation and deployment of the cluster. However, whether availability or downtime is significant depends on the type of service being offered.

Often, it is the application that fails instead of the server. Monitoring applications is important so that application failures can be spotted

and corrected. Local application recovery is important as well, since applications fail more often than nodes, and local recovery decreases downtime and minimizes disruption. Also, monitoring for failures in general is important, since while redundancy protects your system from the first failure, the second failure will take your system down.

Uptime can be improved by assessing cluster hardware and eliminating single points of failure (SPOFs). Clusterwide SPOFs should be eliminated entirely, while individual-node SPOFs should be evaluated to see if eliminating them would improve uptime. In a shared storage cluster, the real SPOF is storage and should be addressed through replication by making sure that the external array is configured as RAID 1. Power supply, mechanical devices, and the connection to the storage are the node SPOFs. One way to eliminate the connection to a storage SPOF is to have multiple connections to the storage from a node. This is called a multipath cluster.

The biggest Linux-specific problem faced by cluster manufacturers with binary modules is simply keeping up with the kernel patches and releases. Another problem is the dreaded "oops," which kills kernel processes and then tries to continue. If the kernel was in a critical section at oops time, the system may hang. Large Block Device support (LBD) is a Linux feature that helps clusters. It is limited to 2TB in the 2.4 kernel. Multipath solutions are different for every vendor in the 2.4 kernel, but an attempt is being made to unify the architecture on Device Mapper for 2.6.

FREENIX SESSION: SYSTEM BUILDING

Summarized by Brian Cornell

KDE Kontact: An Application Integration Framework

David Faure, Ingo Klöcker, Tobias König, Daniel Molkentin, Zack Rusin, Don Sanders, and Cornelius Schumacher, KDE Project

Cornelius Schumacher presented Kontact, a Personal Information Manager (PIM) for the K Desktop Environment (KDE). Kontact was designed to integrate individual components such as Kmail, Korganizer, Kaddressbook, Knotes, Knode, and Kpilot. The developers wanted an interface with which all of these programs could be used together, without maintaining the separation between the individual projects.

Kontact was designed with the basic goal of keeping all of the components in it as separate as possible without the user being able to tell. To satisfy this, the components had to be integrated on an application level and still be able to run alone. But to maintain the semblance of integration, the components needed an integrated UI, inter-component communication, and shared settings.

With these constraints in mind, the KDE Kontact team designed Kontact to use plugins from each application with a Kpart for the user interface. The components then communicate using DCOP. Using the Kparts—basically component versions of the applications—Kontact embeds each component into a unified Kontact user interface. The components can also use a unified configuration through Kconfig. Kontact is an ongoing project: http://www.kontact.org.

mGTK: An SML Binding of GTK+

Ken Friis Larsen and Henning Niss, IT University of Copenhagen, Denmark

Henning Niss presented mGTK, a binding to the GTK+ graphics toolkit for the Standard ML language. The goal of this project was to provide SML access to a good general-purpose toolkit. Keeping this in mind, the developers wanted a direct binding to the C interface of GTK; they wanted it to work under any SML compiler, and they wanted compile-time type checking. Other interfaces to GTK only give errors at runtime, making it harder to fix bugs and optimize programs.

SML is a functional language with a formal definition. It is separated into two parts: the core language and the module language. There are many implementations of SML, and the mGTK developers targeted two of them, Moscow ML and MLton. They used a system of type constraints, including what are known as phantom types to enforce type checking.

Using mGTK, GTK+ classes are translated to SML signatures. Class types are represented as SML types, and methods are implemented as functions. mGTK can automatically generate the SML binding based on the gtk.defs file that comes with the GTK API. mGTK is available at http://mgtk.sourceforge.net.

Xen and the Art of Repeated Research

Bryan Clark, Todd Deshane, Eli Dow, Stephen Evanchik, Matthew Finlayson, Jason Herne, and Jeanna Neefe Matthews, Clarkson University

Repeated research is a process often used to verify results in scientific research. Jeanna, Stephen, and Todd presented an application of this process to the world of computer science research. As a class, they tried to reproduce the tests in the paper "Xen and the Art of Visualization." They wanted to see if they could get the same results, and to apply more tests to Xen in hopes of further examining its performance.

Reproducing the environment in which the original Xen tests were run was not easy. They had to first obtain the same hardware and then install the same software used in the original tests. The Xen authors listed the benchmarks they used, making it easy to reproduce those, but assembling and running them was time-consuming. Also, some of the benchmarks used were closed benchmarks, so they had to be replaced with similar open source alternatives. The result of all of the work was that their repeated measurements were within 5% of the original measurements.

The team applied many other tests to Xen: they tested its usability as a set of virtual Web servers; they tested it on commodity hardware, rather than the server machine that the original tests had been run on; and, finally, they compared the performance of Xen to that of an IBM zServer. They learned from this that repeating research is not easy, but it is an important reality check in the development of new technologies.

EXTREME LINUX SIG

Summarized by Bill Bogstad

Scaling Linux to Extremes: Experience with a 512-CPU Shared Memory Linux System

Ray Bryant, John Baron, John Hawkes, Arthur Raefsky, and Jack Steiner, Silicon Graphics, Inc.

Ray Bryant spoke about SGI's Altix Itanium 2-based HPC (high performance computing) servers. Non-shared memory computing clusters are frequently talked about today, but SGI believes that NUMA (non-uniform memory access) shared-memory compute servers remain appropriate for many HPC applications. SGI's Altix systems are architecturally similar to their MIPS-based Origin 3000 servers. The basic building block of an Altix system is a computing brick that has two pairs of Itanium 2 CPUs. Each pair of CPUs can have up to 16GBs of dedicated local memory. Bricks are connected using SGI's NUMAlink technology, which supports cache coherency and uses specialized routers. Altix systems have achieved records on a number of HPC benchmarks, including SPEComp L2001 in June 2004. The underlying interconnect technology supports up to 2048 CPUs, but SGI currently only supports 256 (soon 512) CPUs in a single SSI (shared system image) under Linux.

SGI believes that porting of single CPU applications to an SSI system can be much easier then porting to a computing cluster since non-performance-critical code can be left as non-parallel. When SGI decided to develop a NUMA system using Itanium CPUs, a Linux port to the Itanium was already available and it was decided that it would be easier to start from this port than to move IRIX from MIPS to Itanium. The current goal is that if an application runs on a generic Itanium under RedHat AS 3.0, then it should run on an Altix system.

However, even getting the 2.4 Linux kernel to run well on the Altix hardware has been an interesting challenge. The kernel had to be taught the performance differences between local and remote memory. On a 512-CPU system this is critical, since only 0.4% of total system memory is local (i.e., fast). A new round-robin buffer cache page allocation algorithm is used to avoid having a brick fill up all of its local memory with cached pages, which would leave no local memory in which to run applications. An O(1) scheduler was added with the elimination of a global runqueue lock and a resulting sixfold improvement on some benchmarks. Elimination of system global variables in favor of per-CPU variables and value aggregation as required was needed to support very large systems. Without these changes, the system would spend all of its time pounding the cache coherency hardware just to keep system status variables updated. A number of kernel hash tables are sized at O(1%) of total system memory, which is more memory than exists at any one brick in the system. These tables are now spread out in the same way that the buffer cache is.

Changes were also made to allow system operators effective use of the system. The dplace command allows the operator predictable memory and CPU allocation to the threads in a single process. By specifying the appropriate parameters, performance can be enhanced by taking advantage of knowledge of the memory access patterns of the various threads in a process.

Looking forward, many of SGI's changes have made their way into the 2.6 Linux kernel. As a result, a generic 2.6 kernel will boot on an Altix system. As SGI moves to supporting kernels based on 2.6, they expect improved scalability and the ability to support larger systems.

Quantian: A Single-System Image Scientific Cluster Computing Environment

Dirk Eddelbuettel, Debian Project

Quantian is a Linux distribution that is focused on cluster-based scientific computing. It was first released in March 2003 and has gone through a number of major releases since then. The latest releases can no longer fit onto a single CD and now require a bootable DVD or booting from a hard disk. Quantian's lineage can be traced back to the popular Debian distribution. The path is from Debian to Knoppix to cluster-Knoppix to Quantian. From Knoppix, it inherits read-only media-based simplicity and automatic hardware detection, along with support for persistent data on USB storage devices. clusterKnoppix adds zero-configuration OpenMosix clustering with automatic process migration along with the cluster-compatible Mosix File Sys-

tem. A single machine can be booted from Quantian media and then other machines can network-boot via the PXE protocol and form a single Mosix cluster.

Quantian extends clusterKnoppix with a large number of scientific computing applications. In particular, Beowulf-style clustering tools and libraries are included along with the statistical package R and the SNOW extensions. SNOW allows easy access to high-level parallel statistical computing. Some Knoppix packages that are not related to scientific computing or related software development have been dropped in order to make room for Quantian's scientific computing additions.

Currently, Quantian is essentially a one-man operation maintained by Dirk. He responds to requests for the addition of new packages as time and interest allow. Distribution size, network security concerns, and surveying users for their needs and configurations remain open issues for him. Even though it is primarily a repackaging of other components, Quantian deserves a look if you are interested in scientific computing. At the end of his talk, Dirk mentioned that the laptop he was using was running Quantian with a USB flash drive for persistent storage. It seems that his employer will not let him install Linux on the company-supplied laptop, so he has found another way. Let's hope Dirk keeps finding another way.

Cluster Computing in a Computer Major in a College of Criminal Justice

Boris Bondarenko and Douglas E. Salane, John Jay College of Criminal Justice

John Jay College is a specialized liberal arts college within the City University of New York system. It offers degrees in Law and Police Science, Fire Science, and Forensic Science among others. So, you might ask, just what kind of cluster computing is needed in a College of Criminal Justice? Douglas Salane made it clear that there are a number of areas where significant computing resources can be helpful.

Current and planned projects include simulations of the fires that occurred after the attack on the World Trade Center, database analysis and data mining of the FBI's National Incident-Based Reporting System, and molecular modeling for toxicology studies.

John Jay College has a relatively small cluster-computing facility. The compute cluster consists of 12 nodes with two CPUs each. A separate database cluster has four nodes, and the computing laboratory has 30 Linux workstations. Still, they had to go through much of the same decision-making processes that larger facilities might go through. Blade/rack systems or piles of PCs? What network file system to use? What interconnect technology? How to manage and monitor the cluster? How to test the correct functioning of the cluster? A cluster-specific Linux distribution or self-configuration?

Verifying the correct functioning of the cluster was of particular concern to Douglas. This concern was strengthened when the test software that is included in the BLACS portion of the ScaLAPACK software library reported incorrect results for some of its tests. In the end, the error was traced to a faulty Gigabit Ethernet card in one of the machines. Other cluster-computing packages don't always provide those kinds of tests. On the other hand, ScaLAPACK can be difficult to use.

For a small site, just figuring out what cluster-computing software is available and how to set it up is a significant undertaking. Unfortunately, Linux distributions, like the previously mentioned Quantian, were not available when they first started working on their cluster. Support for heterogeneous clusters would also help by allowing them to expand the size of their cluster over time without sacrificing performance to the demands of optimizing software to the lowest common denominator.