;login:

THE MAGAZINE OF USENIX & SAGE

October 2001 • Volume 26 • Number 6

inside:

CONFERENCE REPORTS

USENIX 2001 Annual Technical Conference

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

conference reports

This issue's reports are on the USENIX 2001 Annual Technical Conference.

Our thanks to the Summarizers:
:

Josh Simon, Coordinator extraordinary
Brandon Ching
Peter da Silva
William R. Dieter
Kartik Gopalan
Chris Hayner
Adam Hupp
Rosemarie Nickles
Joseph Spadavecchia
Jon Stoffel
Kenneth G. Yocum

USENIX 2001 Annual Technical Conference

Boston, Massachusetts

June25–30, 2001

Introductory Remarks and Keynote Address

Summarized by Josh Simon

INTRODUCTORY REMARKS

The conference began with Dan Geer, the president of the USENIX Association, thanking Clem Cole and Yoonho Park for their work in putting the conference together.

Following the usual general announcements, the Best Paper awards were presented:

- General Track – Best Papers were awarded to "A Toolkit for User Level File Systems," by David Mazieres, and "Virtualizing I/O Devices on VMware . . .," by Jeremy Sugerman et al.
- Freenix Track – Best Paper went to "Nickle," by Bart Massey and Keith Packard; Best Student Paper went to "MEF: Malicious Email Filter," by Matthew Schultz et al.

Following this, USENIX Vice President Andrew Hume presented the USENIX Lifetime Achievement Award (also known as the "Flame") to the GNU Project. Andrew then presented the Software Tools User Group (STUG) Award to the Kerberos development team for its secure, scalable, and relatively simple-to-administer suite of tools. Ted T'so accepted on behalf of the team and donated the $1,000 cash award to USENIX to be used for student stipends for August's USENIX Security Symposium. (See *http://www.usenix.org/directory/awards.html* and *http://www.usenix.org/directory/stug.html* for details.)

KEYNOTE ADDRESS

Dan Frye, director of IBM's Linux Technology Center, spoke about Linux as a disruptive technology. The term isn't intended to have any derogatory connotations; rather, the talk focused on how the growth of Linux has disrupted the status quo of how businesses choose IT products. This year alone IBM is pouring $1 billion into Linux development, working within the public development community, because of business decisions (Linux makes money for the company and for the shareholders) instead of purely technical ones.

A disruptive technology is one where the skills, the desire, and an open culture of significant size all meet. The desire for Linux and the openness of the community are well documented. Further, over time, the skills in computing have moved from mainly academia (meaning colleges and universities) to all levels of education, as well as to hobbyists and even industry, thanks in part to the explosion of games, the Web, the growth in technology, and so on. The increasing commoditization of technology has also fueled the explosion of skills.

IBM believes that Linux as a technology is sustainable in the long term. It's got growing marketplace acceptance, it doesn't lock the customer into a particular vendor for hardware or software, is industry-wide, runs on multiple platforms, and is a basis of innovation. Linux has become critically important for e-business due to the confluence of desire, skills, and the open culture with an ever-growing community size.

Dr. Frye went on to dispel some rumors about Linux in the enterprise environment:

Myth: Open source is undisciplined. Fact: The community is very disciplined, reviewing code and assignments and making sure things are "right" before rolling them into a major distribution.

Myth: Open source is less secure.
Fact: Because of public review and comment to prevent security holes from getting released (or from staying in released code unpatched for long), open source is as or more secure.

Myth: The community doesn't do enterprise features.
Fact: The community wants good designs, but it is not against enterprise features. Designing good, scalable solutions – whether for multiple processors (threading code) or different architectures or clusters, or backing up over high-speed devices (networks) – is a major goal of the community.

Myth: The open source community will fragment.
Fact: Although such fragmentation is possible, IBM believes it is unlikely.

Myth: Traditional vendors cannot participate.
Fact: Untrue; IBM is competing quite well, and other vendors such as Compaq and Dell are making open source OSes available on their hardware platforms as an option for customers.

Myth: Open source doesn't scale.
Fact: Definitely untrue. Open source works on the enterprise scale and in clustering environments.

Myth: Open source has no applications for it.
Fact: Open source has over 2,300 business applications, not counting the many non-business applications that run under the various versions of Linux and *BSD.

Myth: Open source is only a niche market.
Fact: Open source OSes are on nearly a quarter of the servers in production.

Myth: Open source is never used in mission-critical applications.
Fact: While this may have been true in the past, it's becoming less and less so. It should be mission-critical-capable in the very near term.

The IBM Linux Technical Center's mission is to help make Linux better, working within the community. Their URL is *http://oss.software.ibm.com/developerworks/opensourcelinux*.

FREENIX Track

MAC SECURITY

Summarized by Adam Hupp

LOMAC: MAC You Can Live With

Timothy Fraser, NAI Labs

Despite proven usefulness, MAC security models have not been widely accepted. Their primary obstacle has been a high cost of use caused by incompatibility with existing applications and users. LOMAC attempts to solve these problems by implementing MAC security that is transparent to most users, does not require site specific configuration, and is compatible with existing applications. LOMAC uses the Low Water Mark model of protection, which applies well to UNIX systems. It partitions both files and processes into high and low levels. The high level contains critical portions of the system such as init, libraries, and configuration files. The low level is made up of all other system processes and files. Once a high-level process accesses a low-level file it will be demoted. Additionally, low-level processes are unable to signal high-level processes or modify high-level files. This prevents a potentially compromised process from affecting other areas of the system.

LOMAC uses a simple static map to determine the level of files. For instance, all files under /home will be low level, while /usr/sbin would be at a high level. In some cases a program (such as syslogd) must access untrusted resources and still modify high-level files. In these cases the system allows exceptions in order to maintain compatibility. In benchmarks LOMAC had a small performance penalty of between 0–15%. To give a good example of LOMAC's transparency, an experienced user had it installed for 11 days without realizing that it was there.

More information is available at *ftp://ftp.tislabs.com/pub/lomac*.

TrustedBSD: Adding Trusted Operating System Features to FreeBSD

Robert N. M. Watson, FreeBSD Project, NAI Labs

Implementing trusted operating system features can significantly enhance the security of a system. The TrustedBSD Project aims to integrate several new features into FreeBSD that improve security and ease future development work. The most visible new features are MACs, ACLs, and fine-grained privileges. Some of the features such as ACLs and MACs are scheduled to be released in the upcoming FreeBSD 5 kernel. Equally important are auditing, cleaner abstractions, and documentation. The TrustedBSD team found that security checks were often implemented differently in different areas of the kernel, which can lead to bugs.

Watson discussed some of the lessons they had learned in the development process. They found that it was much more effective to work closely with the main developers as opposed to just throwing code over the fence. Another decision that worked well for them was to use existing standards when appropriate. For example, by implementing POSIX.1e ACLs, the Samba server was able to use them with only minor modification. In the future they would like to increase performance and improve the Extended Attribute implementation.

More information can be found at *http://www.trustedbsd.org*.

Integrating Flexible Support for Security Policies into the Linux Operating System

Stephen Smalley, NAI Labs

MACs are able to solve many of the security limitations in current systems but have not yet become widely used. Part of this can be attributed to a lack of flexibility in current systems. Working with Secure Computing Corporation, the NSA developed a flexible architecture called Flask. The security logic in Flask is cleanly separated from the enforcement mechanisms, so users can develop policies based on their particular requirements. The NSA contracted with NAI Labs to create a sample security policy packaged with the software. This sample implementation combines type enforcement, role-based access control (RBAC), and multi-level security (MLS). In SELinux, every operation can have a unique security policy. This allows extremely fine-grained access controls tailored for different uses. Each time an operation is performed the access is revalidated, which means that policy changes take effect immediately.

In benchmarks SELinux showed large performance penalties on some operations, but overall the effect was negligible. Kernel compilation showed a 4% increase in system time and no significant increase in wall time. The benchmarks we done using the very extensive default policy.

More information can be found at *http://www.nsa.gov/selinux*.

SCRIPTING

Summarized by Brandon Ching

A Practical Scripting Environment for Mobile Devices

Brian Ward, Department of Computer Science, University of Chicago

Ward began his presentation by labeling some of the major problems associated with programming for handheld applications. The amount of computational activity and resources available in handheld devices is severely limited because of their size. And their small screen size also presents the problem of proper graphical display. Mobile devices usually do not do any real computation; rather, they primarily display information, which makes proper graphical representation so important.

In lieu of directly tackling such dilemmas, Ward has come up with a parser/compiler similar to PHP, which is called HHL and a virtual machine interpreter called VL. Economizing on space and eliminating redundancy optimizes HHL. It is coded in ANSI Standard C and works like any UNIX compiler.

With these tools, Ward hopes scripting for mobile handheld devices will become more efficient and productive.

Nickle: Language Principles and Pragmatics

Bart Massey, Computer Science Department, Portland State University; Keith Packard, SuSE Inc.

Bart Massey unveiled his and Keith Packard's new C-like programming language called Nickle, revealing the purpose and features of this numerical applications program.

The three main purposes for the Nickle language are calculation, experimentation (algorithms), and prototyping. Massey said that new programming languages should exhibit four basic characteristics. They should serve a useful purpose, serve people other than their creators, be the best language for the given task, and draw on the best practices. According to Massey, Nickle does all of these. With features such as interactive byte code, "C" style programming, powerful numeric types, useful language extensions, and user level threads, Nickle can serve a variety of uses.

USER SPACE

Summarized by Rosemarie Nickles

User-Level Checkpointing for LinuxThreads Programs

William R. Dieter and James E. Lumpp, Jr., University of Kentucky

Dieter introduced the first system to provide checkpointing support for multi-threaded programs that use LinuxThreads, the POSIX-based threads library for Linux. Checkpointing saves the state of a process so that in the event of a system hardware or software failure all would not be lost.

Implementation of the multi-threaded checkpointing library:

- To take a checkpoint all threads are blocked except the main thread. This thread saves the process state and unblocks all the remaining threads.
- To recover from a checkpoint, the checkpointing library restarts the threads which were running when the checkpoint was taken. These threads are blocked until the main thread has loaded the process state from the checkpoint, at which time the threads continue to run from the checkpoint.

The checkpoint library adds little overhead except when taking a checkpoint. This overhead is in proportion to the size of the address space. It is also easy to use. A C programmer needs only to add two lines of code. Source code is available from *http://www.dcs.uky.edu/~chkpt* and *http://mtckpt.sourceforge.net.*

Building an Open Source Solaris-Compatible Threads Library

John Wood, Compaq Computer (UK) Ltd.

This presentation compared Solaris threads to POSIX threads. John Wood discussed the unique Solaris functionality and how to implement:

- Daemon threads
- Join any thread
- Thread suspend and continue

Solving the problem by building an open sourced Solaris compatible threads library would:

- Enable applications that use the Solaris threads API to be ported
- Be an alternative to reworking applications to use POSIX threads
- Not solve generic porting issues

Building an open source Solaris-compatible threads library would eliminate the expense of rewriting the generally non-portable applications that use the Solaris threads application-programming interface.

Questions? Write to *Solaris.complib@compaq.com* or visit *http://www.opensource.compaq.com* or *http://www.tru64unix.compaq.com/complibs/documentation/html/TITLE.HTM*.

ARE MALLOCS FREE OF FRAGMENTATION?

Aniruddha Bohra, Rutgers University; Eran Gabber, Lucent Technologies, Bell Labs

During a comparison study the conclusion was made that mallocs are not free of fragmentation. Nine mallocs were tested with both Hummingbird and GNU Emacs, and the fragmentation varied but none was fragmentation free. PHK/BSD malloc version 42 took first place with a 30.5% fragmentation rate during the Hummingbird test. Doug Lea's malloc version 2.6.6 took the top honors in the GNU Emacs test with a fragmentation rate of 2.69%. PHK/BSD fell to a close fifth with a fragmentation rate of 3.65% in the GNU Emacs test. The worst malloc remained consistent in both tests. Sun OS version 5.8 caused a fragmentation rate of 101.48% in GNU Emacs and failed to finish in the Hummingbird test after causing a heap overflow. Developers should be aware that mallocs are not created equal and should pick one that works well for their workload.

This presentation ended with a plea for further research to understand why certain malloc implementations cause excessive fragmentation.

The Hummingbird and Emacs memory activity traces, the source for the driver program, and the modified bin buddy allocator are available at *http://www.bell-labs.com/~eran/malloc/*.

USER ENVIRONMENT

Summarized by William R. Dieter

SANDBOXING APPLICATIONS

Vassilis Prevelakis, University of Pennsylvania; Diomidis Spinellis, Athens University

Sandboxing helps improve security when running large applications on untrusted data by limiting what resources the program can access. Determining which kinds of access should and should not be allowed is difficult, however. The File Monitoring and Access Control (FMAC) tool helps build sandboxes for applications. To build a sandbox, the user runs the application under FMAC in passive mode on some known safe input. FMAC records the files requested and passes them through to the system. FMAC constructs an access control list (ACL) based on the recorded file requests and uses it to generate a sandbox specification.

When FMAC is run with the sandbox specification it uses chroot to limit access to a particular directory then mounts a special NFS file system on that directory. The special NFS file server allows programs run in the sandbox to access files based on the sandbox specification. Users can view the automatically generated ACL files to see which resources a program uses or modify them by hand to generalize or further limit what resources the program can access. The FMAC tool is designed to balance risk with cost. It is portable, easy to configure, and provides an "adequate level of security" for many users.

One audience member was interested in "session" sandboxes for applications that share multiple files. Vassilis Prevelakis replied that session sandboxes are unnecessary because the sandbox mechanism just filters the view of the file system. There is no need to make multiple copies of files. Other questions related to how well the learning phase covers what the application will try to use once it is in the sandbox. Vassilis said that can be a problem. For example, in one case a user did not access the Netscape help files during the learning session. They were blocked when Netscape ran in the sandbox. Dynamically asking the user if an action should be allowed is generally not safe, because users are confronted with so many pop-up windows they often automatically click "OK."

BUILDING A SECURE WEB BROWSER

Sotiris Ioannidis, University of Pennsylvania; Steven M. Bellovin, AT&T Labs – Research

Due to the explosion in the exchange of information, many modern security threats are data driven. Mail viruses and macro viruses hide inside documents that users want to see and then execute with the same permissions as the users. Sotiris Ioannidis described how SubOS, which is based on OpenBSD, provides a finer-grained level of control than the standard UNIX permission model. When an object, typically a file, arrives at the system, SubOS assigns it a sub-user ID, which corresponds to an access control list. Any time the object is copied, the copies inherit the sub-user ID. Any program that touches the file is limited to the permissions allowed by the sub-user ID.

Sotiris also described a secure Web browser built on SubOS. The browser assigns sub-user IDs to all the objects it downloads. Objects that have valid certificates from trusted sources are given more permissions than untrusted objects. If an object contains code, like JavaScript or Java, the object is run in a separate process with the permissions allowed by its sub-user ID. The damage downloaded scripts can do is limited by their sub-user ID.

When asked what happens if a process accesses two files with different sub-user IDs or communicates through a pipe, Sotiris responded that the process would get the intersection of the two processes' permissions. He also said that, although pipes are currently not covered, all transfer of data should be authenticated. It was pointed out that the application receiving the data from the network needs to help assign permissions to incoming information. Sotiris replied that although the application needs to help with the assignment, once the assignment is made the operating system takes over.

Citrus Project: True Multilingual Support for BSD Operating Systems

Jun-ichiro Hagino, Internet Initiative Japan Inc.

Jun-ichiro Hagino explained how Citrus adds new libraries to NetBSD, and soon OpenBSD, to help applications support multilingual character sets. Character set encodings have evolved from the original 7-bit ASCII to 8-bit encodings that handle most European languages, and to multibyte character sets for larger character sets. External character sets are used outside the program when characters are stored in files. Internal character sets represent characters in memory. Both internal and external character sets continue to evolve, and it is difficult to predict what future character set encodings will look like.

To improve compatibility with existing and future character sets Citrus does not impose a particular internal or external character set encoding. Instead, it dynamically loads a module at runtime to handle a particular user's locale settings. Not imposing a character set helps avoid losing information. For example, multilingual support libraries that use Unicode internally can lose information because some Asian characters that represent different words in different Asian languages map to the same Unicode code points. Citrus avoids this problem by only converting between formats when explicitly requested.

One audience member asked how to separate Citrus from NetBSD to port it to other platforms. Jun-ichiro Itojun said the CVS tree is available and a Citrus Web page will be updated with more information. Another audience member asked how application integration is progressing. Under X11 with KDE and GNOME the window managers can handle it. The older UNIX utilities like vi still do not have multilingual support.

KERNEL

Summarized by Kenneth G. Yocum

Kqueue: A Generic and Scalable Event Notification Facility

Jonathan Lemon, FreeBSD Project

Applications typically receive notifications of events, such as I/O completions, through a select call or by polling. It has been shown that with thousands of event sources, e.g., a Web server with thousands of connections, selective calling/polling does not scale. Kqueue provides a new API for event notification. Kqueue also allows the application to specify filters, so that atypical events can be delivered to the application, including AIO and signals. It was designed to be cheap and fast. Though the setup cost is higher than for polling, it is cheaper when dealing with many possible events. Kqueue filters can also be used to deliver device events or periodic timers.

Improving the FreeBSD SMP Implementation

Greg Lehey, IBM LTC Ozlabs

SMP support in FreeBSD has been woefully inadequate. Typically, only one process could be in the kernel at a time, and blocked interrupts across all processors. Essentially SMP support was provided by one Big Lock. This paper describes their work in applying fine-grain locking for better SMP performance. One problem: interrupt handlers can't block, because they don't have a process context. So give them one, and call it the interrupt thread. Current work is underway to migrate current interrupt handlers to use mutex's in place of calls to spl<whatever>. Though too early for performance numbers, expect the system to scale beyond 32 processors.

Page Replacement in Linux 2.4 Memory Management

Rik van Riel, Conectiva Inc.

The Linux 2.2 virtual memory (VM) subsystem has interesting performance limitations in some circumstances. For instance, pages will be reclaimed from the file cache but not from a day-old idle process. Because page-age information is only accumulated during low-memory situations, a spike in VM activity can cause the system to throw out recently accessed pages. With 2.4 they want to support fine-grain SMP and machines with more than 1GB of memory. They unified the buffer cache, and reintroduced page aging. In general the results have been well received. Performance and stability seem to have improved, though no figures are reported.

STORAGE

Summarized by Adam Hupp

User-Level Extensibility in the Mona File System

Paul W. Schermerhorn, Robert J. Minerick, Peter Rijks, and Vincent W. Freeh, Department of Computer Science and Engineering, University of Notre Dame

The Modify-on-Access (Mona) file system is a new model for manipulating data streams. Transformations are defined on the input and output streams of a file. This allows the system to transparently modify file streams during reads and writes. They are implemented as user-mode shared libraries, as well as at the kernel level. For example, a PHP transformation can automatically parse a PHP template and output the resulting HTML. An FTP transformation could allow users to manipulate remote files as easily as local ones. There are significant advantages for programmers as well. Since common operations can be transparently layered upon each other, developers will be able to use complex functionality through normal I/O mechanisms. It is unnecessary to learn new APIs to use existing components, and writing new components is very simple.

Mona had little overhead when compared to a standard ext2 file system. When tested with complex operations Mona quickly begins to outperform UNIX pipes at equivalent tasks. This speedup (up to 65%) is due to stacked transformations sharing address space and eliminating task switch and buffer copying overheads.

More information is available at *http://www.cse.nd.edu/~ssr/projects/mona.*

Volume Managers in Linux

David Teigland, Heinz Mauelshagen, Sistina Software, Inc.

Volume managers allow disks to be organized logically instead of as fixed sizes. They are becoming critical in expanding Linux systems to the enterprise. This paper gives an overview of volume management software in Linux and some of the developments that are currently being worked on.

The LVM (Logical Volume Manager) and MD (Multi-Disk) driver are the primary volume managers used in Linux. They allow RAID and logical administration of disks, which increases performance and reliability. The latter is useful in both small and large systems. When there is limited disk space, such as on a laptop, the LVM can be used to reallocate partitions that are wasting space. In large systems, the ability to replace disks without bringing down the system is extremely useful.

There are new features being developed for the volume management systems under Linux. The ability to take snapshots of the file system at a point in time was recently added to the LVM. This enables backups to be taken without the risk that a file will be written to. It does not solve the problem of applications leaving data in an inconsistent state, but is a good step in easing backup difficulties. Another new feature currently being implemented is metadata export. This tells the system about the underlying disks and can be used to intelligently place data for optimal performance. In clusters, work is being done on sharing volume management across all nodes. This allows the volumes to be modified on any node, and changes will be consistently propagated across all other nodes.

For more information, see *http://www.sinista.com.*

The Design and Implementation of a Transparent Cryptographic File System for UNIX

Giuseppe Cattaneo, Luigi Catuogno, Aniello Del Sorbo, and Pino Persiano, Dipartimento di Informatica ed Appl., Università di Salerno

Current implementations of distributed file systems lack good protections against eavesdropping and spoofing. The Transparent Cryptographic File System (TCFS) has been developed to provide strong security while remaining simple to use. Files are stored in encrypted form so the remote server will not have access to their contents. Any unauthorized attempt at modification will be immediately noticed. It is implemented in Linux as a layer on top of the VFS, and uses the NFS protocol to communicate with the server.

Key management is often difficult in cryptographic systems, and TCFS has a variety of ways to deal with this. It supports raw keys, basic keys, shared keys, and Kerberized keys. Basic keys use the person's login password as they key. The Kerberized keys allow a user to obtain a ticket from a TCFS key server which then provides access to the files. Shared keys used a secret splitting algorithm in which *n* shares are needed to recreate the original key. Users provide their shares to the kernel, and when enough shares have been received, the file is retrieved. In the future they would like to improve the performance to be closer to standard NFS.

GRAPHICS

Summarized by Rosemarie Nickles

Design and Implementation of the X Rendering Extension

Keith Packard, Xfree86 Core Team, SuSe Inc.

The Xfree86 Core Team picked up the challenge laid down at the 2000 USENIX Technical Conference, where a presentation outlined the state of the X rendering environment and the capabilities necessary to bring X into the modern world. Xfree86 brought forth the X Rendering Extension (Render) with some help with the final architecture from KDE, Qt, Gdk, GNOME and OpenGL. Render replaces the pixel-value-based model of the core X [SG92] rendering system with a RGB model. The examples shown were crisp and clear.

Topics discussed were:

- Anti-Aliasing and Image Compositing
- Rendering Model - Operator
- Premultiplied Alpha
- Basic Compositing in Render
- Text Rendering
- Client-Side Glyphs
- Xft Library
- Polygons, Trapezoids, and Smooth Polygons
- Image Transformation

Scwm: An Extensible Constraint-Enabled Window Manager

Greg J. Badros, InfoSpace, Inc.; Jeffrey Nichols, School of Computer Science, HCI Institute, Carnegie Mellon University; and Alan Borning, University of Washington

Scwm – the Scheme Constraints Window Manager – pronounced "swim," is a complete window manager built for X/11. Scwm's most notable feature is constraint-based layout. Scwm not only embeds the Cassowary Constraint Solving Toolkit for layout constraints but also has a graphical user interface which employs an object-oriented design. Users can create constraint objects or create new constraint classes from existing classes.

Some of the existing constraints were demonstrated and a few of them follow:

- Vertical alignment: This aligns the left edge of one window with the right edge of another
- Constant height/width: If one window were resized the window/windows constrained with it would resize also
- Horizontal/Vertical separation: No matter where a window was moved the window/windows joined in the constraint would always be to the left or above it

Constraints can be enabled or disabled by using the constraint investigation window. Checkboxes are used to enable/disable constraints, and a delete button removes the constraint. There was no question which constraint was being targeted: each time the mouse passed over a constraint in the investigator, the windows related by the constraint were highlighted by a brightly colored line around them.

Scwm can be downloaded from *http://scwm.sourceforge.net*.

The X Resize and Extension – RandR

Jim Gettys, Cambridge Research Laboratory, Compaq Computer Corporation; Keith Packard, Xfree86 Core Team, SuSe Inc.

Have you ever tried to look at your desktop monitor sideways or upside down? The solution is the Resize and Rotate extension (RandR). RandR is designed to allow clients to modify the size, accelerated visuals, and the rotation of an X screen. Laptops and handheld devices need to change their screen size when hooked up to external monitors with different resolutions. RandR allows these changes with simple modifications. A prototype of the RandR is functioning in the TinyX X implementation. This presentation was given using a Compaq iPAQ H3650 Handheld Computer with a HP VGA out PCMCIA card, using Familiar Linux .4, Xfree86 4.1 TinyX Server with RandR extension and MagicPoint Presentation tool.

RESOURCE MANAGEMENT

Summarized by Rosemarie Nickles

Predictable Management of System Resources for Linux

Mansoor Alicherry, Bell Labs; K. Gopinath, Department of Computer Science & Automation, Indian Institute of Science, Bangalore

Alicherry described the Linux scheduler to his audience. He detailed the three scheduling policies: the static priority, the preemptive scheduling, and the "counter" for SCHED_Other. He then turned his attention to resource containers, describing how they are accessed and their parent-to-child hierarchy, as well as the scheduler framework. He explained the source code for changing the CPU share of a container and shell using resource container. The performance overhead chart had the time taken for various operations broken down into microseconds.

For more information: *mansoor@research.bell-labs.com*.

Scalable Linux Scheduling

Stephen Molloy and Peter Honeyman, CITI, University of Michigan

Linux uses a one-to-one thread model, which is implemented easily but potentially overloads the kernel's default scheduler. Servers run multi-thread applications. Experiments conducted by IBM showed that a scheduler for heavily threaded workloads could dominate system time. The Linux scheduler design assumes a small number of ready tasks, such as:

- Simple design means easy implementation
- On runtime
- It performs many of the same calculations on each invocation

The goals are:

- To make the scheduler fast for both large numbers of tasks (server envi-

- ronment) and small numbers of tasks (desktop environment)
- To provide an incremental change to keep current criteria, maintain current interfaces, and preserve goodness metric
- To understand reasons for any performance differences

The solution is to use a sorted table instead of an unsorted queue and only to examine tasks in the highest populated list, falling through to the next list if strictly necessary.

Any questions?

Steve Molloy: *smolloy@umich.edu*
Peter Honeyman: *honey@citi.umich.edu*
CITI: *http://www.citi.umich.edu*
IBM Linux Tech Center:
http://www.linux.ibm.com

A Universal Dynamic Trace for Linux and Other Operating Systems

Richard Moore, IBM, Linux Technology Center

A universal dynamic trace can operate in kernel or user space. It will work under the most extreme conditions at interrupt time or task time or even between contests, and it operates in an MP environment. It also can be used to trace code or data usage. It is dynamic because of the ability to insert tracepoints at runtime without the need to modify or prepare traced code in advance. Actions taken when the tracepoint fires are customizable at runtime. Thus there's a need for a debugging engine to be interfacing with a standard system tracing mechanism. Dynamic Probes provides the debugging engine.

For more information:

Mailing list:
dprobes@oss.software.ibm.com

Web page:

http://oos.software.ibm.com/developerworks/opensource/linux/projects/dprobes

General Track

OPERATING SYSTEMS

Summarized by Kartik Gopalan

Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor

Jeremy Sugerman, Ganesh Venkitachalam, Beng-Hong Lim, VMware Inc.

This paper won the Best Paper Award in the General Track. In a very well presented talk, Jeremy Sugerman described VMware Workstation's approach to virtualizing I/O devices and explained various optimizations that can improve the throughput of the virtualized network interface.

The talk began with a recounting of the concept of virtual machines pioneered by IBM in its mainframe machines. The concept of virtual machines is still beneficial in the context of modern-day desktop PCs due to users' need to run applications from multiple operating systems simultaneously. It can also be useful for server consolidation by enterprises and services providers in order to better utilize resources and ease system manageability.

Jeremy described the VM architecture and the mechanism used for virtualizing the network interface. The virtual NIC appears as a PCI-Ethernet controller to the guest OS and can either be bridged to a physical network connected to the host or connected to a virtual network created within the host. All packets sent out by guest OS are directed to the VMApp by a VMDriver, which transmits the packet out on the physical network through a VMNet driver.

The TCP throughput provided by such an approach on a 733MHz Pentium machine was only 60Mbps as compared to native throughput of 90Mbps. The main reason for lower throughput by VM was due to I/O space accesses requiring world switch to VMApp and the time spent processing within the VMApp. The key strategy to improve performance is to reduce the number of world switches. With optimizations such as making VMM directly handle I/O, not requiring host hardware, clustering packets before sending, and using shared memory between VMNet driver and VMApp, the TCP connection is able to saturate the network link.

This work essentially showed that VMware Workstation's achievable I/O performance strikes a good balance between performance and modern-day desktop compatibility. For more information, visit *http://www.vmware.com*.

Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources

Jeff Bonwick, Sun Microsystems; Jonathan Adams, California Institute of Technology

Jeff Bonwick first reviewed the current state of slab allocation mechanisms. Slab allocators perform object caching to reuse states of previously created objects. However, there are two disadvantages: global locking doesn't scale to many CPUs and allocators cannot manage resource other than kernel memory.

To address scalability, Jeff proposed the "magazine layer" which consists of magazines and depots. Magazines are basically per-CPU caches whereas depots keep a global stockpile of magazines. Each CPU's allocations can be satisfied by its magazine until the magazine becomes empty, at which point a new magazine is reloaded from the depot. The performance measurements indicated almost perfect scaling properties of magazine layer with increasing number of CPUs.

Virtual address allocation is just one example of a more general resource allocation problem, where resource is anything that can be described by a set of integers. Jeff proposed a new general-purpose resource allocator called

"Vmem," which provides guaranteed constant-time performance with low fragmentation and linear scalability. Vmem eliminates the need for special purpose allocators, such as a process ID allocator, in the operating system. The implementation details and performance results were presented. Vmem was shown to provide constant-time performance even with increasing fragmentation.

Following this, Jonathan Adams presented a user-level memory allocation library called "libumem." Libumem is a user-level port of the kernel implementation of magazine, slab, and Vmem technologies. Some of the porting issues reported dealt with replacing CPU ID with thread ID, handling memory pressure, supporting malloc(3C) and free(3C), and needing lazy creation of standard caches. Libumem was shown to give superior malloc/free throughput performance compared to hoard, fixed, and original mtmalloc, ptmalloc, and libc.

Magazines and Vmem are part of Solaris 8. The sources are available for free download at *http://www.sun.com*.

Measuring Thin-Client Performance Using Slow-Motion Benchmarking

S. Jae Yang, Jason Nieh, and Naomi Novik, Columbia University

In this talk, Naomi Novik presented the technique of slow-motion benchmarking for measuring the performance of thin-client machines. First, she introduced the concept of thin clients. Thin clients are designed to provide the same graphical interfaces and applications available on traditional desktop computers while centralizing computing work on powerful servers. All application logic is executed on the server, not on the client. The user interacts with a lightweight client that is generally responsible only for handling user input and output, such as receiving screen display updates and sending user input back to the server over a network connection.

The growing popularity of thin-client systems makes it important to develop techniques for analyzing their performance. Standard benchmarks for desktop system performance cannot be used to benchmark thin-client performance since applications running in a thin-client system are executed on the server. Hence these benchmarks effectively only measure the server's performance and do not accurately represent the user's experience at the client-side of the system. To address this problem, Naomi presented slow-motion benchmarking, a new measurement technique for evaluating thin-client systems.

The performance of a thin-client system should be judged by what the user experiences on the client. Direct instrumentation of thin clients is difficult since many thin-client systems are quite complex. In slow-motion benchmarking, performance is measured by capturing network packet traces between a thin client and its respective server during the execution of a slow-motion version of a standard application benchmark. Slow-motion execution involves altering the benchmark application at the server end. This is done by introducing delays between the separate visual components of that benchmark so that the display update for each component is fully completed on the client before the server begins processing the next one. These results can then be used either independently or in conjunction with standard benchmark results to yield an accurate and objective measure of user-perceived performance for applications running over thin-client systems.

Naomi concluded the talk by presenting slow-motion benchmarking measurements of Web server and video playback benchmarks on client/server systems that included a Sun Ray thin client machine and a Sun server.

STORAGE I

Summarized by Joseph Spadavecchia

The Multi-Queue Replacement Algorithm for Second-Level Buffer Caches

Yuanyuan Zhou and James Philbin, NEC Research Institute; Kai Li, Princeton University

Yuanyuan Zhou presented the Multi-Queue Replacement Algorithm (MQ) for second-level buffer caches. Almost all second-level buffer caches use locality-based caching algorithms, such as LRU. These algorithms do not perform well as second level buffer caches, because they have different access patterns than first-level buffer caches – accesses in the second level are missing from the first. Almost all of today's distributed multi-tier computing environments depend on servers that usually improve their performance by using a large buffer to cache data. There is a strong need for a better second-level buffer cache replacement algorithm.

The authors' research shows that a good second level buffer cache replacement algorithm should have the following three properties: minimal lifetime, frequency-based priority, and temporal frequency.

The minimal lifetime constraint means that warm blocks stay in the buffer cache at least a certain amount of time for a given workload. Frequency-based priority, as its name suggests, assigns blocks priority based on their access frequency. Temporal frequency is used to remove blocks that are not warm.

The MQ algorithm satisfies the three properties listed above and has O(1) time complexity. MQ uses multiple LRU queues to maintain blocks with different access frequencies for different periods of time in the second-level buffer cache.

MQ is also simpler to implement than FBR, LRFU, and LRU-K.

The authors did trace-driven simulations to show that MQ outperforms LRU, MRU, LFU, FBR, LRU-2, LRFU and 2Q as a second-level buffer cache replacement algorithm, and that it is effective for different workloads and cache sizes. In some cases MQ yields a 53% improvement over LRU and a 10% higher hit ratio than FBR.

The proof is in the implementation. The authors tested the performance by implementing MQ and LRU on a storage server with Oracle 8i Enterprise Server as the client. The results obtained using TPC-C benchmark on a 100GB database show that MQ improves the transaction rate by 8–11% over LRU. For LRU to achieve the same performance as MQ requires that the server's cache size be doubled.

Design and Implementation of a Predictive File Prefetching Algorithm

Thomas M. Kroeger, Nokia Clustered IP Solutions; Darrell D. E. Long, University of California, Santa Cruz

Kroeger discussed the design and implementation of a predictive file prefetching algorithm. Research has shown that the patterns in which files are accessed can predict upcoming file accesses. Almost all modern caches do not take file access patterns into account. Heuristics that expect sequential accesses cannot be applied to files, because the concept of a file does not have a successor. Hence, modern caches fail to make use of valuable information that can be used to reduce I/O latency.

Previously the authors developed a compression-modeling technique called Partitioned Context Modeling (PCM) that monitors file access to predict upcoming requests. PCM works in a linear state space through compression, but experimentation showed that it does not predict far enough into the future. Therefore, the authors developed Extended Partition Context Modeling (EPCM) that predicts much farther into the future.

The authors implemented predictive prefetching systems (PCM and EPCM) in Linux and tested them with the following four application-based benchmarks:

1. Andrew Benchmark
2. GNU ld of the Linux kernel
3. Glimpse index of /usr/doc
4. Building SSH

The Andrew Benchmark is a small build. Though dated, it is widely used and accurately portrays the predictive relationship between files. The GNU ld of the Linux kernel was used to represent a workload of non-sequential file accesses. The Glimpse indexing of /usr/doc generated a workload representing a traversal of all files under a given directory. Finally, the building of SSH 1.2.18 through 1.2.31 was used to represent the compile edit cycle. The system was able to train on the initial version (1.2.18) and then used that training on sequentially-modified versions (1.2.19 – 1.2.31).

The results of testing show that I/O latency reduced by 31–90% and elapsed time reduced by 11–16%. With EPCM, the Andrew Benchmark, GNU ld, Glimpse, and SSH saw elapsed time improvements of 12%, 11%, 16%, and 11%, respectively. I/O latency was improved as much as 90%, 34%, 31%, and 84%, respectively.

Question: Were any tests done on a multi-user system where accesses are not associated with the tasks of a single user? It seems that it would be harder to predict file access with multiple users. Answer: Tests like these have not been done yet.

Extending Heterogeneity to RAID Level 5

T. Cortes and J. Laborta, Universitat Politécnica de Catalunya

Cortes described work on extending heterogeneity to RAID level 5 (RAID5), which is one of the most widely used types of disk arrays. Unfortunately, there are some limitations on the usage of RAID5. All disks in a RAID5 array must be homogeneous. In many environments, especially low-cost ones, it is unrealistic to assume that all disks available are identical. Furthermore, over time disks are upgraded and replaced resulting in a heterogeneous unit. According to studies by IBM, disk capacity nearly doubles while prices per MB decrease by 40% every year. Consequently, it is neither convenient nor efficient to maintain a homogeneous RAID5 disk array.

There are some projects that have already focused on solving this problem; however, they deal only with multimedia systems. The solution the authors described is intended for general purpose and scientific settings, though it also works well for multimedia applications.

The authors presented a block distribution algorithm called AdaptRaid5 that they used to build disk arrays from a set of heterogeneous disks. Surprisingly, in addition to providing heterogeneity, AdaptRaid5 is capable of servicing many more disk requests per second than RAID5. This is because RAID5 assumes that all disks have the lowest common speed, whereas AdaptRaid5 does not.

Experimental results were shown comparing the performance of traditional RAID5, RAID5 using only fast disks (OnlyFast), and AdaptRaid5. AdaptRaid5 significantly outperformed RAID5 and OnlyFast for capacity evaluation, full-write, and small-write performance measures. However, with more than six disks OnlyFast performed better than AdaptRaid5 for read, and real-workload performance measures. This is because AdaptRaid5 must account for slow disks, whereas OnlyFast cannot.

The authors measured real-workload performance by using a trace file supplied by HP. Performance gains obtained using AdaptRaid5 versus RAID5 over five disks were almost 30% for reads, and 35% for writes. AdaptRaid5 versus OnlyFast performance gains ranged from nearly 30% for reads to 39% for writes. OnlyFast had an approximate 3% read performance gain over AdaptRaid5 when eight fast disks were used, but this is because slow disks are never used in OnlyFast.

TOOLS

Summarized by Peter da Silva

Reverse-Engineering Instruction Encodings

Wilson Hsieh and Godmar Beck, University of Utah; Dawson R. Engler, Stanford University

Hsieh presented this paper. The problem the authors were trying to solve was how to efficiently produce code generators for just-in-time (JIT) compilers like Kaffe. The JIT compiler has to produce efficient instruction sequences quickly and reliably; creating the tables for the code generator from an instruction sheet is complex and error-prone.

Most systems, however, already have a program that knows about the instruction set of the computer, the assembler. By generating instruction sequences and passing them through the assembler, their system, DERIVE, can produce tables that describe the instruction set and can be used to drive code generators.

Wilson described how DERIVE takes a description of the assembly language and repeatedly generates instruction sequences that step-by-step probe the underlying instruction set to derive register fields, opcode fields, and labels. There are three phases: the register solver, immediate solver, and jump solver.

The register solver tests each argument at a time, sequencing through all possible registers. In a RISC CPU this is simple, and a single pass through the assembler can provide all possible bitmaps for analysis. But for a complex instruction set like the Intel x86 many combinations of registers have unique encodings or are even illegal, so the assembler has to be called over and over again for each combination.

The immediate solver and jump solver work similarly, calling the register solver to extract the encodings. The immediate solver (which also handles absolute jumps) performs a linear search through the possible arguments to find the maximum size, then solves each possible argument size separately. The jump solver does a similar job, except it has to generate appropriate labels and adjust for scaling.

Solving an instruction set can take between 2.5 minutes and four hours (for the Intel x86 architecture), depending on the complexity of the encoding.

The generated tables are a set of C-like structures that are passed to a code generator, which produces C macros to generate the final code. These tables are surprisingly efficient: they were able to reduce the size of the Kaffe code generator for the x86 architecture by 40% with one day's work.

Source code is available from *http://www.cs.utah.edu/~wilson/derive.tar.gz.*

An Embedded Error Recovery and Debugging Mechanism for Scripting Language Extensions

David M. Beazley, University of Chicago

What happens if you have an error in C or C++ code called from a scripting language? Well, normally if you have an error in a high-level language, the interpreter gives you a nicely formatted backtrace that shows exactly where your program died. Similarly, if your C code crashes you get a core dump that can be examined by a debugger to produce a nicely formatted backtrace showing you exactly where your program died.

In a scripting language extension, you get a low-level backtrace of the high-level-language stack, which generally consists of layer after indistinguishable layer of the same three or four interpreter routines over and over again. Digging useful information out of this can be challenging.

To solve this problem, WAD (Wrapped Application Debugger) runs as a language extension itself and sets up signal handlers for all the common traps, such as SIGSEGV, SIGBUS, and so on. When an exception occurs, WAD unrolls the stack and generates a formatted dump of the low-level code, using whatever debugging information is available to it in symbol and debugging tables, then simulates an error return to the highest level interpreter stack frame it can find and passes this dump back as the error text.

At this point the scripting language itself can unroll its own stack the rest of the way and pass the combined set of stack traces to the programmer.

No relinking and no separate debugger are necessary.

Beazley proceeded to demonstrate the debugger for both Tcl and Python. For the first, a small wish program opened a

Tk window that provided radio buttons to select exactly what kind of exception to use, using an extension in C that simply produced the requested exception and let WAD and the Tk error handler pop up a window containing the combined stack trace. For the second, he used a Web server running Python extensions, with an error handler that dumped the stack trace to the browser and the Web server's error log.

There are problems to be worked out. Since the debugger doesn't have the same intimate knowledge of the code as the interpreter's error handler, it can leak memory, lose locks, lose open files, and so on. Still, my biggest disappointment is that it's not available for Tru64/Alpha but only for Solaris/Sparc and Linux/x86. For more information, visit *http://systems.cs.uchicago.edu/wad.*

Interactive Simultaneous Editing of Multiple Text Regions

Robert C Miller and Brad A. Myers, Carnegie Mellon University

Miller started out by thanking USENIX for supporting his work.

Then he described the problem he was trying to solve: repetitive text editing is error-prone, even with the assistance of macros, regular expressions, and language-sensitive editors. This paper described a tool that attacks the problem from a different direction, using an interactive program that provides the user with immediate feedback while performing the same editing operation in multiple places in a file.

Lapis is a simultaneous editor; the user selects parts of a region and edits it, and the same operation is performed simultaneously in the same place in all similar regions.

Editing is the easy part. Identifying the fields to be edited is harder. Lapis solves this problem by providing instant feedback and examples. The user identifies a section of the region by selecting it, and then Lapis generalizes the selection and highlights it in all fields. If the program guesses wrong – too much or too little selected in some field – the user can provide more samples to home in on the desired selection.

Splitting the file up into regions can be handled similarly, by selecting a number of examples and entering simultaneous editing mode on these regions. Alternatively a pattern can be selected directly from a nested list in the lower right of the Lapis window, and it will automatically select all the matching regions.

Performance is a problem: the heuristics Lapis uses to recognize patterns and fields are expensive. Lapis solves this by finding all interesting features of each region when the file is split up, then adding new features as the user continues to edit them. Features are never removed from this list; it's faster to skip over false positives.

Robert then described a series of experiments at CMU, where undergraduates were given a group of simple editing jobs and asked to solve them using simultaneous editing and more traditional tools. To stack the deck against himself, he had the students perform the operation using Lapis first, so they were already familiar with the problem when they switched to their traditional tools.

For as few as 3–10 records Lapis was already faster than traditional tools, for users who had never used Lapis before.

A Java implementation is available from *http://www.cs.cmu.edu/~rcm/lapis.*

WEB SERVERS

Summarized by Kenneth G. Yocum

High-Performance Memory-Based Web Servers: Kernel and User-Space Performance

Richard Neves, Philippe Joubert, ReefEdge Inc.; Robert King, John Tracey, IBM Research; Mark Russinovich, Winternals Software

This is a four-year-old IBM effort to improve Web performance. The first kernel-mode Web server was produced in 1998. It has made its way into S/390, AIX, Linux, and Windows. The goals were to identify the performance gap between user mode and kernel mode on multiple production OSes without modifying the kernel (TCP/IP stack, drivers, or hardware). They identified three first order performance issues with user-mode servers: data copies, event notification, and data paths. User-mode approaches include reducing memory copies and performing checksum offloading. This means using mechanisms like Fbufs, IO/Lite, a transmit file primitive, and Kqueue.

IBM introduced AFPA (Adaptive Fast Path Architecture), the kernel-mode engine, in 1997. It is integrated with TCP/IP stack and the file system. Other systems included TUX, early Linux kernel-mode Web servers, the Lava hit-server, and Cheetah in the Exokernel work. In AFPA the HTTP module is separated for portability, and it can still run with user-mode Apache or proxy cache. AFPA supports multiple protocols, not just HTTP. Kernel manages zero-copy cache.

The test platform was 12 two-way 450MHz Xeon clients and a uniprocessor server. It achieved 1.2Gbps performance. In summary, user mode is about 3.5 times slower than kernel mode. Interrupt-based architectures are 12% faster than thread-based ones. Zero-copy doesn't help much with requests less than 4K, but direct integration with the TCP/IP stack can improve performance by 55%.

Kernel Mechanisms for Service Differentiation in Overloaded Web Servers

Thiemo Voigt, Swedish Institute of Computer Science; Renu Tewari, Douglas Freimuth, IBM T.J. Watson Research Center; Ashish Mehra, iScale Networks

The Internet is quickly growing and requiring support for new services that depend on highly available servers. Server overload is a problem, and people won't pay for its solution. Traditional servers provide marginal overload protection, but that's not good enough. To get predictability they provide three mechanisms: TCP SYN policing, prioritized listen queue, and URL-based connection control. As with most systems papers, there are three design principles: don't waste cycles, minimize changes to network subsystem, and be able to implement these mechanisms on servers and on Layer4/7 intermediary switches.

The three mechanisms provide support at increasing levels of consumed resources. TCP SYN policing is part of the network stack. It limits the number of connection requests to the server by using token buckets, which have rate and burst attributes. When those are exceeded the SYN is dropped. The prioritized listen queue allows connections to be organized into pools with different priorities. When a TCP connection is established, the socket is placed into the listen queue according to this priority. URL-based connection control inspects cookies to ID clients to allow content-based connection control.

Because it is difficult to identify specific sets of misbehaving clients, SYN policing, though effective, is in practice difficult to tune correctly. The bucket rates should also be adjusted to the level of resource consumption per request. Simple priority listen queue policies allow lower delay and higher throughput for high-priority connections, but may starve low-priority connections. Combining these two techniques can avoid the starvation problem. In general, kernel-based mechanisms are more efficient than user-level mechanisms. They have the opportunity to toss out the connection before it consumes additional resources.

Storage Management for Web Proxies

Elizabeth Shriver, Eran Gabber, Bell Labs; Lan Huang, SUNY Stony Brook; Christopher A. Stein, Harvard University

Proxies are black boxes with a high-end PC inside, or they're just high-end PCs. In any case, they have particular file system performance characteristics. Files are accessed in their entirety, there is a flat namespace, permission checking is rare, and, since they are caches, proxies exhibit less stringent persistence requirements. The bottom line is that traditional file systems have a lot of unnecessary functionality that is unused and, instead, reduces the performance of proxies. The Hummingbird FS is the response to this observation.

The authors implemented the file system as a library that is linked with the application. The file system and application share the buffer cache. There is no reason to copy data, just pass a pointer. Everything is read/written in clusters, and replacement is LRU. Clusters are files stored together on disk. They are associated via calls that give hints to Hummingbird. A file can be in more than one cluster. Large files are special; they are not cached but are kept on disk. Hummingbird also has parameters that affect the sizes and lifetimes of clusters. The application can specify when and how often to write metadata back to disk.

They implemented it and simulated Squid proxy accesses. File reads were much faster. In most cases throughput improved over five times. Because of clustering, Hummingbird issues fewer disk I/Os than UFS, and recovery is much faster than UFS. It takes 30 seconds for Hummingbird to start servicing requests after a system crash with an 18GB disk. UFS takes 20 minutes in order to fsck. The same line of reasoning applies to Web servers as well, though you'll need more functionality, like ls, than the toolkit currently provides.

SCHEDULING

Summarized by Joseph Spadavecchia

Pragmatic Nonblocking Synchronization for Realtime Systems

Michael Hohmuth and Hermann Härtig, Dresden University of Technology

Hohmuth presented work on pragmatic nonblocking synchronization for realtime systems. Recently there has been a stir about nonblocking data structures. Nonblocking synchronization is useful for realtime systems because it is preemptive everywhere, and there is no priority inversion. It has caught the attention of not only the realtime systems community, but also the operating systems and theoretical groups. In spite of the great interest there are very few known implementations that exploit nonblocking synchronization successfully.

Michael explained that the lack of implementation for nonblocking synchronization is partially hardware related. It is difficult to apply to many modern CPU architectures, because implementation relies on hardware support for atomically updating two independent memory words, such as two-word compare-and-swap (CAS2). For example, the popular x86 CPUs do not support such an instruction.

The authors' work is a pragmatic methodology for creating nonblocking realtime systems that even work on CAS2-less architectures. That is, it does not rely solely on lock-free synchronization. It allows locks, but assures that the system remains wait-free. In addition, the methodology is easy to use because it looks like programming with mutex using monitors.

The authors implemented the Fiasco micro-kernel for the DROPS realtime operating system using their methodology. Fiasco is an implementation of the L4 micro-kernel interface that runs on x86 CPUs. C++ was used to implement the kernel, yet it performs well compared to the original, optimized, non-realtime, assembly language implementation.

The performance evaluation results show that the level of preemptability of the Fiasco micro-kernel is close to that of RTLinux. In fact, the maximal lateness in the Fiasco micro-kernel is an order of magnitude smaller than that for the L4/x86 kernel. This is because L4/x86 disables interrupts throughout the kernel to synchronize access to kernel data structures.

Scalability of Linux Event-Dispatch Mechanisms

Abhishek Chandra, University of Massachusetts, Amherst; David Modberger, HP Labs

Chandra discussed the scalability of Linux event-dispatch mechanisms. Today's Internet servers need to service high incoming-request loads while simultaneously handling a large number of concurrent connections. To handle the workload, servers must employ event-dispatch mechanisms provided by the underlying operating system.

Chandra presented a comparative study of the Linux kernel's event-dispatch mechanisms and their performance measures in terms of dispatch overhead and dispatch throughput. The study showed that POSIX.4 realtime signals (RT signals) are a highly efficient mechanism compared to select() and /dev/poll.

Unfortunately, RT signals have a few drawbacks. They use a signal queue, which can overflow. In such an event, a misbehaving connection can starve other connections from the queue; a different mechanism is needed as a fall-back. This may be computationally costly and make applications overly complex. Another drawback to RT signals is that they cannot de-queue multiple signals from the queue simultaneously.

The authors' work includes a solution (signal-per-fd) to RT signals' shortcomings. Signal-per-fd coalesces multiple events and presents them as a single signal to the application. In doing so it also solves the starvation problem by only adding new signals to the signal queue if there is already a signal queued for that fd. Furthermore, it reduces the complexity of the application, removing a need for a fall-back mechanism. Finally, it allows the kernel to return multiple events simultaneously.

An experimental study was done using 252 to 6000 idle connections and 1 byte to 6 KB reply sizes. Results confirm that both RT signals and signal-per-fd have higher throughput, lower CPU usage, and lower response time with many idle connections than do select() and /dev/poll.

Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler

Jason Nieh, Chris Vaill, and Hua Zhong, Columbia University

Vaill presented the Virtual-Time Round-Robin (VTRR) O(1) proportional share scheduler. Proportional share schedulers are useful for dividing scarce resources among users and applications. In proportional share scheduling, a weight is associated with each process. Resources are then divided among processes in amounts proportional to their associated weights.

Early proportional share mechanisms were efficient, but not accurate. One of the oldest proportional share schedulers is Weighted Round Robin (WRR). WRR is an O(1) proportional share scheduler; unfortunately, it is not accurate. This motivated the development of fair queuing algorithms such as Weighted Fair Queuing (WFQ) that provide better accuracy. Unfortunately, in these algorithms the time for selecting a process for execution is a function of the process count.

VTRR is an O(1) proportional share scheduler that is both accurate and efficient. It works by ordering all tasks in the queue by share size. It then allocates one quantum to each task in order, starting with the task with the largest share. Next, VTRR chooses to reset to the first task if the current task has received more than its proportional allocation.

Simulations have shown that VTRR is much more accurate than the WRR proportional share scheduler. On average WRR's error ranges from -398 tu to 479 tu, whereas VTRR's error only ranges from -3.8 tu to 10.6 tu (1 tu is 10 ms). On the other hand, WFQ happens to be more accurate then VTRR; however, VTRR's inaccuracy is on such a small scale that it is below the delay threshold noticeable by most human beings.

Vaill explained that VTRR is simple to implement. It has been implemented in Linux in less than 100 lines of code. For a large numbers of clients, the overhead using VTRR is two orders of magnitude less than the standard Linux scheduler. It is important to note that the Linux scheduler is optimized for interactive tasks, whereas VTRR is not.

The authors performed tests to measure the scheduling behavior of VTRR, WFQ, and the Linux scheduler at a fine time granularity. VTRR and WFQ do a better job of proportional scheduling than the standard Linux scheduler.

In addition, tests with real application workloads (MPEG encoding and running several VMware machines) were performed. In both cases VTRR performs very close to WFQ, trading a very small amount of precision for much lower scheduling overhead. Conversely, the standard Linux scheduler did the worst job in terms of proportional share scheduling.

Invited Talks

Making the Internet Mobile: Lessons from the Wireless Application Protocol

Sandeep Singhal, ReefEdge Inc.

Summarized by Brandon Ching

Singhal spoke on how the Wireless Application Protocol (WAP) can and will meet the needs of the wireless Internet. With the world becoming busier every day and with less time allowed for stationary net access, the growing need for mobile handheld Internet devices is becoming ever more pressing. Along with the demand must come a standard that defines how these devices communicate and work together. That standard is WAP.

WAP is a set of specifications and protocols that explain how things should and must operate while communicating. WAP is similar to TCP/IP in that both are widely accepted standards of how two devices communicate, but with WAP the devices will most likely be cell phones and PDAs instead of workstations and standard Web servers.

Sharing many features with the traditional Internet, WAP allows mobile users to access realtime information easily and gives the added convince of "do it on the go" interaction. This lets you do things such as make flight, hotel, and rental car reservations while at the same time scheduling your 11:00 meeting. Of course WAP also makes stock trading, commerce, voicemail and instant messaging available to you anywhere and anytime.

In case you have been living under a rock for the past five years and insist on asking why bother with all this, Singhal has the answer for you. In terms of corporate and business interests, customer growth and acquisition drive what seems to be an endless push for services and features. And WAP allows a person to keep in touch with family and friends more easily conduct important business and market decision making, and just plain make life a bit more convenient in the process.

All this new technology on the go sounds really nice, so what are the challenges facing this new WAP technology? Since these mobile handheld devices are becoming smaller every day, proper display of information becomes a big problem. The Internet and its services have been designed around traditional PCs, and complex scripting and inefficient HTTP over TCP/IP connections to handheld devices therefore make the acquisition of data and media clumsy. We are also battling over issues such as limited bandwidth and network latency.

The future of WAP looks bright. It has successfully met many challenges, yet it has also fallen short on many Internet expectations. Perhaps the new WAP 2.0 migration to Internet standards will give the performance lift needed for today's unforgiving world.

Evolution of the Internet Core and Edge: IP Wireless Networking

Jim Bound, Nokia Networks; Charles E. Perkins, Nokia Research Center

Summarized by Kartik Gopalan

In this talk, Bound discussed the evolution of IP wireless and mobile computing. He began by observing that the explosion in the number of IP-capable mobile devices, such as cell phones, has placed tremendous pressure on the Internet core infrastructure and edge architecture. The Internet today is characterized by diverse VPNs that have essentially private address spaces and are secure at their edges by use of firewalls, Network Address Translation (NAT), and application level gateway (ALG) mechanisms. In the process, the end-to-end model of the Internet is getting lost. In addition, getting globally routable IPv4 addresses is becoming more and more difficult. For instance, it is virtually impossible for a company to deploy millions of cell phones, each with a globally routable IPv4 address. A solution to this problem is deployment of IPv6, which can restore the end-to-end Internet model and also solve the address space problem. In addition, it enables large-scale deployment of Mobile IP, which is going to revolutionize the Internet.

Bound next discussed the evolution of wireless protocols including GSM, GPRS, UMTS in Europe, CDMA in United States, and finally Mobile IPv6 itself, which promises 2Mbps voice and data over completely IP-based networks. IPv6 is essentially a packet-switching architecture in contrast to today's telephone networks, which are circuit based. One of the challenges is to make IPV6 work in conjunction with SS7, used in today's telephone networks. For instance, IETF's SIGTRAN addresses the transport of packet-based PSTN signaling over IP networks. One of the promising protocols pointed out was the Streaming Control Transport Protocol

(SCTP), which enables true streaming that is not possible using present-day TCP.

In order to tackle rapid consumption of IPV4 addresses and routing table explosions, CIDR was proposed as an interim measure. While CIDR reduced the pressure on address space, it still required NAT and ALGs, which imposed tremendous management burden and created a single point of failure in the network. This also imposed performance penalties and prevented deployment of end-to-end technologies such as IPSec.

IPv6, which was standardized in 1998, promises a solution to these problems. It touts 128-bit addresses, has a simple IP header, and is optimized for 64-bit architecture. IPv6 gets over the need for NAT and ALGs and, furthermore, has been designed to be Mobile IP ready. The primary “wireless” advantage of IPV6 is its extended address space. Handoff is complex in IPV6, but it can bypass the triangular routing problem faced in IPv4. Security required during binding updates can be provided by IPSec. Key management will be a major issue in this scenario, for which AAA servers will form the basis.

Bound concluded the talk by touching on the problem of new frequency spectrums that will be required to enable the diverse mobile devices to communicate. He also stated that people from the circuit-switching world will ultimately adapt to this unified communications infrastructure based on IPv6. However, a lot of testing and trials will be required before this actually happens. Jim’s prediction was that Asia will be the first to embrace the wireless world since a wired infrastructure is not as enmeshed there as in the United States.

Security Aspects of Napster and Gnutella

Steven M. Bellovin, AT&T Labs – Research

Summarized by Chris Hayner

Bellovin began his talk by describing the many functions common to Napster and Gnutella and, by extension, to every other P2P network. Without central servers controlling the data, the clients are free to decide what to share and what to keep secret. The very protocol supplies the index of peers and connectivity information, allowing direct connection from peer to peer without going through any intermediary.

Napster uses a central server as a base for users to query for files and as a supplier of chat functions. A compiled index keeps track of who has what, at what speed they are connected, etc. By selecting a file of interest, a user gets connection information from the server and then initiates a direct connect to the peer who is sharing the file. Also available is a “hot-list” function, allowing a private list of specific users’ connection status.

The Gnutella protocol is different in that there is no central server whatsoever. All users have their own index, which is updated from the indexes of the users they are connected to. This creates a very large network of users connecting to users connecting to users. It is not uncommon for any single user to have up to 10 connections. The Gnutella protocol is an open specification.

The search strength of Gnutella resides in its flooding protocol, wherein a user has the ability to speak to every connected machine. A user searching for a file sends a request to all of his or her neighbors, who in turn forward it to their neighbors. When there is a match, the user directly connects to the user with the file, and download begins. Aside from basic IP address information, there is no authentication of any type.

The talk focused primarily on Gnutella, and at this point, Bellovin discussed at great length the specifics of the Gnutella protocol’s five messages: ping, pong, push, query, and query hits. An in-depth discussion of these is beyond the scope of this summary.

Gnutella suffers from the openness of its protocol in several obvious ways. First, there is no authentication of the IP, so the network could conceivably be used in a flooding attack. There would be a lot of attempts to connect to, say, CNN.com, if it were put in a packet that cnn.com:80 was sharing 10,000 files. Also, the Gnutella packet headers contain the MAC address of a computer using Win95/98/NT. This could be used to link requests to requesters and is an obvious privacy violation.

Using a central authority to authenticate makes it very difficult to fake an IP address. The privacy issues are much more apparent here, as the central site could conceivably keep track of every single session for every single user.

The conclusion was that although Gnutella is the wave of the future, there are significant privacy concerns. Authentication of some kind would make the Gnutella network more legitimate as well. Clients need to be well-written to avoid buffer overflows, which are all too prevalent in some kludgy Gnutella clients.

For more information, see *http://www.research.att.com/~smb*.

Security for e-Voting in Public Elections

Avi Rubin, AT&T Labs Research

Summarized by Adam Hupp

With the controversy surrounding our last election there is an increased push to look at needed improvements to our outdated and error-prone voting technologies. Many people are raising the idea of using the Internet for voting, but what kind of risks would that entail? Avi Rubin, who has studied this area extensively, shared his insights and research.

Rubin was invited by the Costa Rican government to investigate the possible usage of electronic voter registration systems in their 1997 election. Voting is mandatory in Costa Rica, and a person must vote in the same district in which he or she first cast a ballot. This creates unique logistical problems the government was hoping to solve with computer systems. Their goal was to register people at any polling site using computers borrowed from schools. Several significant challenges were discovered during the trial. First, the high proportion of computer illiterate persons necessitated the use of light pens instead of mice. Trust was another problem, since the population would not necessarily trust a US-developed system. This was compounded by the fact that US cryptography export laws prevented the use of most encryption systems. In the end, Cost Rica's voting tribunal became worried about challenges to the new system and decided to cancel the trial.

There have been several other groups looking into this issue lately. The NSF hosted an e-voting workshop that brought together technologists, social scientists, election officials, and the US Department of Justice. The workshop concluded that the US is unprepared for remote electronic voting systems but that modernizing poll sites holds promise.

One of the cornerstones of any voting system is voter confidence. There must be confidence that all votes are counted, are counted only once, and remain private. Even as vexed as the most recent US presidential election was, these problems are still more acute in electronic voting systems. Additionally, electronic systems suffer from new problems, such as selective denial of service. What if a subtle denial of service attack was aimed at a carefully picked geographic area? In a close election this could be enough to change the outcome. Trojan horses and viruses pose another significant threat. With the proliferation of this malicious software, how could we trust the integrity of our computers for something as important as a national election?

Cryptographic protocols are a key component of any online voting system. Rubin described a system called "Sensus" developed by Lorrie Craner. Sensus uses blind signatures and a public key infrastructure (PKI) to provide many of the properties of a good voting system. Unfortunately, it is still vulnerable to non-cryptographic attacks, such as a lack of anonymity and denial of service. This illustrates some inherent problems with voting over the Internet.

A longer (2–3 week) voting period to combat the risk of DDoS attacks on voting systems would still not prevent selective denial attacks that subtly reduce service to targeted areas. Additionally, there is always the possibility of large-scale network failures over any time period, which could prevent the election from happening.

Online Privacy: Promise or Peril?

Lorrie Faith Cranor, AT&T Labs-Research

Summarized by Carson Gaspar

Online privacy has now become enough of an issue that it appears in comic strips. Several (rather humorous) examples from Cathy appeared throughout the talk. After the comic start, the talk moved into a brief overview of how private information can be transmitted without the user's explicit consent. "Browser chatter" refers to the extra information sent by Web browsers to Web servers, which includes the IP address, domain, organization, referrer, client platform (OS and browser), the information being requested, and cookies. This information is available to various parties, including the Web server, the server's sysadmin, one or more ISPs, possible third parties such as advertiser networks, and, potentially, to log files that can be subpoenaed. The talk then moved into more specific examples, with a discussion of Web bugs (an invisible image used to gather information) and inappropriate data in referrer headers (such as credit card information). Examples were given from several sites, most of which are now fixed or defunct.

The talk then moved from technology to political and social issues. Various surveys show that people are increasingly concerned about privacy. The European Union has acted on this, issuing a Data Directive that restricts how information can be collected and distributed. The United States has passed the Children's Online Privacy Protection Act and a few other pieces of legislation and industry-specific regulation, but it is far more piecemeal. Data collected by third parties is being subpoenaed increasingly often, both in criminal and in civil cases. The only way to avoid this is to not store the data in the first place.

Some solutions were then discussed. Voluntary privacy policies, privacy seal programs, legislation, corporate chief privacy officers, and client software can all help make things better, but none are a complete solution. The talk then focused on one particular technology: P3P (Platform for Privacy Preferences Project – *http://www.w3.org/P3P*). P3P provides a means for encoding a site's privacy policy into a machine-parseable format, and a means for a client to retrieve that policy and act on it. The server-side tools are available now, and client tools should start appearing by the end of 2001. Microsoft's IE6 beta already includes some minimal P3P support. The open question is will users obtain and use privacy software, even if it's free?

COMING TO GRIPS WITH SECURE DNS

Jim Reid, Nominum Inc.

Summarized by Chris Hayner

Secure DNS, or DNSSec, was developed as a way of validating the data in DNS lookups. The standard, described in RFC 2535, verifies the authentication of DNS responses and prevents spoofing attacks. The protocol uses DSA or RSA cryptography to digitally sign all DNS traffic.

This service, best implemented in BIND 9, does not do anything to stop DoS attacks. There is also the possibility that the DNS server has been compromised, and even though the signatures continue to be correct, the data could be incorrect. It is important to remember that even though secure DNS is implemented, there are still many Internet security holes to consider. The service also does not provide confidentiality of data. This is both because DNS data is public to begin with, and because in some cases an enormous amount of data would have to be encrypted, wasting a lot of time. Thus, only a hash of the DNS resource record is encrypted.

The new keys in a DNS record include: KEY, which represents the public keys of entities named in DNS and is used to distribute keys. SIG is used to authenticate the other resource records in the DNS response. NXT is used to deny the existence of a particular name in a zone. There is also a TTL, or time to live, set for the key and encrypted along with it. This prevents unscrupulous servers from setting unrealistically long TTLs in the plain-text field. Signatures also include a creation time and an invalidation time for keys. Thus, servers with knowledge of absolute time can easily determine if a key is still in effect.

Each zone would ideally be signed by its parent zone, thus creating a chain of trust all the way back to a root server. This leaves us with the obvious problem of where the chain begins. Therefore there is also the option to self-sign a zone, bringing the problem of authentication back onto the field. This is one of the many difficulties in bringing secure DNS into common use.

There is also a protocol called Transaction Signatures, which is a much simpler and much more inexpensive method of securing DNS transactions. It is a very simple protocol, allowing for authentication using shared-secret authentication. This can be used to authenticate responses as coming from the appropriate server. As yet there is no way of distributing the shared secret key.

ACTIVE CONTENT: REALLY NEAT TECHNOLOGY OR IMPENDING DISASTER?

Charlie Kaufman, Iris Associates

Summarized by Chris Hayner

Kaufman opened up his talk with the revelation that the "world's computing and communications infrastructure is a security disaster waiting to happen." To prove his point, Kaufman reminded us that most computers are connected to the Internet in some way, and that these computers have widely known, unpatched security vulnerabilities. He discussed how the animations, CSS, and rich text have anesthetized the masses to this danger.

Active Content is defined as something that is procedural, rather than interpreted data. Email that is simply read is interpreted, while email containing JavaScript and CSS is procedural, requiring a program to run locally to get the information out of the email. Other examples include Java or ActiveX on Web sites and executables and scripts sent and run as attachments.

The current security procedures against such things are very limited. Virus scans only detect known viruses, and firewalls can be avoided by email attachments, etc. Having to track down attackers is tiresome work, and even if the enemy is sighted, he may just be another victim, passing along the bad word. Using a different platform to work in the Internet is a very short-term solution, a solution which also robs users of the user-friendly tools available in Windows development models.

With one mistake, a computer must be assumed compromised and under the control of malicious malcontents. Disconnect from the network and reinstall is the only solution.

The problem has been with us since the beginning. As OSes have become more user friendly, they have naturally become less security conscious. The world was conquered by DOS, a software never meant to be networked to begin with. As the OS has gotten easier to use, the average user has become more naive to the inherent security risks associated with the Internet.

One possible solution is to use sandboxed applications. This would have the program run only what it needs, and prohibit random system calls. This has been implemented in Java. Problems include buggy sandboxing, the legitimate need some programs have for things such as saving state. Also, naïve users may improperly allow programs to override sandbox rule sets.

Having programs signed and authenticated by an authority could allow for security with attachments. This solution is limited by the configuration on either side, and unavailability of the authority for key authentication could cause delays.

Unicode application in browsers has opened a whole window of problems for stopping the execution of malignant code. This character set provides many different ways to say every letter, not all of which will necessarily be interpreted and blocked by the browser. Thus, writing all the possible permutations of a restricted tag could result in its execution.

The ultimate solution is to have OS-level protection from any application overstepping its bounds. Users should have the lowest level of privilege to be productive, and no more. There are applications like sudo for higher privileges.

Myths, Missteps, and Folklore in Protocol Design

Radia Perlman, Sun Microsystems Laboratories

Summarized by Kenneth G. Yocum

Dr. Perlman reminds us that she's going after the sacred cows. The audience giggles, apprehensively. She gives a couple of guidelines: learn from our mistakes, stop making them, and make new ones. She wants to talk about how we got where we are. She starts with "bridges and routers and switches, oh my!" This is because people who think they know the difference between these usually don't, and those that are confused by these terms do.

A brief overview of the ISO OSI reference model is given. Layer 1 is physical, layer 2 is link (neighbor to neighbor), layer 3 is talking across multi-hops, layer 4 is TCP, and layer 5 and above is boring. Everyone laughs. She goes on to highlight the confusion between bridges and routers. She is annoyed at the Infiniband people for leaving out a hop count. Ethernet muddled everything, and now you need a next hop address in addition to an ultimate destination. So layer 2 source/destination changes with each hop, but layer 3 stays constant. Thus routing algorithm had to be rethought. She dispels the myth that bridges came before routers. People thought Ethernet replaced layer 3, and they put protocols above it without layer 3. Her boss told her, we need a magic box between two Ethernets. She said, no, we need a router. But they said, no no no. Kludge it in without layer 3. And so the bridge was born. A box that listens to all and forwards everything to the other side. Ethernets can now scale physically, but you need a loop-free topology. Without a hop count, loops in your topology are evil. Evil is defined as exponential proliferation. With routers one packet remains solo. With bridges the packet gets repeated on multiple links, and cacophony ensues. Solution? A clever way to turn off certain links. Radia is clever. She creates a spanning tree algorithm. She reads a poem about it. It is funny. It is good. We laugh.

Radia finishes up with routers. She then moves on to IP multicast. She asks, "How did it get so complicated?" It doesn't have to be hard, she says. ATM had point-to-multipoint virtual circuits. One could add destinations to those virtual circuits. IP people wanted the joint to be initiated by a member, not root. That's OK too. Send a message to the root.

IP multicast API design axiom: it should look like Ethernet – multicast above layer 3 should look like multicast on top of layer 2 (Ethernet). Reality is there's no way to do this efficiently. Address allocation is a nightmare. She lists a variety of techniques: DVMRP, PIM-Dense mode, MOSPF, MSDP, and core-based trees (CBT). She lists the problems. She proposes address of eight bytes. Root of tree is intrinsic part and there are no root candidates. Choose root, ask root for address (root, G). Thus, apparently, addresses are trivial to administer, it's easy for routers to know who the root is, and addresses are plentiful. To quote Hoare: "There are two ways to design software. One way is to make it simple so it's obviously not deficient, the other is to make it so complicated there are no obvious deficiencies." We appreciate the relevance of this quote. She begins to delve into IPv6.

She wonders about the demise of CLNP, which was just like IP but had more addresses. It got killed by IETF when they said you can't replace IP with ISO. "Of course not," she says, "one's a packet format, the other's a standards committee." She dispels more IPv6 myths. It is good. Now she talks about unstable protocols. They are bad.

Example: ARPANET flooding. Because everything was homogeneous, they could find the problem. In about 20 hours. That was then, with 100 routers. Today it would be a huge disaster. Thus unstable protocols are bad, self-stabilizing protocols are good. No one argues. Radia knows much, and her logic is good. We are listening intently. She begins to talk about BGP.

She wonders, "Why isn't there just routing?" We use policy-based routing for inter-domain routing, and cost-based routing for intra-domain routing. But BGP doesn't support all policies. And it supports policies that don't converge, ever. The BGP specification "helps" by saying, "Don't do that."

There are more examples of bad protocol design. SSL version numbers whose field location changes. She argues that simplicity is good. Again, we listen, raptly. She summarizes.

The Internet has to be reliable and self-managing. Protocols have to be simple so that multiple vendor implementations have a hope in hell of working. In the presence of failure, it must at least be self-stabilizing. When you're making a protocol, her advice is, first know the problem you're trying to solve. She tells a story to illustrate this point. We love her stories. One day her child was crying in the hallway, holding his hand. She ran over, and said, "Everything will be OK!" kissing his hand to make it better. She asked, "What happened?" He said, "I got pee on my hand." Everyone laughs. People whistle and cheer. It is stupendous.

There are questions. The best one is, "Do you have any more stories?"

Strangely Enough, It All Turns Out Well (Adventures in Venture-Backed Startups and Microsoft Acquisitions)

Stephen R. Walli, Microsoft Corp.

Summarized by William R. Dieter

Walli discussed lessons he learned during the birth, development, and eventual acquisition of Softway Systems, a company he co-founded in 1995. He said the most important factor to the success of a startup is passion for the product. The founders must believe in the product. The book *Silicon Valley Way*, by Elton Sherwin, mirrors Walli's experience at Softway.

Softway started out with the idea of making POSIX compatibility work on NT. With just one person on the payroll and the other founders working other jobs to pay the bills, Softway met its first deadline in March of 1996. Despite front page press coverage at Uniforum that year, bootstrap funds were running out. In its first round of venture capital, Softway took $2.2 million, even though it was offered $5 million, because the founders wanted to retain control of the company. Walli believes not taking more money was a mistake. He said a company has big problems if the founders have to use their stock percentage to win votes on the board of directors. Later, the executive team found themselves spending more time raising money than running the company because of this mistake in both the first and second round of funding.

After the first round of funding, the company continued to grow, and it gained acceptance from some early adopters. By that time, the company of about 27 people was ready to make the leap to mainstream acceptance, as described in *Crossing the Chasm*, by Geoffrey Moore. One issue that Softway's management team did not fully understand was that to cross over the company had to do everything possible to get one big mainstream customer even if it meant neglecting other customers. Many large companies will not commit their business to a new product from a small company until they see other mainstream companies doing it. It is difficult to ignore smaller customers who are willing to pay for the product and who have been loyal in the early adopter phase, but who will not help win mainstream acceptance. In addition, it is crucial that employees know the company's goals so they can explain them to customers. Though Softway eventually got a deal with Dell to ship their product on every machine sold to the U.S. government, they were not able to get enough big mainstream customers to stay afloat.

By November of 1998 Softway had grown to around 40 people. Though Softway was bringing in around $2 million per year, it was still not profitable and money was drying up. When the cash started running out Softway had to lay people off. Layoffs are difficult for a startup because the management team often knows and has worked closely with those who are losing their jobs. Softway hired a banker to try to find a buyer for Softway. *Five Frogs on a Log*, by Mark Feldman and Michael Spratt, discusses what works and what can go wrong in mergers and acquisitions.

After prolonged negotiations that came tantalizingly close with several different companies, Microsoft agreed to buy Softway. All of the employees except for the executive team had to go through a hostile interview for a position at Microsoft. The Microsoft interview procedure is designed to hire only the best employees who will fit into the Microsoft culture. Microsoft's goal is to hire people who will be good for Microsoft first and for the particular position second. As part of Microsoft, Walli and former Softway employees had to adjust to the Microsoft culture detailed in *The 12 Simple Secrets of Microsoft Management*, by David Thielen.

If he had it to do again, Walli said he would take more money sooner because a little stock that's worth a lot is better than a lot of stock worth nothing. He would also be more particular when hiring and focus on "crossing the chasm." It is important to keep everyone focused on the company's mission. Walli said that he would "do it again in a heartbeat" if he found another product for which he had the same passion.

Several questioners wondered how Microsoft deals with employees who have made enough money not to worry about getting fired or raises. Walli replied that the Microsoft culture breeds relentless motion. Lower-level employees are driven by compensation that is closely tied to performance reviews. Those who are fully vested and no longer want to work generally quit and make way for those who are lower down on the ladder. David Thielen's book describes the process in more detail.

The Future of Virtual Machines: A VMware Perspective

Ed Bugnion, VMware Inc.

Summarized by Kartik Gopalan

Bugnion presented the state-of-the-art and future trends in Virtual Machine technology. He began by giving a historical perspective on virtual machines. The IBM mainframes in the 1960s and 1970s, such as IBM VM/370, were expensive and hence were designed with virtualization in mind in order to support efficient use of system resources. In the 1980s, as the desktop PC revolution began, hardware became cheaper and diverse, and the concept of virtualization was forgotten for a while. In the 1990s, Mendel Rosenblum, Ed Bugnion, and others began the Disco project, which aimed at running multiple commodity operating systems on multiprocessor MIPS machines. The project was named Disco since, at the time, virtualization was thought to be just another bad idea, like the disco music from the '70s. However, with budding interest in this technology, VMware took its present shape.

The principal challenges faced by the virtual machine concept were virtualization of the most prevalent IA-32 architecture, the diversity of present-day hardware, and acceptance of the idea by users. The result is the VMware workstation, which has the look and feel of a regular user-level application, and the VMware server, which has a Web-based management interface and remote console facility. Essentially, VMware provides an additional level of indirection between the operating system and the hardware, thus enabling the coexistence of "multiple worlds." A world consists of the OS, applications, and associated libraries.

The basic requirement of VMware is that the CPU needs to be virtualizable. Accordingly, CPU architectures can be classified as "strictly virtualizable" (such as Intel Alpha and Power PC) and "not strictly virtualizable" (such as IA-32 and IA-64). It is the latter class that is the most challenging but also the most useful in present-day scenarios.

The hosted VMware architecture allows a guest OS to execute on a machine where a host OS is already installed and running – for example, allowing Linux to run within a Windows NT environment. Advantages of this architecture are that the guest OS behaves as if it is just another application running on the host OS, the implementation is portable, and it works in the presence of other applications. However, it is limited by the scheduling decisions and resource management policies of the host OS, and it incurs heavy performance overheads due to world switches and during I/O accesses. One of the challenges in this architecture is virtualizing hardware, i.e., supporting any number of virtual devices.

The VMware ESX server architecture eliminates the need for a host OS. It is a micro-kernel-based architecture with a thin VM kernel sitting above the hardware and multiplexing hardware accesses by multiple guest OSes. The principal advantage of this approach is high performance I/O. It also opens up opportunities of customized resource management policies for each guest OS.

Some of the usage scenarios include testing and deployment of new software, server consolidation, allowing applications from multiple worlds to coexist on the same hardware platform, and security. One of the predicted future trends is that virtualization will have an impact on processor architecture and hardware designs. There will be more pressure to build designs that are easily virtualizable with minimum overheads, especially due to trends toward bigger servers and server consolidation. Virtualization, it is also predicted, will impact system software. Many problems, such as performance isolation, are better solved below the operating system. Further, operating systems will be optimized to run with VM, and there's a possibility that device drivers will be written for idealized devices rather than diverse real hardware. This would also allow new innovations in operating systems to take shape quickly and not be bogged down by hardware diversity. And the trend toward building compute clusters based on virtual machines and virtual storage would gain momentum.

Closing Session

The Art and Science of Sociable Machines

Dr. Cynthia Breazeal, MIT Media Lab

Summarized by Jon Stoffel

Dr. Breazeal's closing talk was a fascinating look at how humans and robots can interact, and the ideas behind this interaction.

She started off with a quick survey of robots in film, and how, since the 1950s, they have evolved from single use, barely humanoid robots, into more complex, interactive robots, evolving from HAL to C3PO to Data. She then showed a video of the Sony stand-alone robot SDR doing aerobics, dancing, and kung fu moves that showed how the autonomous state-of-the-art had advanced recently.

The core of the talk was about Kismet, a robotic infant designed by the Sociable Machines Project at MIT. Breazeal gave a quick history of autonomous robots which mirrored the evolution of science fiction robots. The Mars surveyor worked in a slow-changing environment, was isolated, had limited contact with us or other robots, and had prespecified tasks with strictly limited autonomy. RoboCup, a robot soccer league under development, involves a rapidly changing environment, robots that are autonomous but have to work in teams, and very specified tasks. Humanoid interactive robots will need to work in a very complex environment, perform open-ended tasks, be very autonomous, and interact in a complex manner.

The Sociable Machines Project decided to use an "infant caregiver" metaphor for their investigations. This was a change in the standard assumptions for the training environment of robots. The idea was to build the set of constraints from interacting with people, not preprogramming.

Some of the issues involved with a human-centric robot include deciding what matters, deciding when to try to solve a task, evaluating the results of an action, correcting improper actions, recognizing success, and structured learning.

Kismet is the result of their work. It combines elements of the appearance and personality of a human. The robot itself is just a head and neck on a box, but it mimics human child qualities of cuteness by portraying innocence, youth, and curiosity. Kismet is highly expressive, with lips, eyebrows, and big, mobile eyes.

During the talk, we saw several videos of Kismet interacting with women of all ages, from kids to adults. These videos can be found on their Web site (see below).

The interactions demonstrated various areas of perceptual and expressive systems that had been developed in Kismet. These included visual recognition algorithms which were implemented to include such features as "looking" preferences. Sometime Kismet would concentrate on the person's face, at other times it would search for and concentrate on the object being waved at it.

Kismet was also programmed to recognize "vocal affective intent" when people spoke. It was very funny and interesting to see how people used the visual feedback of the robot's shape to drop into baby talk when interacting with Kismet. Kismet was able to recognize and respond to various types of vocal noises including: soothing, attentive, and prohibitive.

A third area was Kismet's emotional systems, the expressive feedback that Kismet gave the user. To keep up the performance, the software consisted of small self-contained modules that were chained together. Kismet generates expressions in a virtual 3-D space, which it then uses to drive its response. The space includes axes of arousal/sleep, calm/excitement, and stress/depression.

Fourth, Kismet's emotive voice gave the user audible feedback. This was driven by the DECtalk speech system. The audience laughed at the disgusted and sad samples that were played.

The highlights of the talk were several videos of Kismet which pulled together all of these subsystems into a whole. They included, for example, Kismet's visual interactions and preferences, described above, and Kismet's being scolded (in German even!) until it would lower it's eyes and look downcast. These were very amazing for their lifelike feel; you started to forget on some levels that Kismet really was just a robot.

Breazeal then concluded her talk with a summary of where we are now and where future work needs to be done. All in all, this was a fascinating talk. For more information, visit *http://www.ai.mit.edu/projects/kismet*.

USENIX Quiz Show

Summarized by Josh Simon

As usual, Rob Kolstad closed the conference with another rousing Quiz Show. With all-new categories and topics this year (most of which were written on Saturday), the audience and contestants once again enjoyed themselves.

The contestants and scores were:

Group 1 Christopher Davis (3500), Steve McIntyre (2900), Perry Metzger (900)

Group 2 Andy Tannenbaum (2100), Mark Langston (2000), Matt Crosby (700)

Group 3 Ethan Miller (3500), Jim Larson (2900), Michael Buselli (1400)

In the finals:

Ethan Miller (5700) Christopher Davis (1700) Andy Tannenbaum (1300)

And in the Tournament of Champions:

Aaron Mandel (2100) Ethan Miller (2100) Trey Harris (1900)

In the tie-breaker Aaron scored 500 and Ethan scored 1000 to be the grand winner.

The USENIX Quiz Show has been produced by Rob Kolstad, Dan Klein, Dave Parter, and Josh Simon. Testers were Rik Farrow and Greg Rose. Special thanks to MSI for audio-video assistance. Prizes were provided by USENIX, Radware, O'Reilly, Prentice Hall, Addison-Wesley, ActiveState, Tandberg, SEI/CERT, and Integrated Computer Solutions. This has been a Klein/Kolstad Production. Copyright (c) 2001.

Photo Galleries

Several photo galleries of events at USENIX 2001 can be found at

http://www.usenix.org/publications/library/proceedings/usenix01/photos.html.

;login: welcomes submissions of photographs of USENIX events. Send us the URL for your particular gallery.