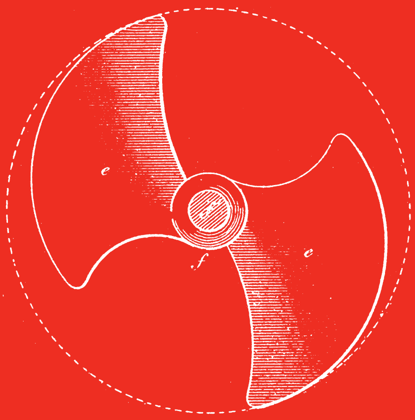
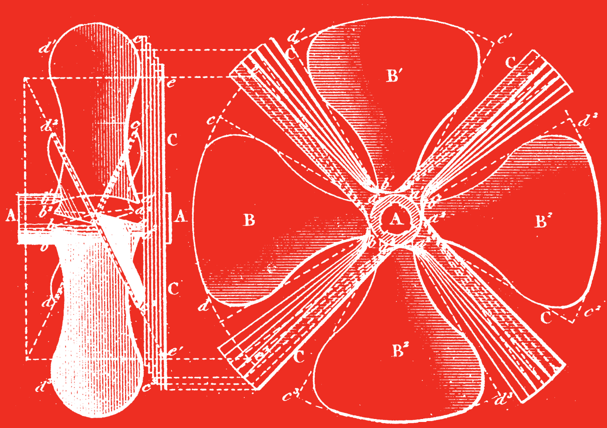
OCTOBER 2009 VOLUME 34 NUMBER 5

;login:

THE USENIX MAGAZINE





USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

23rd Large Installation System Administration Conference (LISA '09)

Sponsored by USENIX and SAGE in cooperation with LOPSA and SNIA

NOVEMBER 1–6, 2009, BALTIMORE, MD
http://www.usenix.org/lisa09

Symposium on Computer-Human Interaction for Management of Information Technology (CHIMIT '09)

Sponsored by ACM in association with USENIX

NOVEMBER 7–8, 2009, BALTIMORE, MD
http://www.chimit09.org/

ACM/IFIP/USENIX 10th International Middleware Conference

NOV. 30–DEC. 4, 2009, URBANA CHAMPAIGN, IL
http://middleware2009.cs.uiuc.edu/

First USENIX Workshop on Sustainable Information Technology (SustainIT '10)

Co-located with FAST '10

FEBRUARY 22, 2010, SAN JOSE, CA
http://www.usenix.org/sustainit10
Submissions due: November 9, 2009

8th USENIX Conference on File and Storage Technologies (FAST '10)

Sponsored by USENIX in cooperation with ACM SIGOPS

FEBRUARY 23–26, 2010, SAN JOSE, CA
http://www.usenix.org/fast10

3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '10)

Co-located with NSDI '10

APRIL 27, 2010, SAN JOSE, CA
http://www.usenix.org/leet10
Submissions due: February 25, 2010

2010 Internet Network Management Workshop (INM '10)

Co-located with NSDI '10

APRIL 27, 2010, SAN JOSE, CA
http://www.usenix.org/inm10
Paper registration due: November 30, 2009

7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 28–30, 2010, SAN JOSE, CA
http://www.usenix.org/nsdi10

2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '10)

JUNE 14–15, 2010, BERKELEY, CA
http://www.usenix.org/hotpar10
Submissions due: January 24, 2010

2010 USENIX Annual Technical Conference (USENIX '10)

JUNE 23–25, 2010, BOSTON, MA
http://www.usenix.org/usenix10
Submissions due: January 11, 2010

USENIX Conference on Web Application Development (WebApps '10)

JUNE 23–25, 2010, BOSTON, MA
http://www.usenix.org/webapps10
Submissions due: January 11, 2010

19th USENIX Security Symposium (USENIX Security '10)

AUGUST 11–13, 2010, WASHINGTON, DC

9th USENIX Symposium on Operating Systems Design and Implementation (OSDI '10)

OCTOBER 4–6, 2010, VANCOUVER, BC, CANADA

For a complete list of all USENIX & USENIX co-sponsored events, see http://www.usenix.org/events.

contents

;login:

VOL. 34, #5, OCTOBER 2009

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Jane-Ellen Long
Casey Henderson
Jennifer Peterson

TYPESETTER
Star Type
startype@comcast.net

USENIX ASSOCIATION
2560 Ninth Street,
Suite 215, Berkeley,
California 94710
Phone: (510) 528-8649
FAX: (510) 548-5738

http://www.usenix.org
http://www.sage.org

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

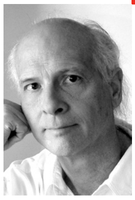
$90 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $125 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

RIK FARROW

musings



Rik is the Editor of *;login:*.

rik@usenix.org

I CAN SEE DARK, OMINOUS CLOUDS out my office window. It's been unusually dry here, although not nearly as dry as Robert Ferrell's home base, San Antonio. Perhaps the clouds I see will produce some much needed rain.

But it's not rain clouds, or the lack of them, that has sysadmins concerned these days. Instead, it's cloud computing that worries many. Cloud computing appears to be storming over the IT world, replacing local servers with ones somewhere "out there." If cloud computing takes over, many fear another wave of sysadmin job losses.

Appropriate use of cloud computing can save money as well as be more energy efficient. And, since it's the latest buzzword, every boss is wondering when his IT department will move "into the cloud," if only so that he can tell his golfing buddies about it.

I have my own worries about cloud computing, concerns over the security of data that will be stored and processed in the cloud. And I am not alone, either.

HotCloud

The HotCloud workshop summaries are included in this issue. I suggest reading these excellent summaries, in particular the report of the panel discussion in which Stefan Savage discusses some security concerns. Savage, like many others, pointed out that data stored off-site gets different US legal treatment from data stored on premises. A subpoena, something a judge must approve, may be required for access to some data stored off-site (Stored Communications Act [1]). Unless a cloud provider can guarantee that data will not be stored outside of the EU, and particularly not in the US, European Union users cannot use that cloud provider to store any confidential data.

Savage also pointed out that unless you are using Infrastructure as a Service (IaaS), you are relying on the cloud provider for privacy, storage availability, integrity, durability, and retention limits. Savage told of a cloud provider that lost client data, and the client had no recourse for the recovery of that data or for damages due to its loss.

Before I read the HotCloud summary, someone I know asked about the security of cloud computing and I came up with a different set of concerns. First, when you run your own servers, you control (or fail to control) the physical security of your servers. You have access to network infrastructure, file and backup storage, and servers themselves. Physical security is the base for computer security, and cloud computing turns this over to someone else.

You might be thinking that won't be a problem. After all, cloud server farms do have physical security, and will in many cases be able to arrange for better physical security than your organization could afford. But this brings about another dark idea. You do not get to hire the people running the cloud server farm, including those whose job it is to replace dead servers or drives in the hot, noisy racks.

You also lose the ability to monitor and log network traffic outside the hosted "server." Even if you don't routinely log traffic, you probably have had to do it when debugging a server on which performance suddenly dropped for no apparent reason. Running a tool like Argus [2] to collect connection logs is not only a good debugging tool, but also great for security audits. But in the cloud you have to hope your provider will do this for you. Right. A good cloud provider will keep logs, but sharing them with you will be difficult, because those logs reveal information about other hosted servers.

And your firewall will be included within the server, not outside it. You likely remember the mantra *security in depth*, but you no longer have an *outside* where you can put the firewall. An attacker who can elevate privileges can delete all logs (or perhaps just `rm -rf` everything), and you will no longer have a second source of logs outside the affected server. Unless you wisely have been saving logs locally, and not in the cloud, you will have lost your logs as well.

While few sites perform real forensics after an incident, your ability to look at drives after an incident will be gone too. Even if you want to find those deleted log files, one of the easiest things to do with disk forensics tools such as Sleuthkit [3], your deleted files really will be gone.

Even the little blinky lights on networking equipment that let you know that there are still packets reaching your server will be gone.

Virtual World

Servers in the cloud are hosted with other servers, sitting on top of VMs. The vendors of virtual machine monitor (VMM) solutions do their best to prevent exploits that can escape the boundaries of the virtual machine into the VMM, but it has happened. I just learned of an incident last week where a hosted server was properly secured, but another hosted server on the same hardware was exploited. The attacker then exploited the VMM and wiped all the other systems hosted under it, including the one properly secured. Of course, there are no disks handy where someone could perform forensics and prove this, but my acquaintance has been kicked out by his hosting provider for "attracting trouble."

Most server hardware today runs Intel or AMD processors, and these processors were not designed with virtualization in mind. These processors *do* have extensions to support virtualization, but these are for performance more than security. Real hardware support for virtual machines means that virtual machines are segregated using hardware beyond the memory management mechanisms (MM) used today. MM was designed to segregate processes, not virtual machines, but this is how it is being used today.

I published a column about virtualization security one year ago [4], and that column is still good reading today.

Virtualization certainly has its place. But if you care about the confidentiality of your data or are legally required to provide auditable, secure data store, you should not be moving into the cloud.

The Lineup

We start this issue with an article by Tom Limoncelli about software. Well, not quite, as Tom expounds on design decisions that fail to take into consideration installation, debugging, and maintenance as they affect the system administrators who manage this software.

Christoph Hellwig describes XFS, one of the file systems supported by Linux versions. During FAST '09, I overheard someone asking a Linux vendor why there needed to be more than one file system type, and I felt more than a little embarrassed. Hellwig explains how XFS is different from the default Linux file systems and when it should be used, and he provides some performance graphs to back up his assertions.

Kristaps Džonsons makes a strong case for creating better documentation. Džonsons says that mdoc comes closest to meeting his set of criteria for good documentation. He includes both syntactic regularity and semantic encapsulation, so that machines can interpret data and so that the documentation also works better for its human users.

Brandon Salmon and his co-authors have also written about file systems but take a very different perspective from Hellwig's. Salmon points out that users look at file systems very differently from the way software engineers and sysadmins do. iTunes, for example, groups music in its GUI very differently from how it lives in the hierarchically organized file systems where the data is actually stored. Perspective, their project, leverages semantic information for data management for home systems that includes intelligent file migration and backup.

Tasneem Brutch has written a survey of tools useful for compiling, profiling, and debugging programs destined for multicore systems. Parallel programming is hard, but there are a growing number of tools designed to make the task easier; Brutch does a great job of covering available tools, both open source and commercial.

Rudi Van Drunen continues his hardware series by discussing the trouble with static. Did you know that before you can even feel a static discharge the voltage has reached 3,000 volts? Rudi explains the sources of static, graphically shows the effects of static when frying microscopic circuitry, then covers countermeasures.

David Blank-Edelman completes his exposition of the Perl Web application framework, CGI::Application, begun in the August issue. He certainly makes things look easy.

Pete Galvin has written an extensive comparison of two new virtualization systems, VMware vSphere and Microsoft's Hyper-V (release 2). Pete, who has previously compared different Solaris-specific forms of virtualization [5], does a thorough job of comparing these two new offerings.

Dave Josephsen reveals a solution to authorized access using OpenVPN, OpenLDAP, and PF that he built in-house with a coworker. You can find the sources for the glue that makes this very cool system work in the online *;login:* at http://www.usenix.org/publications/login/2009-10/.

Robert Ferrell appears to be just as fond of cloud computing as I am, but has a very different manner of expressing his feelings.

Elizabeth Zwicky has reviewed the second edition of David Blank-Edelman's *Automating System Administration with Perl* in her usual style. She also describes a second book, on leadership, as "mostly painless." Then Dave Josephsen covers a book on the Android programming environment, followed by Brandon Ching on a book on OpenSolaris.

For summaries, we begin with the 2009 Annual Technical Conference, followed by the excellent summary of HotCloud written by Alva Couch and Kiran-Kumar Muniswamy-Reddy. Finally, we were fortunate to get some summaries from BSDCan, compiled by Royce Williams.

Those dark clouds I was watching never did produce any rain, unfortunately. And I suspect that everyone rushing into cloud computing will look back in one or two years and wonder why they were so eager to put most of their IT infrastructure, and precious data, into the cloud.

REFERENCES

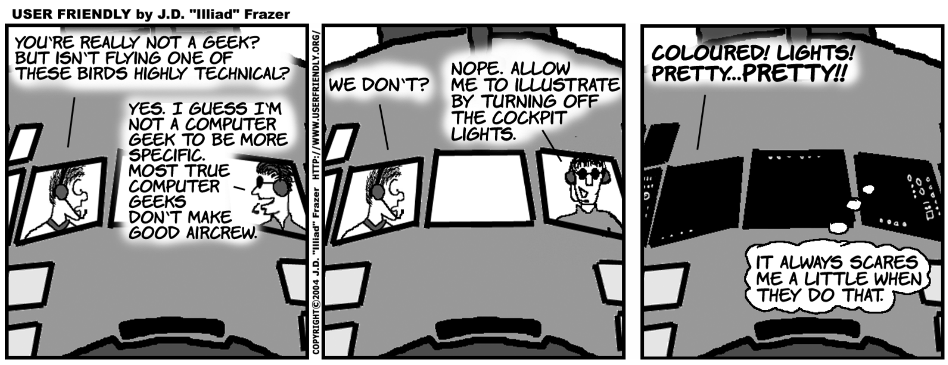
[1] EFF on the Stored Communications Act: http://ilt.eff.org/index.php/Privacy:_Stored_Communications_Act.

[2] Argus network audit and analysis: http://www.qosient.com/argus/.

[3] Sleuthkit and Autopsy open source forensics tools: http://www.sleuthkit.org/.

[4] Rik Farrow, "Musings," *;login:*, October 2008, vol. 33, no. 5: http://www.usenix.org/publications/login/2008-10/openpdfs/musings.pdf.

[5] Peter Galvin, "Solaris Virtualizations," *;login:*, April 2009, vol. 34, no. 2: http://www.usenix.org/publications/login/2009-04/pdfs/galvin.pdf.


USER FRIENDLY by J.D. "Illiad" Frazer
YOU'RE REALLY NOT A GEEK? BUT ISN'T FLYING ONE OF THESE BIRDS HIGHLY TECHNICAL?
YES. I GUESS I'M NOT A COMPUTER GEEK TO BE MORE SPECIFIC. MOST TRUE COMPUTER GEEKS DON'T MAKE GOOD AIRCREW.
COPYRIGHT © 2004 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
WE DON'T?
NOPE. ALLOW ME TO ILLUSTRATE BY TURNING OFF THE COCKPIT LIGHTS.
COLOURED! LIGHTS! PRETTY...PRETTY!!
IT ALWAYS SCARES ME A LITTLE WHEN THEY DO THAT.

THOMAS A. LIMONCELLI

Hey! I have to install and maintain this crap too, ya know!

Thomas A. Limoncelli has written or co-written four books, including *Time Management for System Administrators* (O'Reilly) and *The Practice of System and Network Administration* (Addison-Wesley). He is a system administrator at Google in NYC and blogs at http://EverythingSysadmin.com.

tal+usenix@everythingsysadmin.com

A SURE SIGN THAT SYSADMINS ARE misunderstood and undervalued is that many otherwise great products are difficult to install, maintain, or troubleshoot. Any sysadmin can tell if the installation process was designed as an afterthought. Any sysadmin can point to a variety of . . . I'll be polite and say "design decisions" that make a product difficult to install or completely and utterly impossible to troubleshoot.

A person purchasing a product is focused on the features and benefits and the salesperson is focused on closing the deal. If the topic of installation does come up, a user thinks, "Who cares! My sysadmin will install it for me!" as if such services are free. Ironically, it is the same non-technical executive who dismisses installation and upkeep as if they are "free" who might complain that IT costs are too high and go on a quest to kill IT spending. But I digress.

Installation Woes

I can understand why a product might be difficult to install. It is hard enough to write software, and with the shortage of software developers it seems perfectly reasonable that the installation script becomes an afterthought, possibly given to a low-ranking developer. The person purchasing the product usually requires certain features, and ease of installation is not a consideration during the procurement process. However, my ability to install a product affects my willingness to purchase more of the product.

At a previous job, we were able to massively deploy SGIs because their IRIX operating system installation could be automated. This made it a no-brainer to encourage use of SGIs. When SGI announced they were moving to the Windows operating system, our first question was whether our fully automated Windows installation system [1] could be adapted to their new hardware. We were told in no uncertain terms that this would not be possible for technical reasons related to their custom firmware. We never purchased any of those machines. While SGI's collapse can't be attributed to this one misstep, it did seem to be a symptom of a company that was losing touch with its customers.

Maintenance

Ongoing maintenance and upkeep have similar issues. There have been misguided attempts at making UNIX system administration easier by adding GUIs. A GUI is not automatically easier than command-line tools. Some GUIs get in the way. IBM famously layered a complicated set of commands on top of AIX and layered a complicated GUI on top of that. However, they did two things right. First, their "please wait" icon was adorable. Second, when selecting an action one could always press a function key to reveal the shell command line that was about to be executed. If you had to make the same change on 1,000 machines you did not have to mouse through the same clicks 1,000 times. You simply revealed the command and wrote a shell script to run that command on each machine. Much better.

Although I have not directly used ZFS, I am in awe of the attention paid to making the command line so simple [2]. It takes many times more effort to make a command do the right thing all the time than to simply add more options that an experienced sysadmin will know when to use. Similarly, anyone can add a new button to a GUI, but it takes serious investment of resources to improve the system so that the new button isn't needed.

Perfect Products

I've talked with product managers about why their product is the speedbump that slows me down when troubleshooting a problem that is buried in a network of 150 devices from 15 different companies. In the old days vendors would tell us, "That's why you should buy everything from one vendor—us!" In today's multi-platform arena we're told, "Our goal is to make our product so easy to use you don't need to debug it."

I'm sure that last sentence made you cringe. You get it.

Even a bug-free product requires the ability to troubleshoot problems, because the problems may not be directly related to that product. Imagine an Ethernet switch that is operating perfectly but a user's workstation is not seeing any network connectivity. The ability for a sysadmin to be able to check the status of the connection and verify settings is important in troubleshooting problems like this. Why would anyone make this task difficult? Ah yes, I remember what the product manager said. If the product is perfect it doesn't need troubleshooting.

I've explained to product managers that GUIs are bad when they prevent the basic principles of system administration: change management, automated auditing, backups, and unfettered troubleshooting. We have practices and methodologies we need to implement! Don't get in our way!

The more enlightened product managers understand that the easier it is to automate the installation of their product, the easier it is for me to buy a lot of their product. The more enlightened product managers understand that an ASCII configuration file can be checked in to Subversion, audited by a Perl script, or even generated automagically from a makefile. Sadly, those product managers are rare.

One would think that companies would be investing millions of dollars in research to make sure their products are beloved by sysadmins. This, however, can become a fool's errand.

It costs a lot of money to add features to make a product exceptionally easy to install, maintain, and troubleshoot. In fact, these features may be more difficult than features of the product itself. There are more edge cases and strange situations one must plan for. One may have a good relationship with the direct users of the software, but with the sysadmin organization hidden behind them? That's a lot to ask for. A product manager is expected to know a lot about a product and the industry that uses the product, but knowledge of sysadmin change management, auditing, backups, and such? How can we expect them to know such things when the sysadmin community finds itself at a loss for common terminology and design patterns?

A company could go out of business spending time on these features with very little payback. Enabling software so that a sysadmin can maintain zillions of installations requires specialized expertise, which is expensive to acquire. Scaling software to meet the needs of a single large customer has little payback, especially when such effort could go toward features that attract new customers with less elephantine needs. Spending resources there while your competitor spends money on slick color schemes and spinning icons leads to bankruptcy. In the security world this results in a marketplace where shoddy products are common and the truly great products can't get started [3].

Such features do pay off when customers pay attention to the total cost of ownership (TCO), especially as part of the purchase process. A product that saves money in one area but costs more in maintenance is soon detected when TCO is the focus. The truth is that for most products, operations are more expensive than acquisition. We buy a product once, but it runs hundreds, if not millions, of times. The cost of a hard disk is 20 percent of the cost of providing storage. The remaining 80 percent is consumed by controllers, backup systems, backup media, and other often hidden costs [4].

These operations would be less expensive if product houses could rely on a couple of basic rules of thumb or design patterns to carry them through the process. Shrink-wrapped software already benefits from this: By using common installers they leverage years of experience in getting installation right. As a bonus, commercial software installer kits have APIs to permit automated installs. Open source systems benefit from the user of Autoconf [5] and similar systems.

The solution is more research. More published research will result in a broader array of solutions. It enlarges our toolbox. It makes the world a better place.

I like to think that somewhere out there is a group of researchers studying this kind of thing. I imagine that they find sysadmins who volunteer to be videotaped as they do their job. I imagine the researchers (or their graduate students) poring over those tapes as they try to understand our strange ways. I imagine Dian Fossey studying not Gorillas in the Mist but Sysadmins at the Keyboard.

These researchers do exist.

I've seen them.

For the past two years they've met and exchanged ideas at a conference called CHIMIT (Computer-Human Interaction for Management of IT).

Some of them actually videotape sysadmins and examine what is it about products that makes our jobs more difficult and what makes them better.

My favorite moment was watching a researcher describing his observation of a sysadmin during the heat of a real outage. The sysadmin closed the firewall's GUI and connected to the command-line interface twice, each time in a different window. In one the sysadmin kept repeating a command to output some debugging information. In the other he typed commands to fix the problems. This was something the GUI would never have let him do without risking carpel tunnel syndrome. The researcher beamed as he explained the paradigm we were witnessing. He sounded like he had been lucky enough to catch the Loch Ness Monster on film, but what he had captured was something more valuable: photographic evidence of why sysadmins hate GUIs!

The person sitting next to me sighed and said, "Oh my god. Is that why nobody uses the GUI we spend millions to develop?" I nodded and smiled. The other sysadmins in the audience did too.

I love this conference.

These researchers study people like me and it makes the world a better place.

More than researchers attend. Sysadmins make up a large part of the audience.

The organizers point out that the conference is "an emerging area intersecting the practice and science of systems management, human computer interaction, and service sciences." They welcome participation from all these diverse fields.

This year CHIMIT will be in Baltimore, MD, November 7–9, immediately following LISA '09, which by an amazing coincidence is also in Baltimore, MD, on November 1–6.

Mark the dates on your calendar. See http://www.chimit09.org (and, of course, http://www.usenix.org/lisa09) for more information.

Will you be there? I know I will.

REFERENCES

[1] Fullmer and Levine, "AutoInstall for NT: Complete NT Installation Over the Network": http://www.usenix.org/publications/library/proceedings/lisa-nt98/fulmer.html.

[2] Sun Microsystems, "Solaris ZFS: Simplified Administration": http://www.sun.com/software/solaris/ds/zfs.jsp#4.

[3] Schneier, B., "A Security Market for Lemons," April 2007: http://www.schneier.com/blog/archives/2007/04/a_security_mark.html.

[4] Thomas A. Limoncelli, Christina J. Hogan, Strata R. Chalup, *The Practice of System and Network Administration*, 2nd edition (Addison-Wesley Professional, 2007), Chapter 25.

[5] GNU Project, Free Software Foundation, Autoconf: http://www.gnu.org/software/autoconf/.

CHRISTOPH HELLWIG

XFS: the big storage file system for Linux

Christoph Hellwig is a freelancer providing consulting, training, and, last but not least, contract programming for Linux storage and file systems. He has been working on Linux file systems since 2001 and is one of the most widely known developers in this area.

hch@lst.de

XFS IS A FILE SYSTEM THAT WAS DE-signed from day one for computer systems with large numbers of CPUs and large disk arrays. It focuses on supporting large files and good streaming I/O performance. It also has some interesting administrative features not supported by other Linux file systems. This article gives some background information on why XFS was created and how it differs from the familiar Linux file systems. You may discover that XFS is just what your project needs instead of making do with the default Linux file system.

BACKGROUND AND HISTORY

For years the standard Linux file system was ext2, a straightforward Berkeley FFS derivative. At the end of the 1990s, several competitors suddenly appeared to fill the gap for a file system providing fast crash recovery and transactional integrity for metadata. The clear winner in mainstream Linux is ext3, which added journaling on top of ext2 without many additional changes [7].

XFS has been less well known to many average Linux users but has always been the state of the art at the very high end. XFS itself did not originate on Linux but was first released on IRIX, a UNIX variant for SGI workstations and servers, in December 1994, almost 15 years ago. Starting in 1999, XFS was ported to Linux as part of SGI's push to use Linux and Intel's Itanium processors as the way forward for its high-end supercomputing systems. Designed from the start for large multiprocessor systems and disk arrays [1] rather than for small, single-disk consumer workstations, it was for a long time positioned above the mainstream Linux market. Today even low-end workstations with a small number of CPU cores and disks come close to the limits of ext3 (see Table 1). While there is another adaption of the FFS concept called ext4 under development to mitigate these limits to a certain extent, it seems as though basic FFS design is close to maxed out.

To address these limits, ext3 is evolving into ext4 by incorporating features pioneered by XFS such as delayed allocations and extents. Even with these improvements taking the basic FFS design as far as it can go, it is difficult to match the scalability limits of XFS, which has been designed for large storage systems from day one. In a few years, btrfs, a new file system initiated by Oracle, will mature from development status to hopefully become the new standard file system. As a new design that includes advanced management and self-healing features, btrfs will compete heavily with XFS on the lower end of the XFS market, but we will have to see how well it does on the extreme high end.

Today XFS is used by many well-known institutions, with CERN and Fermilab managing petabytes of storage for scientific experiments using XFS, and kernel.org serving the source code to the Linux kernel and many other projects from XFS file systems.

Limit	ext3	ext4	XFS
max file system size	16 TiB	16 TiB	16 EiB
max file size	2 TiB	16 TiB	8 EiB
max extent size	4 kiB	128 MiB	8 GiB
max extended attribute size	4 kiB	4 kiB	64 kiB
max inode number	2^{32}	2^{32}	2^{64}

(All numbers assume the maximum 4 kiB block size on x86 Linux systems.)

TABLE 1: FILE SYSTEM LIMITS FOR XFS, EXT3 AND EXT4

Space Allocation and Management

Each XFS file system is partitioned into regions called allocation groups (AGs). Allocation groups are somewhat similar to the block groups in ext3, but AGs are typically much larger than block groups and are used for scalability and parallelism rather than disk locality. Allocation groups are typically sized between 0.5 and 4 gigabytes and keep the size of the XFS data structures in a range where they can operate efficiently and in parallel [2].

Historical UNIX file systems, including ext3, use linear bitmaps to track free space, which is inefficient especially for larger contiguous allocations. XFS instead uses a pair of B+ trees in each allocation group. Each entry in the B+ tree nodes consists of a start block and length pair describing a free-space region. The first B+ tree is indexed by the starting block of the free region, and the other is indexed by the length of the free region. This double indexing allows the allocator to consider two goals for new data placement: locality to existing file data, and best fit into free space.

A similar extent concept is used for tracking the disk blocks assigned to each file. In addition to the start block on disk and the length of the contiguous range, the extent descriptor also contains two additional fields. The first one is the logical offset into the file, which allows for efficient sparse file support by skipping ranges that do not have blocks allocated to them. The second one is a simple one-bit flag to mark an extent as unwritten, a concept that will be explained later in this article.

For most files, a simple linear array of extent descriptors is embedded into the inode, avoiding additional metadata blocks and management overhead. For very large files or files containing many holes, the number of extents can be too large to fit directly into the inode.

In this case, extents are tracked by another B+ tree with its root in the inode. This tree is indexed by the offset into the file, which allows an extent descriptor for a given file offset to be found quickly, with no linear search overhead. Figure 1, showing the time needed to remove a large file such as an HD video or virtual machine image, demonstrates how management overhead can be reduced by using extents.

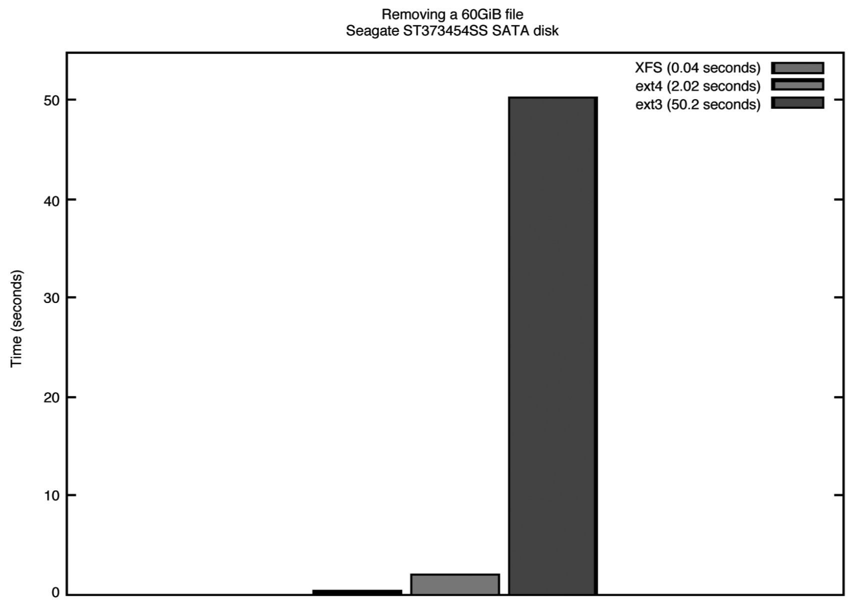


FIGURE 1: TIME SPENT REMOVING A VERY LARGE FILE

Inodes and Extended Attributes

The XFS inode consists of three parts: the inode core, the data fork, and the optional attribute fork. The inode core contains traditional UNIX inode metadata such as owner and group, number of blocks, timestamps, and a few XFS-specific additions such as project ID. The data fork contains the previously mentioned extent descriptors or the root of the extent map. The optional attribute fork contains the so-called extended attributes. The concept of extended attributes is not part of the Posix file system interface but is supported by all modern operating systems and file systems with slightly differing semantics. In Linux, extended attributes are simple name/value pairs assigned to a file that can be listed and read or written one attribute at a time. Extended attributes are used internally by Linux to implement access control lists (ACLs) and labels for SELinux, but they can also be used for storing arbitrary user metadata [3].

The attribute fork in XFS can either store extended attributes directly in the inode if the space required for the attributes is small enough, or use the same scheme of extent descriptors as described for the file data above to point to additional disk blocks. This allows XFS to support extended attribute sizes up to 64 kilobytes, while most other Linux file systems are limited to the size of a single disk block.

The size of the inode core is fixed, and the data and attribute forks share the remaining space in the inode, which is determined by the inode size chosen at file creation time, ranging from 256 to 2048 bytes. For file systems that extensively use ACLs (e.g., for Windows file shares exported by Samba) or for file systems making extensive use of extended attributes, choosing a larger inode size can provide performance improvements, because this extra data can be stored in the inode and does not require reading additional data blocks.

Inodes in XFS are dynamically allocated, which means that, unlike many other Linux file systems, their location and number are not determined at mkfs time. This means that there is no need to predict the expected number of inodes when creating the file system, with the possibility of under- or overprovision. Because every block in the file system can now possibly contain inodes, an additional data structure is needed to keep track of inode locations and allocations. For this, each allocation group contains another B+ tree tracking the inodes allocated within it.

Because of this, XFS uses a sparse inode number scheme where inode numbers encode the location of the inode on disk. While this has advantages when looking up inodes, it also means that for large file systems, inode numbers can easily exceed the range encodable by a 32-bit integer. Despite Linux's having supported 64-bit-wide inode numbers for over 10 years, many user-space applications on 32-bit systems still cannot accommodate large inode numbers. Thus by default XFS limits the allocation of inodes to the first allocation groups, in order to ensure all inode numbers fit into 32 bits. This can have a significant performance impact, however, and can be disabled with the inode64 mount option.

Directories

XFS supports two major forms of directories. If a directory contains only a few entries and is small enough to fit into the inode, a simple unsorted linear format can store all data inside the inode's data fork. The advantage of this format is that no external block is used and access to the directory is extremely fast, since it will already be completely cached in memory once it is accessed. Linear algorithms, however, do not scale to large directories with millions of entries. XFS thus again uses B+ trees to manage large directories. Compared to simple hashing schemes such as the htree option in ext3 and ext4, a full B+ tree provides better ordering of readdir results and allows for returning unused blocks to the space allocator when a directory shrinks. The much improved ordering of readdir results can be seen in Figure 2, which compares the read rates of files in readdir order in a directory with 100,000 entries.

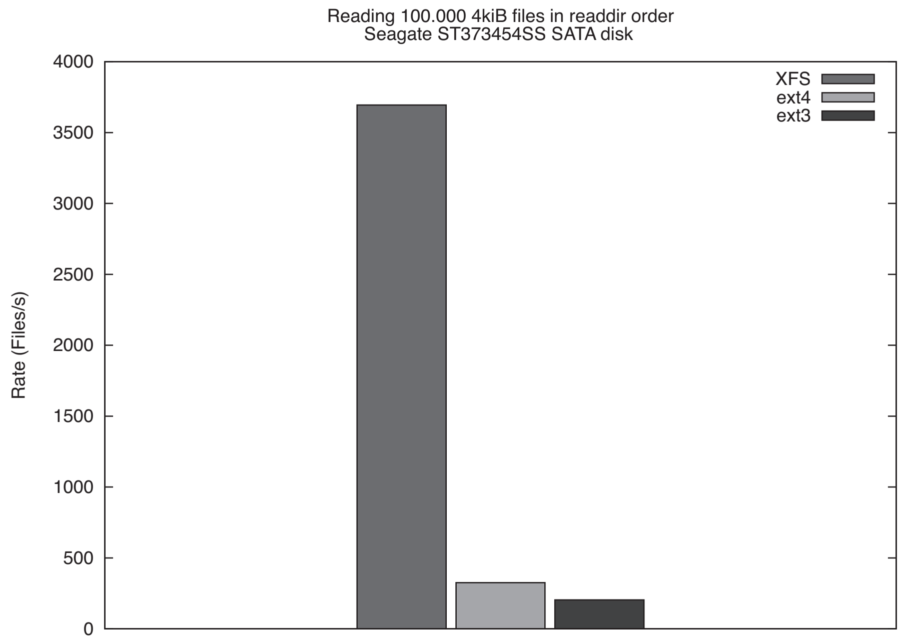


FIGURE 2: COMPARISON OF READING A LARGE (100,000 ENTRY) DIRECTORY, THEN READING EACH FILE

I/O Scalability

From day one, XFS has been designed to deal with high-performance disk subsystems, especially striped disk arrays with large aggregated bandwidth. When XFS was designed, "high performance" meant a few hundred megabytes per second, but 15 years later XFS still keeps up with aggregated bandwidth in the tens of gigabytes per second for a single file system instance [4].

To keep a RAID array busy, the file system should submit I/O requests that are large enough to span all disks in the array. In addition, I/O requests should be aligned to stripe boundaries where possible, to avoid read-modify-write cycles for common usage patterns. Because a single I/O can only be as large as a contiguous range of blocks, it is critical that files are allocated as contiguously as possible, to allow large I/O requests to be sent to the storage. The key to achieving large contiguous regions is a method known as "delayed allocation." In delayed allocation, specific disk locations are not chosen when a buffered write is submitted; only in-memory reservations take place. Actual disk blocks are not chosen by the allocator until the data is sent to disk due to memory pressure, periodic write-backs, or an explicit sync request. With delayed allocation, there is a much better approximation of the actual size of the file when deciding about the block placement on disk. In the best case the whole file may be in memory and can be allocated in one contiguous region. In practice XFS tends to allocate contiguous regions of 50 to 100 GiB when performing large sequential I/O, even when multiple threads write to the file system at the same time [4].

While delayed allocations help with random write workloads if sufficiently large contiguous regions are filled before the flush to disk, there are still many workloads where this is not the case. To avoid fragmentation in pathological cases with random writes filling up a file very slowly (such as some HPC workloads or BitTorrent clients), XFS allows preallocation of blocks on disk to a file before actually writing to it. The preallocation just assigns data blocks to a file without touching the block contents. To avoid security problems with exposing stale data, preallocated extents are marked as unwritten, and any read from them will return zeros. Once data is written to unwritten extents, they are converted to normal extents, which incurs minimal performance overhead compared to a write to a normal allocated extent.

Figures 3 and 4 show some performance enhancement when XFS is compared to ext3, the old Linux standard file system, and to ext4, which adds delayed allocations and extents to it for sequential I/O workloads. On the other hand, Figure 5 shows that the performance for completely random I/O is not only really bad but also doesn't profit much from the XFS features.

Direct I/O

XFS provides a feature, called direct I/O, that provides the semantics of a UNIX raw device inside the file system namespace. Reads and writes to a file opened for direct I/O bypass the kernel file cache and go directly from the user buffer to the underlying I/O hardware. Bypassing the file cache offers the application full control over the I/O request size and caching policy. Avoiding the copy into the kernel address space reduces the CPU utilization for large I/O requests significantly. Thus direct I/O allows applications such as databases, which were traditionally using raw devices, to operate within the file system hierarchy.

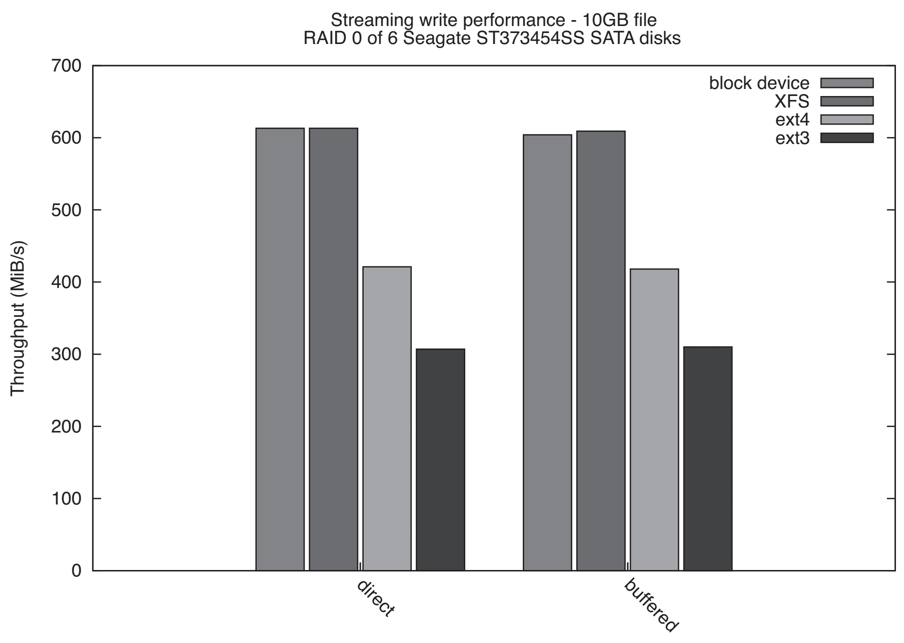


FIGURE 3: COMPARING BLOCK DEVICE, XFS, EXT4, AND EXT3 WHEN WRITING A 10 GB FILE

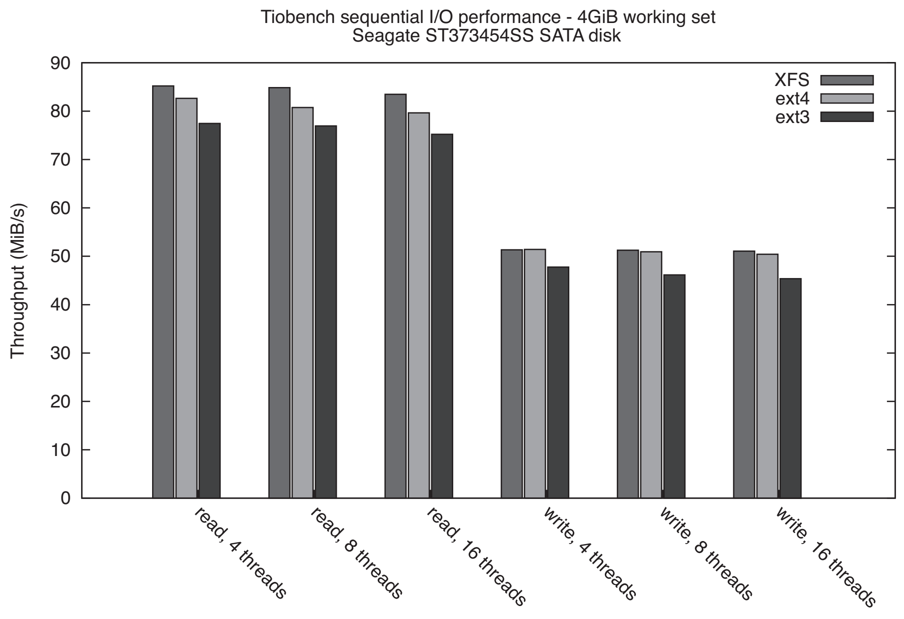


FIGURE 4: COMPARING SEQUENTIAL I/O PERFORMANCE BETWEEN XFS, EXT4, AND EXT3

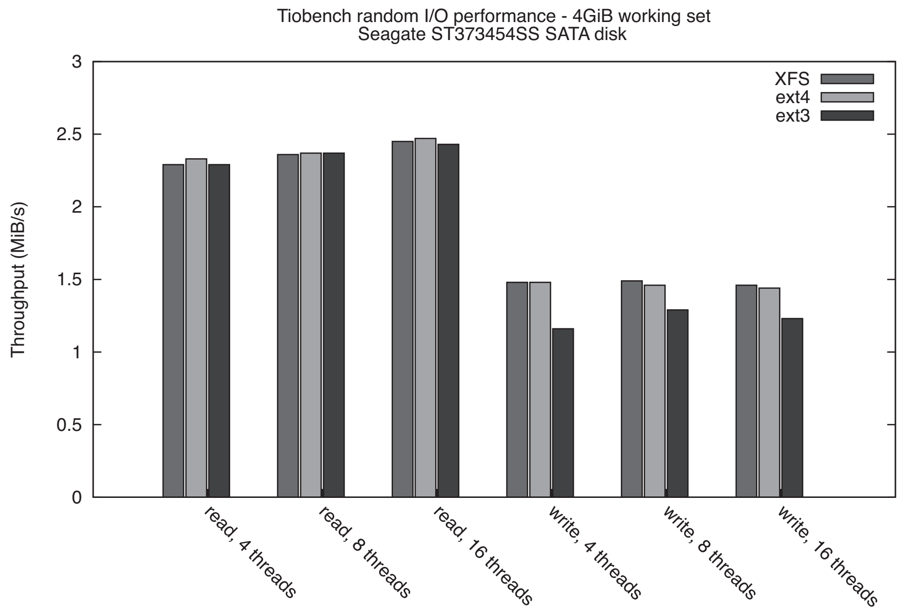


FIGURE 5: COMPARING RANDOM I/O PERFORMANCE BETWEEN XFS, EXT4, AND EXT3

Direct I/O has been adopted by all major Linux file systems, but the support outside of XFS is rather limited. While XFS guarantees the uncached I/O behavior under all circumstances, other file systems fall back to buffered I/O for many non-trivial cases such as appending writes, hole filling, or writing into preallocated blocks. A major semantic difference between direct I/O and buffered I/O in XFS is that XFS allows multiple parallel writers to files using direct I/O, instead of imposing the single-writer limit specified in Posix for buffered I/O. Serialization of I/O requests hitting the same region is left to the application, and thus allows databases to access a table in a single file in parallel from multiple threads or processes.

Crash Recovery

For today's large file systems, a full file system check on an unclean shutdown is not acceptable because it would take too long. To avoid the requirement for regular file system checks, XFS uses a write-ahead logging scheme that enables atomic updates of the file system. XFS only logs structural updates to the file system metadata, but not the actual user data, for which the Posix file system interface does not provide useful atomicity guarantees.

XFS logs every update to the file system data structures and does not batch changes from multiple transactions into a single log write, as is done by ext3. This means that XFS must write significantly more data to the log in case a single metadata structure gets modified again and again in short sequence (e.g., removing a large number of small files). To mitigate the impact of log writes to the system performance, an external log device can be used. With an external log the additional seeks on the main device are reduced, and the log can use the full sequential performance of the log device.

Unfortunately, transaction logging does not help to protect against hardware-induced errors. To deal with these problems, XFS has an offline file system checking and repair tool called xfs_repair. To deal with the ever growing disk sizes and worsening seek rates, xfs_repair has undergone a major overhaul in the past few years to perform efficient read-ahead and caching and to make use of multiple processors in SMP systems [6].

Disk Quotas

XFS provides an enhanced implementation of the BSD disk quotas. It supports the normal soft and hard limits for disk space usage and number of inodes as an integral part of the file system. Both the per-user and per-group quotas supported in BSD and other Linux file systems are supported. In addition to group quotas, XFS alternatively can support project quotas, where a project is an arbitrary integer identifier assigned by the system administrator. The project quota mechanism in XFS is used to implement directory tree quota, where a specified directory and all of the files and subdirectories below it are restricted to using a subset of the available space in the file system. For example, the sequence below restricts the size of the log files in /var/log to 1 gigabyte of space:

```
# mount -o prjquota /dev/sda6 /var

# echo 42:/var/log >> /etc/projects
# echo logfiles:42 >> /etc/projid
# xfs_quota -x -c 'project -s logfiles' /var
# xfs_quota -x -c 'limit -p bhard=1g logfiles' /var
```

Another enhancement in the XFS quota implementation is that the quota subsystem distinguishes between quota accounting and quota enforcement. Quota accounting must be turned on during mount time, while quota enforcement can be turned on and off at runtime. Using an XFS file system with quota accounting but no enforcement provides an efficient way to monitor disk usage. For this reason, XFS also accounts (but never enforces) quotas usage for the superuser.

The xfs_quota command [8] seen in the example above offers full access to all features in the XFS quota implementation. In addition, the standard BSD quota and edquota tools can be used to administer basic quota functionality.

Day-to-Day Use

A file system in use should be boring and mostly invisible to the system administrator and user. But to get to that state the file system must first be created. An XFS file system is created with the `mkfs.xfs` command, which is trivial to use:

```
# mkfs.xfs /dev/vg00/scratch
meta-data      =/dev/vg00/scratch   isize=256     agcount=4, agsize=1245184 blks
               =                    sectsz=512    attr=2
data           =                    bsize=4096    blocks=4980736, imaxpct=25
               =                    sunit=0       swidth=0 blks
naming         =version 2           bsize=4096    ascii-ci=0
log            =internal log        bsize=4096    blocks=2560, version=2
               =                    sectsz=512    sunit=0 blks, lazy-count=0
realtime       =none                extsz=4096    blocks=0, rtextents=0
```

As seen above, the `mkfs.xfs` command returns the geometry information for the file system to make sure all parameters are set correctly. There are not many parameters that must be manually set for normal use. For software RAID arrays, XFS already extracts the proper stripe size and alignment parameters from the underlying device, but if that is not possible (e.g., for hardware RAID controllers), the parameters can be set manually. The following example creates a file system aligned correctly for a RAID5 array with 8+1 disks and a stripe unit of 256 kiB:

```
# mkfs.xfs -d su=256k,sw=8 /dev/sdf
```

Other interesting options include using an external log device and changing the inode size. The example below creates a file system with 2 kiB-sized inodes and an external log device:

```
# mkfs.xfs -i size=2048 -l logdev=/dev/vg00/logdev /dev/vg00/home
```

For more details, see the `mkfs.xfs` man page and the XFS training material [5].

A command worth note is `xfs_fsr`. FSR stands for file system reorganizer and is the XFS equivalent to the Windows defrag tool. It allows defragmention of the extent lists of all files in a file system and can be run in background from cron. It may also be used on a single file.

Although all normal backup applications can be used for XFS file systems, the xfsdump command is specifically designed for XFS backup. Unlike traditional dump tools such as dumpe2fs for ext2 and ext3, xfsdump uses a special API to perform I/O based on file handles similar to those used in the NFS over the wire protocol. That way, xfsdump does not suffer from the inconsistent device snapshots on the raw block device that plague traditional dump tools. The `xfsdump` command can perform backups to regular files and tapes on local and remote systems, and it supports incremental backups with a sophisticated inventory management system.

XFS file systems can be grown while mounted using the xfs_growfs command, but there is not yet the ability to shrink.

Conclusion

This article gave a quick overview of the features of XFS, the Linux file system for large storage systems. I hope it clearly explains why Linux needs a file system that differs from the default and also shows the benefits of a file system designed for large storage from day one.

ACKNOWLEDGMENTS

I would like to thank Eric Sandeen for reviewing this article carefully.

REFERENCES

[1] Adam Sweeney et al., “Scalability in the XFS File System,” *Proceedings of the USENIX 1996 Annual Technical Conference*.

[2] Silicon Graphics Inc., XFS Filesystem Structure, 2nd edition, http://oss.sgi.com/projects/xfs/papers/xfs_filesystem_structure.pdf.

[3] Linux attr(5) man page: http://linux.die.net/man/5/attr.

[4] Dave Chinner and Jeremy Higdon, “Exploring High Bandwidth Filesystems on Large Systems,” *Proceedings of the Ottawa Linux Symposium 2006*: http://oss.sgi.com/projects/xfs/papers/ols2006/ols-2006-paper.pdf.

[5] Silicon Graphics Inc., XFS Overview and Internals: http://oss.sgi.com/projects/xfs/training/index.html.

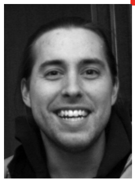
[6] Dave Chinner and Barry Naujok, Fixing XFS Filesystems Faster: http://mirror.linux.org.au/pub/linux.conf.au/2008/slides/135-fixing_xfs_faster.pdf.

[7] Dr. Stephen Tweedie, “EXT3, Journaling Filesystem,” transcript of a presentation at the Ottawa Linux Symposium 2000: http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html.

[8] xfs_quota(8)—Linux manpage: http://linux.die.net/man/8/xfs_quota.

KRISTAPS DŽONSONS

fixing on a standard language for UNIX manuals



Kristaps Džonsons is a graduate student in theoretical computer science at KTH/CSC. He also writes open source software, such as mdocml (mandoc), sysjail, and the mult forks of OpenBSD and NetBSD, with the BSD.lv Project.

kristaps@bsd.lv

"A UNIX UTILITY WITH POOR DOCUmentation is of no utility at all." When sitting down to document utilities and file formats, devices, system calls, and games, there are many UNIX manual formats to choose from, each suffering from limitations.

In this article I survey the cadre of formats and propose fixing on a standard, a format optimally serving readers and writers. I begin by defining the applicable environment, where manuals are written and read, then enumerate criteria for a standard within that space. Among the formats surveyed, I determine that mdoc suffers the fewest limitations. mdoc is popular in BSD UNIX, but it is available pre-installed on any modern UNIX system, from GNU/Linux to Mac OS X to OpenSolaris.

First, it's important to ask: Why doesn't a standard already exist? In short, the current spread of formats—diverse as it may be—is *good enough*. UNIX users, programmers, and administrators tolerate the menagerie so long as the output of the man utility is roughly consistent. I propose that the benefits of fixing on a standard, from consistent authorship to powerful analytical tools, stipulate only a minimal burden of change: policy creation, education of authors, and slow migration from substandard formats.

The troff Condition

We generally associate the man utility with documentation, but, internally, it only locates manuals, invokes an output formatter, then pages to the screen. This formatter constitutes the primary mechanism of manual production. UNIX systems overwhelmingly use troff [2] as a formatter, usually in the form of a modern implementation such as GNU troff (groff) [3], or Heirloom troff [4]. I'll refer to "troff" as a stand-in term for any of these implementations.

I define a format as *reasonable* only if it's accepted by troff with specific, documented utility for formatting UNIX manuals. A format is *semi-reasonable* if it's indirectly accepted—losslessly transformed into an accepted form by an existing intermediate translation utility. In this study, I consider only reasonable and semi-reasonable formats.

An example of an unreasonable format is HTML, which is neither accepted by troff, losslessly translatable, nor has a UNIX manual mode. The panoply of common word-processing formats, such as the Open Document Format and Rich Text Format, are similarly unreasonable.

The roff me, ms, and mm macro packages, while accepted by troff and occasionally used for older manuals, are not considered as having a specific utility for UNIX manuals; thus, I consider them unreasonable. texinfo [5], while being used for general documentation, is also not specifically used for UNIX manuals and is therefore unreasonable.

Criteria

I define the set of standardization criteria as follows: *structural readability*, such that end users are presented with structurally consistent output between manuals; *syntactic regularity*, such that machines may disambiguously scan and parse input; and a rich *semantic encapsulation* for meaningful machine interpretation of contextual data.

We're comfortable with conventional man output: margin widths, text decorations, and so on. Structural readability stipulates consistent output given a heterogeneous set of input documents. Syntactic regularity is both a formal term, regarding grammar, and a subjective one, regarding the writer's ease of composition. In this article, I focus on the former: input languages must be reliably machine-parseable. Lastly, semantic encapsulation requires the annotation of information. Meaningful manual terms, such as function prototypes and cross-links, must be disambiguously annotated, as machines cannot reliably classify context in unstructured text.

By fixing on a language that meets these criteria, we guarantee maximum, meaningful exposure of our manual, and we expand the end user's documentation tool set—these days, necessarily constrained by the chaos of volatile conventions and irregular formats—with sophisticated tools for cross-referencing, formatting, and so on.

Survey: man and POD

troff accepts the "roff" type-setting language as input; however, direct usage of roff has been eclipsed by the use of macro packages simplifying the language—macros, like procedural functions, are a roff language feature allowing complex macro blocks to be referenced by a simple call. troff internally replaces these macros with roff during a pre-processing phase.

The man macro package became the first common format for creating UNIX manuals (predated by the mm and me packages) and established the backspace-encoded, 78-column display style enjoyed to this day.

```
.SH SYNOPSIS
.B find
[\fB\-dHhLXx\fR]
```

FIGURE 1: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH MAN

The fragment in Figure 1 illustrates a manual's synopsis section: the SH macro (all roff macros appear on lines beginning with the '.' control character) indicates section titles, and B applies a boldface type to its argument. The argument string is bracketed by boldface character escapes. In general, man macros describe the presentation of terms.

```
==head1 SYNOPSIS

    B<find> S<[ B<-dHhLXx> ]>
```

FIGURE 2: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH POD

The man format also forms the basis for the Perl "POD" (Plain Old Documentation) language, illustrated in Figure 2. POD, like man, is a presentation format.

Survey: mdoc

The other roff manual-formatting macro package is mdoc, which, beyond sharing common ancestry, is fundamentally different from man. Instead of annotating presentation, mdoc semantically annotates its terms. In Figure 3, for example, Op indicates an option string, usually displayed as enclosed in brackets, followed by a series of flags offset by the Fl macro. The proper presentation of these macros is managed by the formatter.

```
.Sh SYNOPSIS
.Nm find
.Op Fl dHhLXx
```

FIGURE 3: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH MDOC

Both man and mdoc are accepted natively by troff. POD is the default format for embedding manuals in Perl documents, and it translates directly into man with the perlpod utility for indirect acceptance by troff.

Survey: DocBook

The DocBook [6] suite, like troff, is a general-purpose typesetter. Unlike troff, its input language, also called "DocBook," is based on XML (historically, SGML). DocBook has a schema for annotating UNIX manuals, illustrated in Figure 4, translating into man with docbook2x and docbook-to-man for further compilation by troff.

```
<refsynopsisdiv>
        <cmdsynopsis>
                <command>find</command>
                <arg choice="opt">
                        <option>dHhLXx</option>
                </arg>
        </cmdsynopsis>
</refsynopsisdiv>
```

FIGURE 4: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH DOCBOOK

The necessary complexity of processing XML demands a significant infrastructure of compilers and schemas to correctly transform materials. docbook-to-man (which operates only on SGML DocBook) requires an SGML parser, the appropriate DTD files, and a driving script. Importantly, existing tools for translation only produce man-lossy transition from semantically encoded to presentation-encoded documents.

Evaluation

The criteria described earlier in this article were structural readability, syntactic regularity, and semantic encapsulation. I noted that these criteria only apply to reasonable or semi-reasonable formats.

By virtue of being directly accepted by troff, mdoc and man are both eminently reasonable. DocBook and POD, on the other hand, require specialized utilities to translate input into man. Although these utilities must in general be downloaded and installed, their popularity makes them readily available on most systems, and thus they are semi-reasonable.

The matter of structural readability may be reduced to the author's level of influence on presentation. DocBook and mdoc manage presentation, while man and POD must be styled by the author. Given a non-uniform distribution of authors, it's safe to say that mdoc and DocBook satisfy readability more readily than the presentation languages. In other words, an author's control over function prototype styling will almost certainly produce varied output.

Syntactic regularity is both grammatical and structural. DocBook, by virtue of XML, follows a context-free grammar (upon combination with the tag schema); mdoc, man, and POD are context-sensitive. The matter of structural regularity, on the other hand, is largely subjective; some prefer the terseness of roff macros, while others prefer more descriptive DocBook tags.

In general, it's safe to say that DocBook's context-free foundationspromotes its syntactic regularity above the others. The matter of structural regularity, while important, remains a subjective matter.

The last criterion, semantic encapsulation, is by far the most significant in terms of meaningful analysis of data. POD and man, as with any presentation language, are semantically opaque: beyond using heuristic analysis, the content of these manuals is closed to machine interpretation.

```
\fIvoid\fP \fBexit\fP \*(lp\fIint\fP\*(rp
```

FIGURE 5: FUNCTION PROTOTYPE ENCODED IN MAN

DocBook and mdoc, however, are rich with semantic meaning; by careful analysis of the parse tree, machines can cross-link references, group terms, and perform many other useful operations. Figures 5 and 6 illustrate presentation and semantic encapsulation, respectively.

```
.Ft intmax_t
.Fn imaxabs "intmax_t j"
```

FIGURE 6: FUNCTION PROTOTYPE ENCODED IN MDOC

As noted earlier, DocBook's translation tools don't currently produce mdoc, which amounts to a lossy transform. Thus, while DocBook itself may be semantically rich, its intermediate format, and thus troff input, is not.

The format fitting all criteria with the fewest limitations is mdoc, featuring a reasonable, semantically rich language for manual data annotation. The lossy translation of DocBook to man, as well as its requirement of downloading additional processing tools, render it substandard.

The man and POD formats, as presentation languages, are opaque to machine interpretation. I consider this an insurmountable limitation, since it prohibits meaningful analysis of manual data.

Adoption

The hindrance of mdoc's widespread adoption is as much due to its poor exposure beyond the BSD UNIX community as to the limited semantic functionality of its popular compiler, groff.

Documentation for the mdoc format is, at this time, constrained to templates, the formidable mdoc.samples manual distributed with most BSD UNIX operating systems, and the minimal mdoc manual in general UNIX systems. Furthermore, unlike man, which exports few macros, the complexity of mdoc, with well over 100 available macros, makes introductory reference materials critical.

Although serving to format mdoc manuals for regular output, groff offers no semantic-recognition features: for example, HTML output (via grohtml) correctly cross-referencing manual references. This is a matter of groff's design, which internally translates mdoc into a presentation-based intermediate form, thus losing the semantic annotations of the input.

Fortunately, groff's limitations are being addressed by the mandoc [1] utility, which exports a regular syntax tree of mdoc input (and man, within the limitations of presentation encoding) for analysis. The issues of good introductory documentation and exposure, unfortunately, remain unsatisfied.

Conclusion

By using mdoc to write manuals, powerful documentation analysis is made considerably easier—arguably, by using man, POD, or a similar presentation format, meaningful analysis isn't possible at all. This is demonstrated by the total lack of manual analysis beyond the man, apropos, and whatis utilities, and various patchwork presentation services (such as man.cgi [7] and man2web [8]) in use today. Attractive, cross-referenced hypertext references, section-by-section querying of local manual sets, and other possibilities arise by fixing on mdoc, possibilities hindered by the preponderance of presentation-based, opaque languages.

REFERENCES

[1] mdocml: http://mdocml.bsd.lv.

[2] troff: http://www.troff.org.

[3] groff: http://www.gnu.org/software/groff/.

[4] heirloom: http://heirloom.sf.net/doctools.html.

[5] texinfo: http://www.gnu.org/software/texinfo/.

[6] docbook: http://www.docbook.org.

[7] mancgi: http://www.freebsd.org/cgi/man.cgi/help.html.

[8] man2web: http://man2web.sf.net.

BRANDON SALMON, STEVEN W. SCHLOSSER, LORRIE FAITH CRANOR, AND GREGORY R. GANGER

perspective: semantic data management for the home

Brandon Salmon is finishing his PhD at Carnegie Mellon and will be joining Tintri in the fall. He is interested in bringing user-centered design techniques, which have been so effective at improving the usability of user interfaces, to bear on system architectures, making them easier to understand and more accommodating of users' social systems.

bsalmon@cs.stanford.edu

Steve Schlosser received his PhD from Carnegie Mellon University in 2004, studying the use of alternative storage technologies in computer systems. He was a senior researcher at Intel Research Pittsburgh from 2004 to 2009 and is now a member of the technical staff at Avere Systems.

schlos@averesys.com

Lorrie Faith Cranor is an associate professor of computer science and of engineering and public policy at Carnegie Mellon University, where she is director of the CyLab Usable Privacy and Security Laboratory (CUPS). See http://lorrie.cranor.org/.

lorrie@cmu.edu

Greg Ganger is a professor of electrical and computer engineering at Carnegie Mellon University and the director of the Parallel Data Lab (PDL). His broad research interests in computer systems includes storage, OS, security, and distributed systems.

ganger@ece.cmu.edu

DISTRIBUTED STORAGE IS COMING home. An increasing number of home and personal electronic devices create, use, and display digitized forms of music, images, and videos, as well as more conventional files (e.g., financial records and contact lists). In-home networks enable these devices to communicate, and a variety of device-specific and datatype-specific tools are emerging. The transition to digital homes gives exciting new capabilities to users, but it also makes them responsible for administration tasks which in other settings are usually handled by dedicated professionals.

It is unclear that traditional data management practices will work for "normal people" reluctant to put time into administration. For example, most home users are accustomed to semantic organization (via applications such as iTunes) when accessing their data for daily use, but are forced by filesystem design to use a hierarchy when managing this same data.

We present the Perspective distributed file system, part of an expedition into this new domain for distributed storage. You can think of Perspective as in "Seeing many views, one gains Perspective." One focus of Perspective is simplifying data management tasks for home users. For example, Perspective's design allows home users to manage their data using the same semantic primitives they utilize for daily access. As with previous expeditions into new computing paradigms, it is in order to gain experience that we are building and utilizing a system representing the vision. In this case, however, the researchers are not representative of the user population. Most users will be non-technical people who just want to use the system but must (grudgingly) deal with administration tasks or live with the consequences. Thus, organized user studies will be required as complements to systems experimentation.

Perspective's design is motivated by a contextual analysis and early deployment experiences [3]. Our interactions with users have made clear the need for decentralization, selective replication, and support for device mobility and dynamic membership. An intriguing lesson is that home users rarely organize and access their data via traditional hierarchical naming—usually they do so based on data attributes. Computing researchers have long talked about attribute-based data navigation (e.g., semantic file systems [1]), while continuing to use directory hierarchies. However, users of home and personal storage live it. Popular interfaces (e.g., iTunes, iPhoto, and even drop-down lists of recently opened Word documents) allow users to navigate file collections via attributes such as publisher-provided metadata, extracted keywords, and date/time. Usually, files are still stored in underlying hierarchical file systems, but users often are insulated from naming at that level and are oblivious to where in the namespace given files end up.

Users have readily adopted these higher-level navigation interfaces, leading to a proliferation of semantic data location tools. In contrast, the abstractions provided by file systems for managing files have remained tightly tied to hierarchical namespaces. For example, most tools require that specific subtrees be identified, by name or by "volumes" containing them, in order to perform *replica management tasks*, such as partitioning data across computers for capacity management or specifying that multiple copies of certain data be kept for reliability. Since home users double as their own system administrators, this disconnect between interface styles (semantic for data access activities and hierarchical for management tasks) naturally creates difficulties.

The Perspective distributed file system allows a collection of devices to share storage without requiring a central server. Each device holds a subset of the data and can access data stored on any other (currently connected) device. However, Perspective does not restrict the subset stored on each device to traditional volumes or subtrees. To correct the disconnect between semantic data access and hierarchical replica management, Perspective replaces the traditional volume abstraction with a new primitive we call a view. A *view* is a compact description of a set of files, expressed much like a search query, and a device on which that data should be stored. For example, one view might be *"all files with type=music and artist=Beatles stored on Liz's iPod"* and another *"all files with owner=Liz stored on Liz's laptop."* Each device participating in Perspective maintains and publishes one or more views to describe the files it stores. Perspective ensures that any file that matches a view will eventually be stored on the device named in the view.

Since views describe sets of files using the same attribute-based style as users' other tools, view-based management is easier than hierarchical file management. A user can see what is stored where, in a human-readable fashion, by examining the set of views in the system. She can control replication and data placement by changing the views of one or more devices. Views allow sets of files to overlap and to be described independently of namespace structure, removing the need for users to worry about application-internal file naming decisions or difficult volume boundaries. Semantic management can also be useful for local management tasks, such as setting file attributes and security, as well as for replica management. In addition to anecdotal experiences, an extensive lab study confirms that view-based management is easier for users than volume-based management [4].

Our Perspective prototype is a user-level file system which runs on Linux and OS X. In our deployments, Perspective provides normal file storage as well as being the backing store for iTunes and MythTV in one household and in our research environment lounge.

Storage for the Home

The home is different from an enterprise. Most notably, there are no sysadmins—household members generally deal with administration (or don't) themselves. The users also interact with their home storage differently, since most of it is for convenience and enjoyment rather than employment. However, much of the data stored in home systems, such as family photos, is both important and irreplaceable, so home storage systems must provide high levels of reliability in spite of lax management practices. Not surprisingly, we believe that home storage's unique requirements would be best served by a design different from enterprise storage. This section outlines insights gained from studying use of storage in real homes and design features suggested by them.

WHAT USERS WANT

A contextual analysis is an HCI research technique that provides a wealth of *in situ* data, perspectives, and real-world anecdotes on the use of technology. It consists of interviews conducted in the context of the environment under study. To better understand home storage, we extensively interviewed all members of eight households (24 people total) in their homes and with all of their storage devices present. We have also gathered experiences from early deployments in real homes. This section lists some guiding insights (with more detailed information available in technical reports [3]).

Decentralized and dynamic: The users in our study employed a wide variety of computers and devices. While it was not uncommon for them to have a set of primary devices at any given point in time, the set changed rapidly, the boundaries between the devices were porous, and different data was "homed" on different devices with no central server. One household had set up a home server, at one point, but did not re-establish it when they upgraded the machine due to setup complexity.

Money matters: While the cost of storage continues to decrease, our interviews showed that cost remains a critical concern for home users (note that our studies were conducted well before the fall 2008 economic crisis). While the same is true of enterprises, home storage rarely has a clear "return on investment," and the cost is instead balanced against other needs (e.g., new shoes for the kids) or other forms of enjoyment. Thus, users replicate selectively, and many adopted cumbersome data management strategies to save money.

Semantic naming: Most users navigated their data via attribute-based naming schemes provided by applications such as iPhoto, iTunes, and the like. Of course, these applications stored the content in files in the underlying hierarchical file system, but users rarely knew where. This disconnect created problems when they needed to make manual copies or configure backup/synchronization tools.

Need to feel in control: Many approaches to manageability in the home tout automation as the answer. While automation is needed, the users expressed a need to understand and sometimes control the decisions being made. For example, only 2 of the 14 users who backed up data used backup tools. The most commonly cited reason was that they did not understand what the tool was doing and, thus, found it more difficult to use the tool than to do the task by hand.

Infrequent, explicit data placement: Only 2 of 24 users had devices on which they regularly placed data in anticipation of needs in the near future. Instead, most users decided on a type of data that belonged on a device (e.g., "all my music" or "files for this semester") and rarely revisited these decisions—usually only when prompted by environmental changes. Many did regularly copy new files matching each device's data criteria onto it.

From the insights above, we extract guidance that has informed our design of Perspective.

Peer-to-peer architecture: While centralization can be appealing from a system simplicity standpoint and has been a key feature in many distributed file systems, it seems to be a non-starter with home users. Not only do many users struggle with the concept of managing a central server, many will be unwilling to invest the money necessary to build a server with sufficient capacity and reliability. We believe that a decentralized, peer-to-peer architecture more cleanly matches the realities we encountered in our contextual analysis.

Single class of replicas: Many previous systems have differentiated between two classes: permanent replicas stored on server devices and temporary replicas stored on client devices (e.g., to provide mobility) [5, 2]. While this distinction can simplify system design, it introduces extra complexity for users and prevents users from utilizing the capacity on client devices for reliability, which can be important for cost-conscious home consumers. Having only a single replica class removes the client-server distinction from the user's perception and allows all peers to contribute capacity to reliability.

Semantic naming for management: Using the same type of naming for both data access and management should be much easier for users who serve as their own administrators. Since home storage users have chosen semantic interfaces for data navigation, replica management tools should be adapted accordingly—users should be able to specify replica management policies applied to sets of files identified by semantic naming.

In theory, applications could limit the mismatch by aligning the underlying hierarchy to the application representation, but this alternative seems untenable in practice. It would limit the number of attributes that could be handled, lock the data into a representation for a particular application, and force the user to sort data in the way the application desires. Worse, for data shared across applications, vendors would have to agree on a common underlying namespace organization.

Rule-based data placement: Users want to be able to specify file types (e.g., "Jerry's music files") that should be stored on particular devices. The system should allow such rules to be expressed by users and enforced by the system as new files are created. In addition to helping users to get the right data onto the right devices, such support will help users to express specific replication rules at the right granularity to balance their reliability and cost goals.

Transparent automation: Automation can simplify storage management, but many home users (like enterprise sysadmins) insist on understanding and being able to affect the decisions made. By having automation tools use the same flexible semantic naming schemes as users do normally, it should be possible to create interfaces that express human-readable policy descriptions and allow users to understand automated decisions.

Perspective Architecture

Perspective is a distributed file system designed for home users. It is decentralized, enables any device to store and access any data, and allows decisions about what is stored where to be expressed or viewed semantically.

Perspective provides flexible and comprehensible file organization through the use of *views*. A view is a concise description of the data stored on a given device. Each view describes a particular set of data, defined by a semantic query, and a device on which the data is stored. A view-based replica management system guarantees that any object that matches the view query will eventually be stored on the device named in the view.

We envision views serving as the connection between management tools and the storage infrastructure. Users can set policies through management tools, such as the one described in Figure 1, from any device in the system at any time. Tools implement these changes by manipulating views, and the underlying infrastructure (Perspective) in turn enforces those policies by keeping files in sync among the devices according to the views. Views provide a clear division point between tools that allow users to manage data replicas and the underlying file system that implements the policies.

A primary contribution of Perspective is the use of semantic queries to *manage the replication of data*. Specifically, it allows the system to provide accessibility and reliability guarantees over semantic, partially replicated data. This builds on previous semantic systems that used queries to *locate* data and hierarchies to manage data.

View-based management enables the design points outlined above. Views provide a primitive allowing users to specify meaningful rule-based placement policies. Because views are semantic, they unify the naming used for data access and data management. Views are also defined in a human-understandable fashion, providing a basis for transparent automation. Perspective provides data reliability using views without restricting their flexibility, allowing it to use a single replica class.

The Perspective prototype is implemented in C++ and runs at user-level using FUSE to connect with the system. It currently runs on both Linux and Macintosh OS X. Perspective stores file data in files in a repository on the machine's local file system and metadata in a SQLite database with an XML wrapper.

A user study evaluation using this prototype shows that, by supporting semantic management, Perspective can simplify important management tasks for end users. View-based management allowed up to six times as many users to complete management tasks correctly than traditional hierarchical systems did [4].

PLACING FILE REPLICAS

In Perspective, the views control the distribution of data among the devices in the system. When a file is created or updated, Perspective checks the attributes of the file against the current list of views in the system and sends an update message to each device with a view that contains that file. Each device can then independently pull a copy of the update.

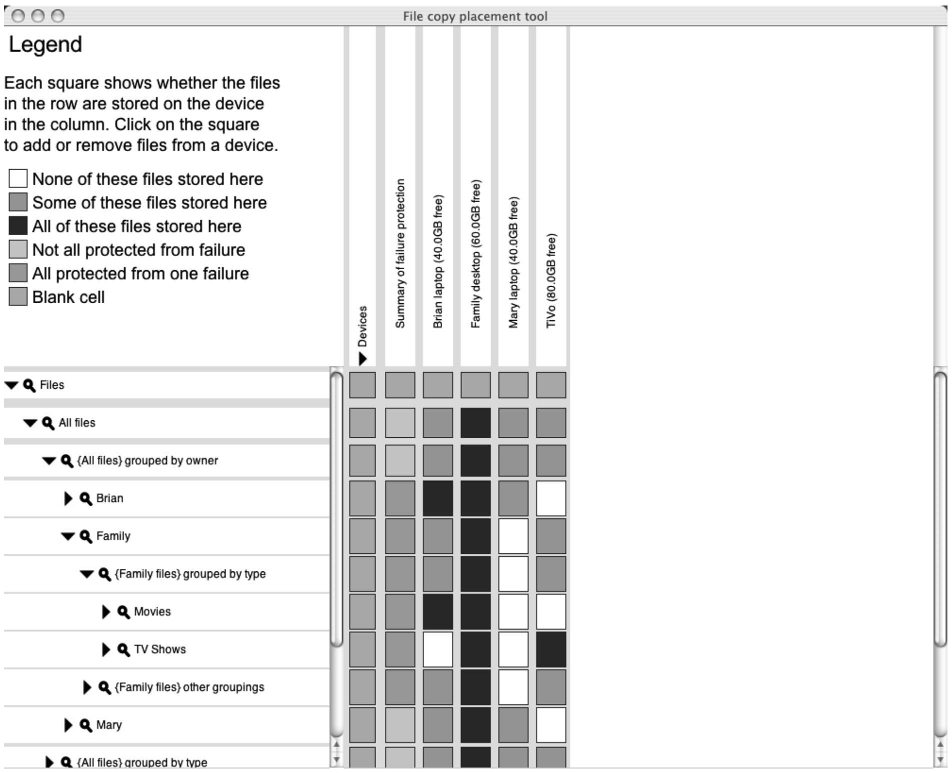
When a device, A, receives an update message from another device, B, it checks that the updated file does, indeed, match one or more views that A has registered. If the file does match, then A applies the update from B. If there is no match, which can occur if the attributes of a file are updated such that it is no longer covered by a view, then A ensures that there is no replica of the file stored locally.

This simple protocol automatically places new files, and also keeps current files up to date according to the current views in the system. Perspective's protocols ensure that this property holds in the face of disconnection, device addition, and device failure, without requiring any centralized control. Perspective's protocols also ensure that files and updates are never lost due to view changes [4].

Each device is represented by a file in the file system that describes the device and its characteristics. Views themselves are also represented by files. Each device registers a view for all device and view files to ensure they are replicated on all participating devices. This allows applications to manage views through the standard file system interfaces, even if not all devices are currently present.

VIEW-BASED DATA MANAGEMENT

In this subsection, we present three scenarios to illustrate view-based management. Each scenario assumes an interface that allows users to manipulate views. While we envision systems containing a number of tools and interfaces, Figure 1 shows the interface we currently provide Perspective users.


File copy placement tool

Legend

Each square shows whether the files in the row are stored on the device in the column. Click on the square to add or remove files from a device.

☐ None of these files stored here
▨ Some of these files stored here
■ All of these files stored here
▨ Not all protected from failure
▨ All protected from one failure
▨ Blank cell

Devices
Summary of failure protection
Brian laptop (40.0GB free)
Family desktop (60.0GB free)
Mary laptop (40.0GB free)
TiVo (80.0GB free)
Files
All files
{All files} grouped by owner
Brian
Family
{Family files} grouped by type
Movies
TV Shows
{Family files} other groupings
Mary
{All files} grouped by type

FIGURE 1: A SCREEN SHOT OF THE VIEW MANAGER GUI. ON THE LEFT ARE FILES, GROUPED USING FACETED METADATA. ACROSS THE TOP ARE DEVICES. EACH SQUARE SHOWS WHETHER THE FILES IN THE ROW ARE STORED ON THE DEVICE IN THE COLUMN.

Traveling: Harry is visiting Sally at her house and would like to play a new U2 album for her. Before leaving, he checks the views defined on his wireless music player and notices that the songs are not stored on the device, although he can play them from his laptop, where they are currently stored. He asks the music player to pull a copy of all U2 songs, which the player does by creating a new view for this data. When the synchronization is complete, the file system marks the view as complete, and the music player informs Harry.

He takes the music player over to Sally's house. Because the views on his music player are defined only for his household, and the views on Sally's devices for her household, no files are synchronized. But queries for "all music" initiated from Sally's digital stereo can see the music files on Harry's music player, so while he is visiting they can listen to the new U2 album from Harry's music player on Sally's nice stereo speakers.

Crash: Mike's young nephew Oliver accidentally pushes the family desktop off the desk onto the floor and breaks it. Mike and his wife Carol have each configured the system to store their files both on their respective laptops and on the desktop, so their data is safe. When they set up the replacement computer, a setup tool pulls the device objects and views from other household devices. The setup tool gives them the option to replace an old device with this computer, and they choose the old desktop from the list of devices. The tool then creates views on the device that match the views on the old desktop and deletes the device object for the old computer. The data from Mike and Carol's laptops is transferred to the new desktop in the background over the weekend.

Short on space: Marge is trying to finish a project for work on her home laptop. While she is working, a capacity automation tool on her laptop alerts her that the laptop is short on space. It recommends that files created over two years ago be moved to the family desktop, which has spare space. Marge, who is busy with her project, decides to allow the capacity tool to make the change. She later decides to keep her older files on the external hard drive instead, and makes the change using a view-editing interface on the desktop.

Conclusion

Home users struggle with replica management tasks that are normally handled by professional administrators in other environments. Perspective provides distributed storage for the home with a new approach to data location management: the view. Views simplify replica management tasks for home storage users, allowing them to use the same attribute-based naming style for such tasks as for their regular data navigation.

ACKNOWLEDGMENTS

We thank Rob Reeder, Jay Melican, and Jay Hasbrouck for helping with the users studies. We also thank the members and companies of the PDL Consortium (including APC, Cisco, DataDomain, EMC, Facebook, Google, HP, Hitachi, IBM, Intel, LSI, Microsoft, NetApp, Oracle, Seagate, Sun, Symantec, and VMware) for their interest, insights, feedback, and support. This material is based on research sponsored in part by the National Science Foundation, via grants #CNS-0326453 and #CNS-0831407, and by the Army Research Office, under agreement number DAAD19-02-1-0389. Brandon Salmon is supported in part by an Intel Fellowship.

REFERENCES

[1] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, and James W. O'Toole Jr., "Semantic File Systems," *Operating Systems Review* 25(5): 16–25, 1991: http://reference.kfupm.edu.sa/content/s/e/semantic_file_systems__50646.pdf.

[2] Daniel Peek and Jason Flinn, "EnsemBlue: Integrating Distributed Storage and Consumer Electronics," *Symposium on Operating Systems Design and*

Implementation (OSDI), USENIX Association, 2006: http://www.cs.ubc.ca/labs/dsg/Sem_Winter_2007/ensemblue.pdf.

[3] Brandon Salmon, Frank Hady, and Jay Melican, "Learning to Share: A Study of Sharing Among Home Storage Devices," Technical Report CMU-PDL-07-107, Carnegie Mellon University, October 2007: http://www.pdl.cmu.edu/PDL-FTP/Storage/CMU-PDL-07-107.pdf.

[4] Brandon Salmon, Steven W. Schlosser, Lorrie Faith Cranor, and Gregory R. Ganger, "Perspective: Semantic Data Management for the Home," *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, USENIX Association, 2009: http://www.usenix.org/events/fast09/tech/full_papers/salmon/salmon.pdf.

[5] M. Satyanarayanan, "The Evolution of Coda," *ACM Transactions on Computer Systems*, 20(2): 85–124, May 2002: http://www.cs.cmu.edu/~satya/docdir/p85-satyanarayanan.pdf.

TASNEEM G. BRUTCH

migration to multicore: tools that can help

Tasneem Brutch is a Senior Staff Engineer at Samsung Research in San Jose, CA. She holds a BS in Computer Science and Engineering, a Master's in Computer Science, and a PhD in Computer Engineering from Texas A&M University. She was awarded a USENIX scholarship for her Ph.D. research. She has 12 years of industry experience, working as Senior Engineer and Architect at Hewlett-Packard and Intel.

t.brutch@samsung.com

THE ADVENT OF MANYCORE SYSTEMS requires that programmers understand how to design, write, and debug parallel programs effectively. Writing and debugging parallel programs has never been easy, but there are many tools that can help with this process. In this article I provide a survey of useful tools and resources for multithreaded applications.

Multiple threads are said to execute concurrently when they are interleaved on a single hardware resource, which limits the overall maximum performance gains from threading. When multi-threaded applications run simultaneously on different hardware, threads in an application are said to execute in parallel. To achieve software parallelism, hardware must be able to support simultaneous and independent execution of threads [1].

Performance gains through parallelism are proportional to effective partitioning of software workloads across available resources while minimizing inter-component dependencies. Performance is impacted by issues such as communication overhead, synchronization among threads, load balancing, and scalability as the number of cores changes.

It is recommended that performance bottlenecks which impact both serial and parallel applications be removed prior to parallelizing an application. This includes optimizing existing serial applications for the multicore memory hierarchy prior to parallelization.

A number of tools can be used to assist with the migration of sequential applications to multicore platforms. This article focuses on tools for C and C++ programming languages in Windows and Linux environments. Most of the tools noted here are open source or built on top of open source tools. The discussion is intended to be a starting point and is not comprehensive of all available tools. Figure 1 provides a high-level view of various categories of tools and the workflow between them [2]. Tool categories identified in the figure are discussed in this article.

Threading APIs

First, I include a brief discussion of threading APIs, as the choice of APIs may affect the selection of tools. A number of open source multi-threading programming APIs are available for both shared memory and distributed memory systems.

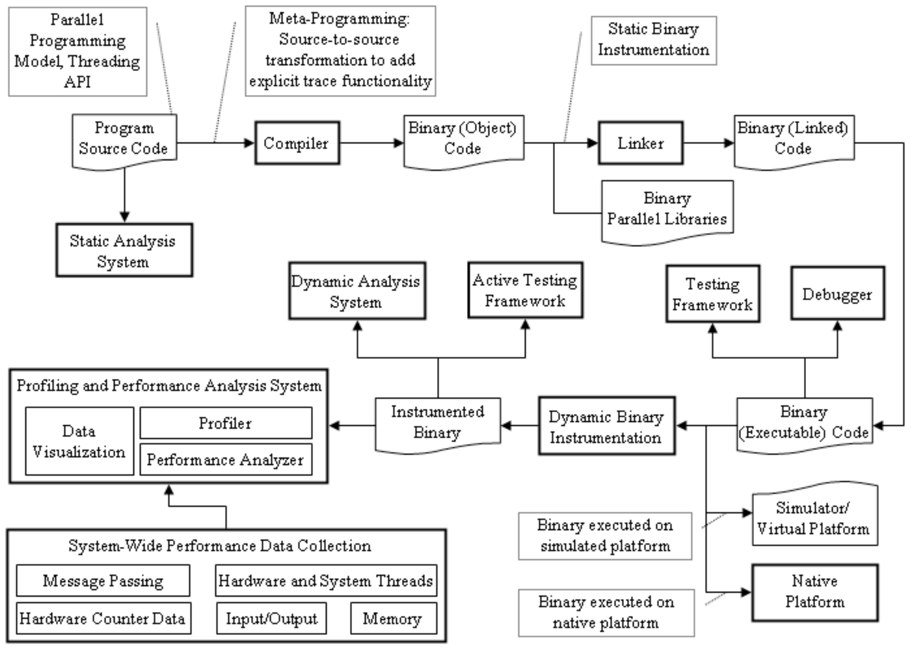

Parallel Programming Model, Threading API
Meta-Programming: Source-to-source transformation to add explicit trace functionality
Static Binary Instrumentation
Program Source Code
Compiler
Binary (Object) Code
Linker
Binary (Linked) Code
Binary Parallel Libraries
Static Analysis System
Dynamic Analysis System
Active Testing Framework
Testing Framework
Debugger
Profiling and Performance Analysis System
Data Visualization
Profiler
Performance Analyzer
Instrumented Binary
Dynamic Binary Instrumentation
Binary (Executable) Code
Simulator/ Virtual Platform
Binary executed on simulated platform
System-Wide Performance Data Collection
Message Passing
Hardware and System Threads
Hardware Counter Data
Input/Output
Memory
Binary executed on native platform
Native Platform

FIGURE 1: CATEGORIES OF TOOLS AND THE WORKFLOW BETWEEN THEM

MULTI-THREADING APIS FOR SHARED MEMORY SYSTEMS

OpenMP (Open Multi-Processing) is a multi-threading API, which consists of a set of compiler directives, library routines, and runtime environment variables, and is available for C, C++, and Fortran. Data may be labeled as either *shared* or *private*. The OpenMP memory model allows all threads to access globally shared memory. Private data is only accessible by the owning thread. Synchronization is mostly implicit, and data transfer is transparent to the programmer. OpenMP employs a fork-join execution model and requires an OpenMP-compatible compiler and thread-safe library runtime routines [3], [4].

Pthreads (POSIX Threads) is defined in the ANSI/IEEE POSIX 1003.1-1995 standard. It is a set of C language programming types and procedure calls which do not require special compiler support. The header file pthread.h needs to be included. Pthreads uses a shared memory model, that is, the same address space is shared by all threads in a process, making inter-thread communication very efficient. Each thread also has its own private data. Programmers are responsible for synchronizing access to globally shared data. Pthreads is now the standard interface for Linux, and pthreads-win32 is available for Windows [5].

GNU Pth (GNU Portable Threads) is a less commonly used POSIX/ANSI-C–based library. It uses non-preemptive, priority-based scheduling for multi-threading in event-based applications. All threads execute in the server application's common address space. Each thread has its own program counter, signal mask, runtime stack, and errno variable. Threads can wait on events such as asynchronous signals, elapsed timers, pending I/O on file descriptors, pending I/O on message ports, customized callback functions, and thread and process termination. A Pthreads emulation API is also optionally available [6].

Threading Building Blocks (TBB) is a C++ template library that consists of data structures and algorithms for accessing multiple processors. Operations are treated as tasks, by specifying threading functionality in terms of logical tasks, as opposed to physical threads. TBB emphasizes data parallel programming [7].

MULTI-THREADING ON DISTRIBUTED MEMORY SYSTEMS

Message Passing Interface (MPI) is a library specification for message passing on massively parallel machines and workstation clusters which supports point-to-point and collective communication. Operations in MPI are expressed as functions. The MPI standard originally targeted distributed memory systems, but now MPI implementations for SMP/NUMA architectures are also available. The programmer is responsible for identification of parallelism and its implementation using MPI constructs. Objects called “communicators” and “groups” define communication between processes [8] [9].

PLATFORM-SPECIFIC MULTI-THREADING APIS

Open Computing Language (OpenCL) is a C-based framework for pragmas for general-purpose parallel programming across heterogeneous platforms. It is a subset of ISO C99 with language extensions. The specification includes a language for writing kernels and APIs for defining and controlling a platform, and it provides online or offline compilation and build of compute kernel executables. It includes a platform-layer API for hardware abstraction and a runtime API for executing compute kernels and managing resources. It uses task-based and data-based parallelism, and implements a relaxed-consistency, shared memory model [10].

Compilers and Compiler-Based Instrumentation

A number of C and C++ *compilers* are available to programmers for compiling applications using OpenMP and Pthread APIs. Information about selection of appropriate options for OpenMP and Pthreads and inclusion of appropriate include files can be obtained from their documentation.

Figure 1 illustrates various stages of compiler-based instrumentation. Code may be modified by the compiler for generating trace information. Instrumentation may be source-to-source, static binary, or dynamic. *Source-to-source instrumentation* modifies source code prior to pre-processing and compilation. In *static binary instrumentation* the compiled binary code is modified prior to execution [11].

Static Code Analyzers

Static code analyzers help detect issues beyond the limits of runtime coverage which may not have been reachable by functional test coverage. Static code analysis is done on the source code without executing the application, requiring any instrumentation of the code, or developing test cases. Potential errors are detected by modeling software applications using the source code. These models can be analyzed for behavioral characteristics. Static analysis exhaustively explores all execution paths, inclusive of all data ranges, to ensure correctness properties, such as absence of deadlock and livelock. Static analyzers cannot model absolute (wall-clock) time but can model relative time and temporal ordering. A directed control flow graph is developed, built on the program’s syntax tree. The constraints associated with variables are assigned to the nodes of the tree. Nodes represent program points, and the flow of control is represented by edges. Typical errors detected by using analysis based on a control flow graph include: illegal number or type of arguments, non-terminating loops, inaccessible code, uninitialized variables and pointers, out-of-bound array indices, and illegal pointer access to a variable or structure. Due to the large number of possible interleavings in a multi-threaded application, model checking is computationally expensive and limited in its applicability [11].

The use of static code analyzers helps maintain code quality. Their integration in the build process is recommended to help identify potential issues earlier on in the development process.

Berkeley Lazy Abstraction Verification Tool (BLAST) is a software model checker for C programs. It is used to check that software satisfies the behavioral properties of its interface. It constructs an abstract model, which is model checked for safety properties. Given a temporal safety property for a C program, BLAST attempts to statically prove that it either satisfies the safety property or demonstrates an execution path that is in violation of the property. It uses lazy predicate abstraction and interpolation-based predicate discovery to construct, explore, and refine abstractions of the program state space. BLAST is platform independent and has been tested on Intel x86/Linux and Intel x86/Microsoft Windows with Cygwin. BLAST was released under the Modified BSD license [12].

Debuggers

Enhanced complexity of multi-threaded applications results from a number of factors, such as non-deterministic thread scheduling and preemption, and dependencies between control flow and data [11]. Non-deterministic execution of multiple instruction streams from runtime thread scheduling and context switching generally stems from the operating system's scheduler. The use of debuggers themselves may mask issues caused by thread interactions, such as deadlocks and race conditions. Factors such as thread priority, processor affinity, thread execution state, and starvation time can affect the resources and execution of threads.

A number of approaches are available for debugging concurrent systems, including traditional debugging, and event-based debugging. *Traditional debugging*, or breakpoint debugging, has been applied to parallel programs, where one sequential debugger per parallel process is used. These debuggers can provide only limited information when several processes interact. *Event-based* or *monitoring debuggers* provide some replay functionality for multi-threaded applications, but can result in high overhead. Debuggers may display control flow, using several approaches, such as textual presentation of data, time process diagrams, or animation of program execution [13].

The use of a threading API may impact the selection of a debugger. Using OpenMP, for example, requires the use of an OpenMP-aware debugger, which can access information such as constructs and types of OpenMP variables (private, shared, thread private) after threaded code generation.

Dynamic Binary Instrumentation (DBI)

Dynamic binary instrumentation analyzes the runtime behavior of a binary application by injecting instrumentation code which executes as part of the application instruction stream. It is used to gain insight into application behavior during execution. As opposed to static binary analysis, which exhaustively exercises all code paths, DBI explores only executed code paths. DBIs may be classified as either lightweight or heavyweight. A *lightweight DBI* uses architecture-specific instruction stream and state, while a *heavyweight*

DBI utilizes an abstraction of the instruction stream and state. Lightweight DBIs are not as portable across architectures as heavyweight DBIs. Valgrind [22], which is discussed later, is an example of a heavyweight DBI, and Pin [24], also discussed in this article, is an example of a lightweight DBI.

Profiling and Performance Analysis

Profilers are useful for both single- and multi-threaded applications. They facilitate optimization of program decomposition and efficient utilization of system resources by inspecting the behavior of a running program and helping to detect and prevent issues that can impact performance and execution. Issues encountered in multi-threaded applications include:

- Large number of threads, leading to increased overhead from thread startup and termination [1].
- Overhead from concurrent threads exceeding the number of hardware resources available [1].
- Contention for cache usage resulting from the large number of concurrent threads attempting to use the cache [1].
- Contention for memory use among threads for their respective stack and private data structure use [1].
- Thread convoying, whereby multiple software threads wait to acquire a lock [1].
- Data races occurring when two concurrent threads perform conflicting accesses and no explicit mechanism is implemented to prevent accesses from being simultaneous [14].
- Locking hierarchies causing deadlocks, which result in all threads being blocked and each thread waiting on an action by another thread [11].
- Livelocks (similar to deadlocks except that the processes/threads involved constantly change with respect to one another, with neither one being able to progress) can occur with some algorithms, where all processes/threads detect and try to recover from a deadlock, repeatedly triggering the deadlock detection algorithm [1].

Approaches to profiling can be identified as either *active* or *passive*. Compiler-based probe insertion is an example of active profiling, where execution behavior is recorded using callbacks to the trace collection engine. In passive profiling, control flow and execution state are inspected using external entities, such as a probe or a modified runtime environment. Passive profiling may require specialized tracing hardware and, in general, does not require modification of the measured system [11].

Data may be gathered by a profiler using a number of methods. *Event-based profilers* utilize sampling based on the occurrence of processor events. *Statistical profilers* use sampling to look into the program counter at regular intervals, using operating system interrupts. *Instrumenting profilers* insert additional instructions into the application to collect information. On some platforms, instrumentation may be supported in hardware using a machine instruction. *Simulator* or *hypervisor-based data collection* selectively collects data by running the application under an instruction set simulator or hypervisor.

Profilers may also be classified based on their output. *Flat profilers* show average call times, with no associated callee or context information. *Call-graph profilers* show call times, function frequencies, and call chains.

Profilers can provide behavioral data only for control paths that are actually executed. Execution of all relevant paths requires multiple runs of the application, with good code coverage. Code coverage can be improved using carefully selected input data and artificial fault injection. Fine-grained behavioral data from a running system can be coupled with offline analysis.

Profilers may not be portable across architectures, as they may require special hardware support. Others may focus only on user-space applications. A profiler may be designed to focus on analyzing the utilization of one or more system resources, such as call stack sampling, thread profiling, cache profiling, memory profiling, and heap profiling. Profilers can include aspects of absolute (wall-clock) time in their analysis [11].

OProfile is a profiling and performance monitoring tool for Linux on a number of architectures, including x86, AMD Athlon, AMD64, and ARM. It provides system-wide profiling, with a typical overhead of 1% to 8%, and includes a number of utilities. It consists of a kernel driver and a daemon for collecting sample data. OProfile uses CPU hardware performance counters for system-wide profiling, which includes hardware and software interrupt handlers, kernel and kernel modules, shared libraries, and applications [15].

DTrace is a dynamic tracing framework created by Sun Microsystems. It is now available for a number of operating systems, including Linux. It can be used to get an overview of the running system and is used for tuning and troubleshooting kernel and application issues on production systems, in real time. It allows dynamic modification of the OS kernel and user processes to record additional data from locations of interest using "probes." A probe is a location or activity with which DTrace can bind a request to perform a set of actions: for example, the recording of a stack trace. The source code for this tool has been released under the Common Development and Distribution License (CDDL) [16].

GNU Linux Trace Toolkit next generation (LTTng) is a static and dynamic tracer that supports C and C++ on Linux (and any language that can call C). It is supported on x86, PowerPC 32/64, ARMv7, OMAP3, sparc64, and s390. LTTng is available as a kernel patch, along with a tool chain (ltt-control), which looks at process blocking, context switches, and execution time. It can be used for performance analysis on parallel and real-time systems. LTTV is a viewing and analysis program designed to handle huge traces. Tracers record large amounts of events in a system, generally at a much lower level than logging, and are generally designed to handle large amounts of data [17].

CodeAnalyst by AMD is a source code profiler for x86-based platforms with AMD microprocessors that is available for Linux and Windows environments. It has been built on top of the OProfile Linux tool for data collection, and provides graphical and command line interfaces for code profiling, including time-based and event-based profiling, thread analysis, and pipeline simulation [18].

Data Visualization

Visualization of profile data facilitates the comprehensibility of data and enhances its usability. A number of tools provide a standard interface for visualization of different types. *Gnuplot* is a portable, command-line–driven, interactive data and function plotting utility. It is copyrighted but can be freely distributed [19]. *Graphviz* is open source graph visualization software, which can be used to represent structural information as diagrams of abstract graphs and networks [20].

Dynamic Program Analysis

Dynamic program analysis is done by executing programs built either on actual hardware or on a virtual processor. Dynamic analysis checks program properties at runtime, and it generally identifies the problem source much faster than extensive stress testing does. Issues can be detected much more precisely, using code instrumentation and analysis of memory operations. These tools are generally easy to automate, with a low rate of false positives. For dynamic testing to be effective the test input has to be selected to exercise proper code coverage.

Valgrind is an instrumentation framework for building dynamic analysis tools. It is available for x86/Linux, AMD64/Linux, PPC32/Linux, and PPC64/Linux. Work on versions of Valgrind for x86/Mac OS X and AMD64/Mac OS X is currently underway. The Valgrind framework is divided into three main areas: core and guest maintenance (coregrind), translation and instrumentation (LibVex), and user instrumentation libraries [21]. Valgrind tools are used for detecting memory management and threading issues, and for application profiling [22]. Helgrind, Memcheck, Cachegrind, and Massif are some of the tools included in Valgrind's tool suite:

Helgrind is a thread debugger to detect data races in multi-threaded applications. It detects memory locations accessed by multiple Pthreads that are lacking consistent synchronization.

Memcheck is used to detect memory management–related issues for C and C++.

Cachegrind provides cache profiling and simulation of L1, D1, and L2 caches. **Callgrind** extends Cachegrind to provide visualization information about callgraphs.

Massif performs detailed heap profiling by taking regular snapshots of a program's heap, to help identify parts of the program contributing to most memory allocations.

DynInst allows dynamic insertion of code in a running program. It uses dynamic instrumentation to allow modification of programs during execution, without re-compilation, re-linking, and re-execution. DynInst was released by the Paradyn Parallel Tools Project and has been used by applications such as performance measurement tools, correctness debuggers, execution drive simulations, and computational steering. The recent release of DynInst supports PowerPC (AIX), SPARC (Solaris), x86 (Linux), x86 (Windows), and ia64 (Linux) [23].

Pin from Intel is a framework for building program analysis tools using dynamic instrumentation. It is an example of dynamic compilation targeting a VM which uses the same ISA as the underlying host [11]. It is an open source tool and does runtime binary instrumentation of Linux applications, whereby arbitrary C/C++ code can be injected at arbitrary places in the executable. Pin APIs allow context information, such as register contents, to be passed to the injected code as parameters. Any registers overwritten by the injected code are restored by Pin. It also relocates registers, in-lines instrumentation, and caches previously modified code to improve performance. The Pin architecture consists of a virtual machine (VM), a code cache, and an instrumentation API, which can be invoked by custom plugin utilities called Pintools. The VM consists of a Just-in-Time (JIT) compiler, an emulation unit, and a dispatcher. Instructions, such as system calls, which cannot be executed directly are intercepted by the emulator. The dispatcher checks for the next code region in the code cache. If it is not present in the code cache, it is generated by the JIT compiler [24], [25], [26].

Active Testing

Active testing consists of two phases. Static and dynamic code analyzers are first used to identify concurrency-related issues, such as atomicity violations, data races, and deadlock. This information is then provided as input to the scheduler to minimize false positives from the concurrency issues identified during the static and dynamic code analysis. The tool CalFuzzer uses this approach.

CalFuzzer provides an extensible active testing framework for implementing predictive dynamic analysis to identify program statements with potential concurrency issues, and it allows implementation of custom schedulers, called *active checkers*, for active testing of concurrency-related issues [27].

System-Wide Performance Data Collection

In addition to problem partitioning and load balancing, programmers need access to systemwide resource usage data, as well as the ability to relate it to application performance. Availability of standardized APIs can facilitate access to such low-level system data by profilers and performance analyzers. PAPI is one such attempt at an API for accessing hardware performance counters.

The Performance Application Programming Interface (PAPI) project at the University of Tennessee defines an API for accessing hardware performance counters, which exist as a small set of registers for counting events. Monitoring the processor-performance counters enables correlation between application code and its mapping to the underlying hardware architecture and is used in performance analysis, modeling, and tuning. Tools that use PAPI include PerlSuite, HPCToolkit, and VProf. PAPI is available for a number of environments and platforms, including Linux, on SiCortex, Cell, AMD Athlon/Opteron, Intel Pentium, Itanium, Core 2, and, for Solaris, UltraSparc [28].

Conclusion

The increased complexity of multi-threaded parallel programming on multicore platforms requires more visibility into program behavior and necessitates the use of tools that can support programmers in migrating existing sequential applications to multicore platforms. This article presents a survey of different categories of tools, their characteristics, and the workflow between them. Most of the tools discussed are open source, or built on top of open source tools, for C and C++.

ACKNOWLEDGMENTS

I would like to thank Sungdo Moon from Samsung and *;login:* editor Rik Farrow for their review of the article and valuable suggestions.

REFERENCES

[1] Shameem Akhter, Jason Roberts, *Multi-Core Programming: Increasing Performance through Software Multithreading* (Intel Press, 2006).

[2] H. Wen, S. Sbaraglia, S. Saleem, I. Chung, G. Cong, D. Klepacki, "A Productivity Centered Tools Framework for Application Performance Tuning," *Proceedings of the Fourth International Conference on the Quantitative Evaluation of Systems*, Sept. 2007.

[3] "OpenMP API Specification for Parallel Programming": http://openmp.org/wp/.

[4] "GNU OpenMP (GOMP) Project": http://gcc.gnu.org/projects/gomp/.

[5] Lawrence Livermore National Lab, "POSIX Threads Programming": https://computing.llnl.gov/tutorials/pthreads/.

[6] Ralf S. Engelschall, "GNU Portable Threads," June 2006: http://www.gnu.org/software/pth/.

[7] Intel, "Threading Building Blocks 2.2 for Open Source": http://www.threadingbuildingblocks.org/.

[8] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard, version 2.1," June 2008: http://www.mpi-forum.org/docs/mpi21-report.pdf.

[9] Message Passing Interface (MPI) standard Web site: http://www.mcs.anl.gov/research/projects/mpi/.

[10] Khronos Group, "Open-CL, the Open Standard for Parallel Programming of Heterogeneous Systems": http://www.khronos.org/opencl/.

[11] Daniel G. Waddington, Nilabja Roy, Douglas C. Schmidt, "Dynamic Analysis and Profiling of Multi-Threaded Systems," 2007: http://www.cs.wustl.edu/~schmidt/PDF/DSIS_Chapter_Waddington.pdf.

[12] Thomas A. Henzinger, Dirk Beyer, Rupak Majumdar, Ranjit Jhala, "BLAST: Berkeley Lazy Abstraction Software Verification Tool": http://mtc.epfl.ch/software-tools/blast/.

[13] Yusen Li, Feng Wang, Gang Wang, Xiaoguang Liu, Jing Liu, "MKtrace: An Innovative Debugging Tool for Multi-Threaded Programs on Multiprocessor Systems," *Proceedings of the 14th Asia Pacific Software Engineering Conference*, 2007.

[14] Liqiang Wang, Scott D. Stoller, "Runtime Analysis of Atomicity for Multithreaded Programs," *Proceedings of the IEEE Transactions on Software Engineering*, Feb. 2006.

[15] OProfile Web site: http://oprofile.sourceforge.net/about/.

[16] Sun Microsystems, "Solaris Dynamic Tracing Guide," Dec. 2007: http://wikis.sun.com/display/DTrace/Documentation.

[17] GNU Linux Trace Toolkit next generation (LTTng) Project Web site: http://ltt.polymtl.ca/.

[18] AMD, "CodeAnalyst Performance Analyzer": http://developer.amd.com/CPU/CODEANALYST/Pages/default.aspx.

[19] Gnuplot Web site: http://www.gnuplot.info/.

[20] Graphviz graph visualization software Web site: http://www.graphviz.org/.

[21] Daniel Robson, Peter Strazdins, "Parallelization of the Valgrind Dynamic Binary Instrumentation Framework," *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications*, 2008.

[22] Valgrind Web site: http://valgrind.org/.

[23] DynInst by the Paradyn Parallel Tools Project: http://www.dyninst.org/.

[24] Pin Web site: http://www.pintool.org/.

[25] "Pin: A Framework for Building Program Analysis Tools using Dynamic Instrumentation": http://www.cc.gatech.edu/~ntclark/8803f08/notes/pin.pdf.

[26] Steven Wallace, Kim Hazelwood, "SuperPin: Parallelizing Dynamic Instrumentation for Real-Time Performance," *Proceedings of the International Symposium on Code Generation and Optimization*, 2007.

[27] Pallavi Joshi, Mayur Naik, Chang-Seo Park, Koushik Sen, "CalFuzzer: An Extensible Active Testing Framework for Concurrent Programs," *Proceedings of the 21st International Conference on Computer Aided Verification*, 2009: http://berkeley.intel-research.net/mnaik/pubs/cav09/paper.pdf.

[28] Performance Application Programming Interface (PAPI) Web site: http://icl.cs.utk.edu/papi/.

RUDI VAN DRUNEN

chips and static electricity

Rudi Van Drunen is a senior UNIX systems consultant with Competa IT B.V. in The Netherlands. He also has his own consulting company, Xlexit Technology, doing low-level hardware-oriented jobs.

rudi-usenix@xlexit.com

THERE IS ALWAYS A WHOLE LOT OF discussion about the effects and dangers of static electricity for electronics. In this article I will try to explain the issues involved with static electricity, the myths connected to the phenomenon, the dangers for your systems, and how to avoid them.

Static electricity has been known for a very long time. In the seventeenth century, people began to build machines to generate static electricity artificially. Benjamin Franklin demonstrated the effect of static electricity with his famous kite experiment in a lightning storm. As lightning is a form of electrostatic discharge on a very large scale, you can imagine what discharges on a small scale can do to the sub-micron–sized patterns on a chip. To explain just this effect, I first have to describe the way chips (i.e., integrated circuits) are built and then describe the properties of electrostatic discharge (ESD) and their impact on the chips. The last part of this article describes the measures you can take to minimize the damage to your hardware due to ESD.

Chips

A chip, or, better, an integrated circuit, is a device built out of transistors that are created in different forms of silicon on a silicon substrate (a wafer). The different forms of silicon are the N type, with an extra electron in the shell around the molecule, and the P type, where the shell is short one electron.

A transistor is an electronic element that can actively control the flow of current. A simple transistor has three terminals: an input, an output, and a control terminal. If you apply a voltage to the control terminal, you turn on or off the “switch” between the input and the output of the transistor. The current flowing is many times larger than the current you need to control the switch/gate. That is why it is said that transistors can amplify current.

There are a number of different transistors. The current amplification type (that actually needs a current on the control terminal) is called a bipolar transistor, whereas others that just need a static voltage (an electric field) to switch current on or off are called field effect transistors. Field effect transistors are commonly used to build digital logic integrated circuits, as found in your computers.

A field effect transistor (FET) is created on a chip by virtually building a channel in a well of P-type silicon in the substrate. This channel is the “conductor" for the current (electrons) that will flow through when switched on. Then a very thin insulating layer on top of the channel separates the gate or control electrode from the current channel. When voltage is applied to that electrode, an electric field is formed within the channel, preventing the current from flowing through (see Figure 1).

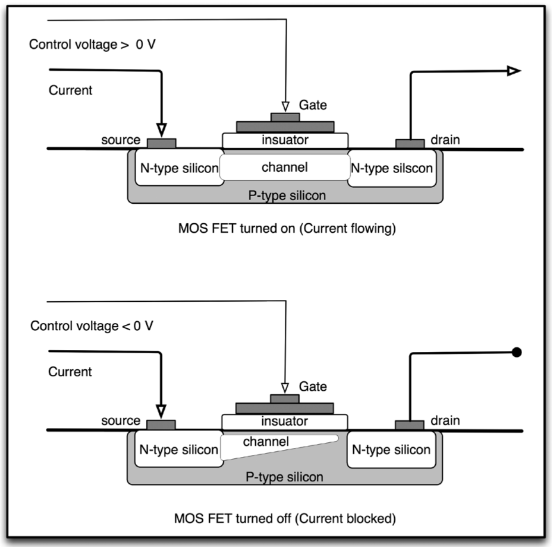


FIGURE 1: OPERATION OF A MOSFET TRANSISTOR

The type of insulation between the gate and the channel is related to the name of the device. Most current logic chips are fabricated using a MOSFET technology (Metal Oxide Semiconductor Field Effect Transistor).

Typically, a transistor on a chip takes about 3 units of space, where a unit of space is related to the technology the chip manufacturer uses. State-of-the-art manufacturers are now able to make 15 nm patterns, resulting in a transistor size of 45 nm (1 nm equals approximately 0.000000004 inch).

The transistors are interconnected by conductors in different layers, often made out of metal (see Figure 2). These transistor circuits form basic logic elements that build functional blocks.

Creating those patterns on a piece of silicon is a very complex lithographic and chemical process which takes place in multi-million-dollar clean rooms and takes a complete article to describe.

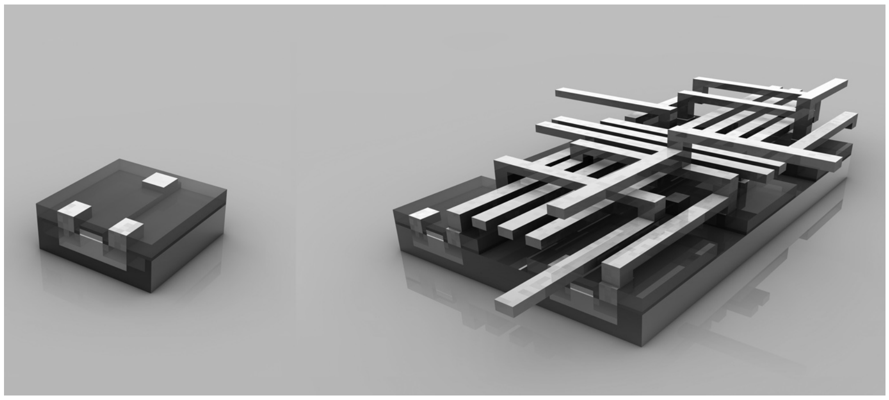


FIGURE 2: SCHEMATIC DRAWING OF A SINGLE TRANSISTOR ELEMENT AND INTERCONNECTS BETWEEN THE ELEMENTS (SOURCE: INTEL)

A logic chip is composed of functional blocks (counters, adders, etc.). These blocks can be described by functions that then are implemented on the chip level by a number of transistors. A modern, high-end CPU contains about 2,300,000,000 transistors (an 8-core Nehalem-EX server processor from Intel). Figure 3 shows an example of the patterns on a chip.

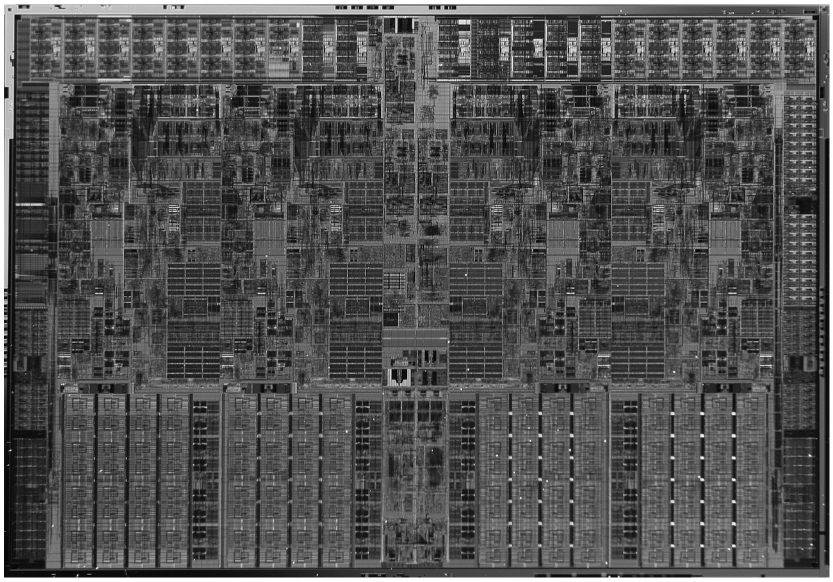


FIGURE 3. A DIE PHOTOGRAPH OF A 4-CORE NEHALEM PROCESSOR (SOURCE: INTEL)

Design of a Chip

Nowadays, the design process of a digital integrated circuit is comparable to writing a piece of software. The functional requirements are implemented using a hardware design language such as VHDL or VeriLog. The VHDL code then is simulated to verify functionality. Once approved, the code is run through a silicon compiler that generates the actual chip artwork for the lithographic process.

To verify the functionality on the physical hardware level, there are chips containing programmable arrays of logic blocs, where the VHDL program is used to make interconnects between standard blocks. These chips (field-programmable gate arrays [FPGAs]) can be used as an intermediary (verify, prototype) stage before the actual production of custom silicon. Although far less dense than full custom chips, these devices can be the solution for smaller series or prototyping.

As you might imagine, those tiny traces on a chip and tiny insulating layers are extremely fragile and susceptible to overvoltage or overcurrent. Both overvoltage and overcurrent on the chip can be the result of an electrostatic discharge.

Static Electricity: Charging

Static electricity can be thought of as a difference in the electrical charge between two (non-conductive) bodies. This electrical charge is often the result of a friction between two substances. That friction will lead to exchange of loosely coupled electrons from the one substance to the other. One part will become negatively charged (with an excess of electrons), while the other part will become positively charged (with fewer electrons). This process of charging is called triboelectrification and is the most common way that static charges are built up.

In addition, a strong electric field in the neighborhood (e.g., the high voltage of a CRT tube) can separate electrons in a body, and this builds up a charge on components, a process known as inductive charging. The net charge of the complete device will not change, but it now has regions of excess positive and negative charge. Touching these regions with a tool that has a neutral charge will result in an ESD.

The last method of charging a body is conductive charging, where there is physical contact between a charged body and a body with a different potential. The charge will transfer from one body to the other, leaving them both equally electrically charged.

The difference in charge that builds up can be big—many thousands of volts. The charge depends on the materials and the friction between them. For example, lightning, a discharge of static electricity that is built up by friction between clouds, can reach 100,000 volts.

Static Electricity: Discharging

Discharge of electrostatic charges can be achieved by having the charge bleed off to ground. This can occur slowly and with a low current (due to high impedance to ground) or quickly with higher current with a direct contact: for example, you can charge your body by walking over nylon carpet and then discharged by touching a grounded object (e.g., a door knob).

Another form of discharging is corona discharge. This effect occurs only when working with very high electric fields and high voltage charges. The high voltage creates a high electric field around the charged object. When a grounded object enters the field, the difference in charge is so high that a spark occurs on the sharp edges of the object (where the field has the highest density). A spark is triggered when the electric field strength exceeds approximately 4–30kV/cm.

Problems

Static discharge will both apply a very high voltage on a chip and, as the charge is flowing away, impose a high current through the sensitive semiconductor. Due to the high voltage and the high current, problems on the chip can occur. Traces (connections) between transistors can be burned away completely, and partially destroyed traces can eventually lead to failure. These problems will emerge later in the component's life, when traces that are partially cut by the discharge fail completely due to heat because they are thinner now, causing the same current in the chip to heat up much faster in operation. A device can be subjected to a number of weak ESD pulses, with each successive pulse further degrading a device until, finally, there is a catastrophic failure. There is no known practical way to screen for walking-wounded devices. Figures 4 and 5 show some of the serious issues that can occur due to an ESD pulse on a chip.

Latch-up

Another problem that can occur on a chip that is hit by a relatively mild ESD that does no physical harm is what is called "latch-up." This is an effect caused by unwanted structures that are on the chip as a result of other structures. Design and placement of transistors on a chip should avoid this, but often they cannot be avoided, or only at high chip area-cost. These structures form stray semiconductors that are triggered by out-of-bound voltage spikes (not only ESD, but also power supply spikes, especially when turning on the power supply) and can prevent other transistors from working once triggered. Power-cycling the device restores the chip to its default state.

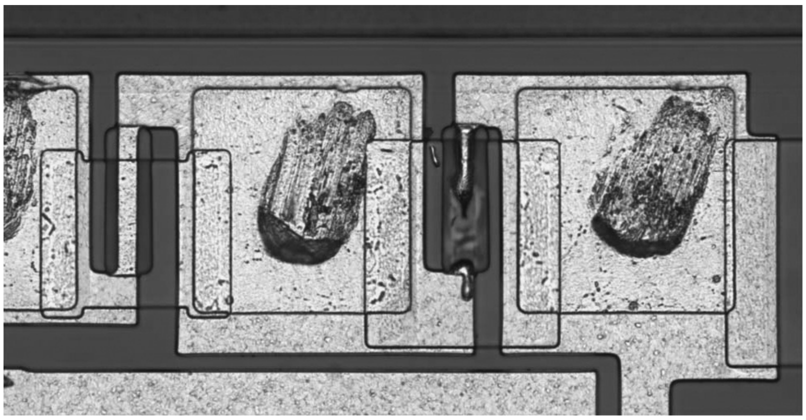


FIGURE 4: INTERCONNECT BURNED AWAY (SOURCE: BUNNIE, HTTP://WWW.BUNNIESTUDIOS.COM/BLOG, WITH PERMISSION)

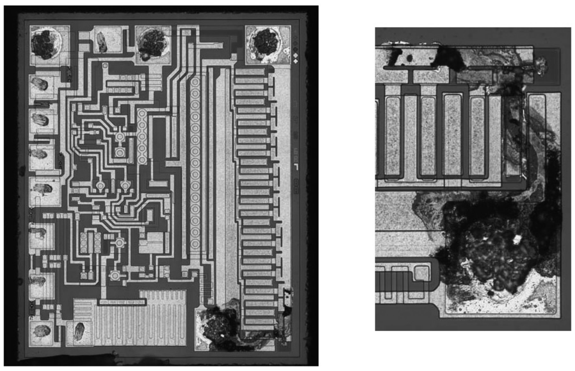


FIGURE 5: SERIOUS DEFORMATION OF CHIP PATTERN THROUGH ESD (SOURCE: BUNNIE, HTTP://WWW.BUNNIESTUDIOS.COM/BLOG, WITH PERMISSION)

How Much?

Often ESD leaves no trace whatsoever. You often do not feel, hear, or see anything, but there is a discharge that may harm your systems. The table below shows the voltage that is related to the ESD vs. the traces it shows.

Feel static discharge	> 3000 volts
Hear static discharge	> 6000 volts
See static discharge (spark)	> 9000 volts

Protection

Almost all chips have diodes and other measures built into their design to protect against static discharges on the leads by bleeding them off, but they cannot protect against all static discharges. And some chips cannot have these protections since the chips' functionality would be affected by them. Examples of this are very low-voltage and low-current chips, radio front ends, and analog or touch devices.

The first measure in ESD protection is to minimize the buildup of static electricity; the less charge there is, the less likely that it will harm your electronics. Often this is done by being careful not to use materials and fabrics in the environment that build up static electricity easily, such as nylon clothing and nylon carpet. Friction between nylon and almost any other material will quickly produce static charge. It would be useful to post warning signs in work areas and rooms where you have your equipment operating. The more conductive the material is, the less static buildup you will face.

Another factor in the buildup of static electricity is relative humidity. This is especially true in winter (or in the summer in desertlike areas). With low humidity comes low air conductance, so static charges are not able to leak to earth and will build up fast, making dangerous electrostatic charges about 10 times more likely to occur than in humid air. Other problems can occur if you run your systems in a data center that has too low relative humidity; the airflow within some systems can cause buildup of static electricity on the components in the machine.

A third measure is trying to let the static charge bleed away to ground or at least have your system on the same charge level as the operator or your tools. This can be accomplished by using a wristband when operating and fixing internals of your systems. The wristband will ease out the potential that your body holds to the potential the machine is on. In the wristband cable there is a resistor to prevent high currents and sparks when connecting the wire.

Practical Tips

The following are practical tips to minimize the damage due to ESD.

- Be aware of clothing, footwear, and carpets. These are by far the most likely cause of charge buildup on your body. There are special conductive (enhanced with carbon particles) garments, shoes, furniture, and carpets. Avoid nylon in all cases. The non-conductive wheels of office chairs can be pretty nasty, too.
- When handling equipment, always touch both the grounded machine and the rack to be sure that all are at the same level.
- Keep your hardware grounded at all times, to prevent buildup of electrostatic charges. You can do that by keeping the chassis grounded by a small wire and alligator clips or keeping the chassis physically connected to the grounded rack.
- Be careful with styrofoam. Do not let pieces of packing foam get close to the printed circuit board. Some manufacturers use styrofoam that has conductive particles in it, which has less static buildup.
- Use grounding wristbands and connect them to the earth/chassis.
- Leave components in their ESD-protective carriers as long as possible. If you need to have them on your desk, sometimes the back of a mouse mat has conductive rubber, so it is wise to leave them on the mat before handling. Also, components that you remove from your machine need to be protected.
- If you need to ship ESD devices, use ESD-protective bags or wrap them in aluminum foil and put them in an antistatic plastic bag. Note that if there is a battery on the board you should not use aluminum foil, as you may short-circuit things.

- If the humidity in the room drops below 30%, use an air humidifier or adjust your air-conditioning systems accordingly.

If something ESD-related happens and you notice that there is a static discharge, either by feeling it or noticing a spark, mark the components and machine that you think might be affected so that you can relate problems with these devices or components later to this event.

Conclusion

Integrated circuits of all sorts are susceptible to ESD. When handling devices, you should always be aware that ESD pulses can occur and should minimize the likelihood of it. Chips can be seriously impaired, even without a visible spark, resulting in problems later on. There are a number of simple measures to take to prevent static buildup and ESD.

ACKNOWLEDGMENTS

Thanks go to Rik Farrow for the comments on the initial draft of this article and to Intel Corp and Bunnie Studios LLC for providing the imagery. A last word of thanks goes to Competa IT, my employer, for giving me the freedom to write this article.

Come and join us, right after LISA '09, in Baltimore, MD

CHIMIT 09

Symposium on Computer-Human Interaction for Management of Information Technology

November 7-8, 2009

Baltimore, MD

September 1, 2009
Registration Opens

October 2, 2009
Early Registration Closes

October 30, 2009
Online Registration Closes

www.chimit09.org

acm USENIX IBM hp

DAVID N. BLANK-EDELMAN

practical Perl tools: scratch the webapp itch with CGI::Application, part 2

David N. Blank-Edelman is the director of technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Automating System Administration with Perl* (the second edition of the Otter book), available at purveyors of fine dead trees everywhere. He has spent the past 24+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

LAST TIME, WE HAD THE PLEASURE OF exploring the basics of a Web application framework called CGI::Application (CGI::App). I appreciate this particular package because it hits a certain sweet spot in terms of its complexity/learning curve and power. In this column we'll finish up our exploration by looking at a few of the more powerful extensions to the framework that can really give it some oomph.

Quick Review

Let's do a really quick review of how CGI::App works, so that we can build on what we've learned so far. CGI::App is predicated on the notion of "run modes." When you are first starting out, it is easiest to map "run mode" to "Web page" in your head. You write a piece of code (i.e., a subroutine) that will be responsible for producing the HTML for that page and returning it as a scalar value.

To switch from one run mode to another using the classic CGI::App method, the first run mode produces an HTML page containing a form or a URL to follow. This form or link provides the name of the destination run mode in a hidden field (for POSTs) or as a parameter (for GETs). CGI::App looks at the incoming request and determines which run mode to call based on that info. If this sounds too cumbersome or too Web 1.0-ish, you can improve on this method by using the module CGI::Application::Plugin::Dispatch. We won't see an example here, but C::A::P::Dispatch lets you encode the run mode in the path used to call the script using clean, REST-y URLs.

Constructing a Web application using CGI::App consists of:

1. Writing a bunch of run mode subroutines to generate Web pages.
2. Placing them in a file to be loaded as a Perl module (with a little OOP pixie dust sprinkled on top).
3. Writing a separate three-line instance script that loads this module and sets things in motion (this is actually the script that gets called by the Web server directly).

If the CGI::App basics seem easy to grasp, then the mental model CGI::App presents is working for you. Other Perl frameworks like Catalyst seem, at least at first glance, to be more complex (although Dave Rolsky's excellent blog post at http://blog.urth.org/2009/07/what-is-catalyst-really.html shows this is mostly a PR problem), so perhaps something with this level of simplicity will appeal to you. If it does, then read on, because we're going to show a few features/plugin modules that can pump up the volume while mostly leaving this simplicity intact.

Working with Templates

The first and perhaps the most significant feature is the support for templating available in vanilla CGI::App. If you've ever had to construct a Web site or webapp that involved several pages, you've probably already done at least a basic form of templating, because you needed all of the pages to have some common elements such as headers and footers. Perhaps your code looked something like:

```
$webpage = $header;
$webpage =. generate_the_actual_content();
$webpage =. $footer;
```

Perl has had various template modules of varying complexity available since the Mesozoic era. Most are predicated on the notion that you'll write content that includes some special tokens or markup which is later replaced with real data by the templating engine. For example:

```
<html>
<head><title><TMPL_VAR NAME=PAGETITLE></title></head>
<body>
   <p>Dear <TMPL_VAR NAME=FULLNAME>:</p>
   <p>Get it <a href="<TMPL_VAR NAME=BOOKURL>">here!</a></p>
</body>
</html>
```

The right Perl code, when called with that document, will spit out the document with all of the <TMPL_VAR NAME={something}> mentions replaced with the contents of the named variable. The good news is that CGI::App has this code built in. Out of the box it supports HTML::Template, which uses templates like the one above. It can also use other templating engines (e.g., Template Toolkit), but for this column we'll stick with the built-in support.

To use HTML::Template from within a CGI::App application, you first tell it where to find the templates it will use. This is most commonly done in one of the global initialization subroutines for an application like cgiapp_init:

```
sub cgiapp_init{
   my $self = shift;
   $self->tmpl_path('/path/to/your/templates');
}
```

Once the webapp knows where to find templates, it can load them for processing using the load_tmpl() method. Once loaded, we can replace the <TMPL_VAR> tokens with their values using a set of param() method calls and then produce the final result using output(). Here's an example run mode showing all of these steps:

```
sub welcome_doc : Runmode {
   my $self = shift;

   # die_on_bad_params set to 0 tells HTML::Template to avoid getting huffy
   # if we attempt to replace a parameter that doesn't exist in the template.
   # This will come in handy in the next section of this column.
   my $template = $self->load_tmpl( 'begin.tmpl', die_on_bad_params => 0 );
```

```
$template->param(PAGETITLE => 'Welcome Page');
$template->param(FULLNAME  => 'Mr. Mxyzptlk');
$template->param(BOOKURL=>
  'http://oreilly.com/catalog/9780596006396/');
return $template->output();
```

This is easier and more flexible than the method we saw in the first column of using CGI.pm methods to construct pages. As a related aside, I should point out that there is nothing (the name notwithstanding) that requires you to generate HTML using HTML::Template templates. In the last sizable webapp I built, I used the built-in HTML::Template support to have the webapp generate custom shell scripts that could be run separately from the webapp. At some point you start to realize that all the world's a template.

CGI::App Plugins

Now let's move into the territory of CGI::App enhancements provided by various plugin modules. I'd like to highlight three of them and then finish up with a small example that demonstrates everything we've talked about in these two articles. CGI::Application::Plugin::ValidateRM is the first plugin I'd like to mention.

CGI::APPLICATION::PLUGIN::VALIDATERM

Any decent Web application will validate its input. If your form asks for a phone number, the users need to be told they've done something wrong if they type in a set of letters (PEnnsylvania 6-5000 notwithstanding). You can block input errors via JavaScript before they are submitted, but for Perl programmers without JavaScript experience, it's probably easier to validate the input server-side with a module like Data::FormValidator. C::A::P::ValidateRM lets you hook Data::FormValidator into CGI::App in a relatively easy-to-use fashion. First we'll look at how Data::FormValidator is called and then at how C::A::P::ValidateRM lets us use it in a CGI::App context.

Data::FormValidator expects to receive an "input profile." The input profile specifies which fields are optional/required, any constraints on the field, filters to run on the input before checking, and what to do about errors. Here's an example input profile:

```
{  required=> [qw(inputfirstname inputlastname)],
   filters => ['trim'],
   constraint_methods => {
     inputfirstname => qr/^[A-Z]+[A-Za-z-'.]{1,25}$/,
     inputlastname => qr/^[A-Z]+[A-Za-z-'.]{1,25}$/,
   },
   msgs => {
     any_errors => 'err__',
     prefix => 'err_',
   },
}
```

Before validation is attempted, all fields are filtered, so leading and trailing whitespace is trimmed using a built-in filter (Data::FormValidator has a number of others). This profile expects there to be two required fields. Those fields have to be composed of one to twenty-five alpha characters (plus a few punctuation characters), specified here as regexps. We are using a custom constraint here, but Data::FormValidator::Constraint has a number of preset constraints such as “ip_address,” “email,” and “american_phone” you could use—you don’t need to roll your own.

And, finally, in the input profile, we specify how validation results (messages, or msgs for short) will be reported. In the example above, we ask that something called err__ be set if there are any validation errors at all and that the error messages for each failed validation be returned using the convention err_{fieldname} (i.e., err_inputfirstname and err_inputlastname). Looking at how these validation results are used is a good segue to how forms are validated within CGI::App.

With C::A::P::ValidateRM, we use a method called check_rm():

```
use CGI::Application::Plugin::ValidateRM
        (qw/check_rm check_rm_error_page/);

# in some run mode subroutine...
my $results = $self->check_rm(
                         'get_basic_info',
                           {required=> [qw(inputfirstname inputlastname)],
                              ... # same stuff as above
                           }
) || return $self->check_rm_error_page();
```

The check_rm() method takes a “return run mode” and an input profile. If the input profile turns up any validation errors, the return run mode specified in the first argument will be called to produce an error page. This error page is returned instead of any HTML the run mode would normally generate.

The parts of the validation handling are starting to come together, so let’s review and see what we are missing. When CGI::App enters a run mode, validation code checks that the input sent to that run mode (e.g., posted from a form) is valid. If it isn’t, CGI::App calls the return run mode (usually the run mode we just came from) and asks it to produce an HTML error page so that the user can try resubmitting. How does a run mode actually generate an error page? That’s the last part we have to cover.

First, each run mode is passed a hash reference as a second argument if it is being called as a return run mode. This hash reference contains the validation error messages mentioned earlier. To make use of them via the templating already discussed, we add a line to each run mode to insert the validation results into the document being generated:

```
sub welcome_doc : Runmode {
   my $self = shift;
   my $errs = shift;

      ... # do the stuff for this run mode including load_tmpl()
   $template->param($errs) if ref $errs;
}
```

The bold line above inserts the error information into our template. We’ll need a slightly more sophisticated template to incorporate these error messages:

```
<html>
<head><title><TMPL_VAR NAME=PAGETITLE></title></head>
<body>
   <!-- TMPL_IF NAME="err__" -->
   <p> Some data in your form was missing or invalid. </p>
   <!-- /TMPL_IF -->
```

```
<form method="POST">
   <input type="hidden" name="rm"value="next_runmode" />
   First Name: <input type="text" size=30  name="inputfirstname"/>
   <TMPL_VAR NAME="err_inputfirstname"> <br>
   Last Name:</label> <input type="text" size=30  name="inputlastname"/>
   <TMPL_VAR NAME="err_inputlastname">
   <input type="submit" name="submit" value="Submit" />
</form>

</body>
</html>
```

This sample introduces the TMPL_IF syntax from HTML::Template. If the named parameter is present it will output the contents of the tag. In our validation input profile we specified that we wanted `err___ set` if any errors were encountered. Here's where that pays off. The other addition to our standard template is the use of `TMPL_VAR NAME="err_{fieldname}"` tags. These will get replaced with field-specific error messages (by default, CSS-styled in bold red text). The end result of all this fuss is that your webapp will accept input via a form, validate that input using a pretty powerful specification (worse case: written in Perl itself), and automatically spit out an error form with the valid information still filled in and any validation errors flagged with pretty error messages. All of that can be done with much less code than you'd normally need if you were writing it yourself.

That's the (considerable) positive side of using C::A::P::ValidateRM. The hidden negative side of using this plugin that isn't explicitly mentioned in any of the documentation is the increased complexity validation like this can add to each of your run mode subroutines. Before, your code path might have been super simple—you enter each run mode, do your work, and move on to the next run mode—but now you might have a second entry path into each run mode to handle validation errors. If you've written the run mode code to expect to run only once per session/user, any situation where the webapp doubles back on itself can complicate lots of things. It certainly can make debugging the application a little harder. Caveat coder.

CGI::APPLICATION::PLUGIN::SESSION

The next plugin we are going to see requires considerably less background to cover. CGI::Application::Plugin::Session lets you use the handy CGI::Session module easily from within CGI::App. Because HTTP is a stateless protocol, a Web programmer has to do a bit of work in concert with the user's browser to present the illusion that the user is operating within a "session." CGI::Session handles all of the behind-the-scenes plumbing for that (cookies, session caches and expiry, etc.). Using all of this power becomes really easy thanks to CGI::Application::Plugin::Session:

```
use CGI::Application::Plugin::Session;

sub run_mode_A : Runmode {
  my $self = shift;

  $self->session->param('some_parameter' => 'some value to store');
}

sub run_mode_B : Runmode {
   my $self = shift;
   $some_value = $self->session->param('some_parameter');
}
```

In the example above, the two run modes share a piece of information called "some_parameter" by storing and retrieving it as a session parameter. Neither run mode has to know just how that magic takes place behind the scenes. You can tweak just how this is done, e.g., what database is used for persistent storage, via CGI::Session setup arguments called from the plugin.

CGI::APPLICATION::PLUGIN::DBH

Our last CGI::App plugin for this column is another door opener. CGI::Application::Plugin::DBH makes it easy to bring the power of Tim Bunce's fundamental database-independent interface (DBI) package into your webapp. If your webapp needs to work with data found in an SQL database, C::A::P::DBH provides an efficient way to do it. The key efficiency this plugin provides is "lazy loading"; the plugin is smart enough not to spin up a database connection unless the specific run mode in play needs it.

To use C::A::P::DBH, you describe the DBI connection in the CGI::App initialization routine:

```
use CGI::Application::Plugin::DBH (qw/dbh_config dbh/);

sub cgiapp_init {
  my $self = shift;

  $self->dbh_config({standard DBI connect() arguments});
}
```

If you'd prefer to keep the configuration information in your instance script, arguably a better place for it, there's an alternative syntax mentioned in the documentation.

With your database configuration specified, the other run mode subroutines can make use of a DBI database handle:

```
sub lookup_data {
  my $self = shift;

  my $data = $self->dbh->selectrow_arrayref(qq{SELECT name,uid
         FROM users});
  # ... do something with $data->[0] and $data->[1]
}
```

C::A::P::DBH lets you use named DBI database handles if your webapp has a need for connections to multiple databases.

Sample Code

So let's put this all together into a toy webapp that demonstrates everything we've talked about in both parts of this series. We're going to look at the code in an outside-in fashion. The first piece of code a user executes is the instance script. This is the script that is called when we go to http://www.server.com/LoginExample.cgi:

```
use strict;
use lib '/path/to/webapps/lib';

use LoginExample;

my $app = LoginExample->new();
$app->run();
```

This super-simple script just spins up the application module and its run mode subroutines. That's a fairly sizable file (LoginExample.pm). Here it is in its entirety:

```
use strict;

package LoginExample;
use base 'CGI::Application';
use CGI::Application::Plugin::AutoRunmode;
use CGI::Application::Plugin::Session;
use CGI::Application::Plugin::DBH          (qw/dbh_config dbh/);
use CGI::Application::Plugin::ValidateRM  (qw/check_rm
        check_rm_error_page/);
use Data::FormValidator::Constraints       (qw/FV_length_between email/);

# Configure the database connection and the location of the templates
sub cgiapp_init {
   my $self = shift;
   $self->dbh_config( "dbi:SQLite:dbname=loginex.sqlite", "", "" );
   $self->tmpl_path('templates');
}

# The initial run mode displays a form and handles another pass
# through the form should it not be filled out correctly. It also
# stashes the time the script was first run by the user in a session object
sub get_userinfo : StartRunmode {
   my $self = shift;
   my $errs = shift;

   $self->session->param( 'starttime' => time )
        unless $self->session->param('starttime');

   my $template = $self->load_tmpl( 'getuser.tmpl',
        die_on_bad_params => 0 );

   # if we're showing errors from validation
   $template->param($errs) if ref $errs;

   return $template->output();
}

# First test to make sure it has received valid output. If it has, query a
# database and the session object for other fields and display the info
# via a simple template
   my $self = shift;

   # Validate input from getuser_info's form
   my $results = $self->check_rm(
      'get_userinfo',
      {  required              => [qw/fullname email/],
         filters               => ['trim'],
         constraint_methods  => {
            fullname           => FV_length_between( 1, 50 ),
            email              => email(),
         },
         msgs => {
            any_errors => 'err__',
            prefix => 'err_',
         },
      }
   ) || return $self->check_rm_error_page();
```

```
    my $template
        = $self->load_tmpl( 'showuser.tmpl', die_on_bad_params => 0 );

    my $email = $self->query->param('email');

    my $idnumber = $self->dbh->selectrow_array(
        qq{SELECT idnumber FROM users WHERE email = \"$email\"});

    $template->param( FULLNAME      => $self->query->param('fullname') );
    $template->param( EMAIL         => $email );
    $template->param( IDNUMBER      => $idnumber );
    $template->param(
        STARTTIME => scalar localtime $self->session->param('starttime') );

    $self->session->delete();
    $self->session->flush();

    return $template->output();
}

1;
```

After the modules are loaded, we perform all of our initialization work by configuring the database handle and template directory. Next come the two run modes for this module: get_userinfo and show_userinfo. The first run mode uses a template to display a two-field form (we'll see the template in just a moment) to collect information from the user. In the process, it makes note of the time the run mode was first entered by squirreling away that value in a session object. This run mode also handles a redisplay of that form should the user not provide valid information.

The second run mode, show_userinfo, checks the input it has been passed from get_userinfo. If the input is not valid, it creates an error page by calling get_userinfo again. If the input is valid, it retrieves info from the form parameters (`$self->query->param()`), an SQLite database (via DBI), and the session object (`$self->session->param()`). This information is inserted into a template, the session object is disposed of, and CGI::App is handed output to display.

To see what is displayed for each run mode, let's look at the two templates. Here's getuser.tmpl:

```
<html>
<head><title>Login Example </title></head>
<body>

    <!-- TMPL_IF NAME="err__" -->
    <p> Some data in your form was missing or invalid. </p>
    <!-- /TMPL_IF -->

<form method="POST">
<input type="hidden" name="rm" value="show_userinfo" />
Full Name: <input type="text" size=30  name="fullname"/>
<TMPL_VAR NAME="err_fullname"> <br>
Email: <input type="text" size=30  name="email"/>
<TMPL_VAR NAME="err_email">  <br>
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
```

and here's showuser.tmpl:

```
<html>
<head><title>Login Example </title></head>
<body>
   Full Name: <TMPL_VAR NAME=FULLNAME><br>
   Email: <TMPL_VAR NAME=EMAIL><br>
   ID: <TMPL_VAR NAME=IDNUMBER><br>
   Started: <TMPL_VAR NAME=STARTTIME>
</body>
</html>
```

Leftovers

We're very out of room (it's amazing how fast you can use up 140 characters), so let me just say that there are a number of CGI::App plugins you'll definitely want to explore if you decide to start building Web applications with it. The best place to start is to look at the "bundled" version being produced (search CPAN for "Titanium") to have all of the current "best practices" CGI::App plugins at your fingertips. Beyond that, search CPAN for "CGI::Application::Plugin" for a plethora of choices. Have fun writing webapps (perhaps for the first time) with CGI::Application. Take care, and I'll see you next time.

Save the Date!

S³ Solaris Security Summit

Tuesday, November 3, 2009 - Baltimore, MD

Join us in Baltimore, MD on November 3, 2009, for a free one day event for the latest in how to **Make System Security Work for You!** The summit is co-located with the 2009 Usenix LISA conference.

Join practitioners, designers, and developers of Solaris security technologies for up to date information, presentations, and demonstrations of the latest in Solaris security solutions, OpenSolaris and other Open Source security projects, and virtualization and cloud security initiatives.

See you in Baltimore!

FREE registration at http://wikis.sun.com/display/secsummit09

PETER BAER GALVIN

Pete's all things Sun: VMware vSphere 4 vs. Microsoft Hyper-V R2

Peter Baer Galvin is the chief technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is co-author of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter blogs at http://www.galvin.info and twitters as "PeterGalvin."

pbg@cptech.com

LONG GONE ARE THE DAYS WHEN "SUN Microsystems" meant only Solaris on SPARC. Sun is pushing hard to be a platform provider for multiple operating systems, as well as a provider of their own Solaris operating system. In some ways this column is a continuation of my April column (*;login:* April 2009, Volume 34, Number 2), which contained a virtualization guide. That column discussed the virtualization offerings created by Sun. This column explores the rich, controversial, and important terrain of two of the market leaders in virtualization, VMware and Microsoft. The topic is not Sun-specific but is probably of interest to many Sun customers. The Sun x86 servers are certified to run VMware and Windows, as well as Solaris and Linux. In my experience, Sun x86 servers, especially the x4600 with its eight sockets and quad-core CPUs, make excellent virtualization servers. The question then becomes, which virtualization technology to run? OpenSolaris has Xen built in, but many sites want mainstream and well-tested solutions. That brings us to the two contenders discussed in the remainder of this column.

VMware is certainly the company one thinks of when "virtualization" is mentioned, and for good reason. They have the largest market share of virtualization solutions, have had products available for many years, and are leading the way to the virtualized data center through their product features and best practices. Microsoft gets everyone's attention when they enter a market, and although late to the game, they are attacking it fiercely by including virtualization in Windows Server. The question on many minds is, "Which is better?" or even "Is Hyper-V good enough?" In this column I'll discuss the features of the latest server virtualization offerings from both companies—VMware vSphere 4 was recently announced and is already shipping, and Microsoft Hyper-V R2 is part of Windows Server 2008 R2, which is currently in beta test and expected to ship in October. Along with features, the discussion must also include (list) pricing, because much of Microsoft's push is based on the lower cost of Hyper-V.

VMware vSphere 4

vSphere 4 is the name of a suite of products from VMware consisting of the ESX and ESXi type 1 hypervisors installed on servers and the vCenter Server administration software. The important features of vSphere 4 include:

- Distributed Resource Scheduler (DRS): aggregates resources across one or more compute clusters and dynamically allocates them to VMs based on business logic.
- Distributed Power Management (DPM): automates energy efficiency in DRS clusters by optimizing power consumption.
- Virtual Machine File System (VMFS): clustered file system, shares storage among cluster nodes.
- Thin Provisioning: dynamic allocation of storage as needed.
- Virtual Switch: provides advanced networking features per guest on a host.
- vNetwork Distributed Switch: simplifies provisioning, control, and administration of VM networking.
- vMotion: live migration of VMs across servers in a cluster with no disruption or loss of service.
- Storage vMotion: relocates virtual disks among storage resources within a cluster (but not between clusters).
- High Availability (HA): automated restart (within minutes) of VMs on other cluster nodes in the event of server failure.
- Fault Tolerance: a second VM mirrors a primary one in lockstep, providing continued operation if the first VM or its hardware fails (but limits VMs to only 1 vCPU, and use of many other features not allowed).
- Data Recovery: agentless backup of VMs (for small environments).
- vShield Zones: creates and enforces security zones that are maintained even during vMotion.
- VMsafe: enables the use of third-party security products within VMs.
- vApp: logical collection of components of an application, described via OFV format.
- Site Recovery Manager (SRM): automates DR failover between sites and failover testing, via integration with networking and storage components.

Both VMware vSphere 4 and Microsoft Hyper-V manage multiple hosts as clusters of resources. A cluster is the entity into which a VM is deployed. Cluster resources are allocated to VMs. VMware, through vMotion, can move VMs among hosts in a cluster. If a cluster node fails, the VMs that were running on that node can be automatically restarted on another node in that cluster. Cluster hosts share access to storage to allow this functionality. Note that vSphere has no native ability to replicate storage between clusters (such as to a DR site) but, instead, integrates with replication provided by storage vendors (which are usually licensed features of the storage arrays).

There is no standard benchmark of virtualization performance currently available, although the SPEC organization is working on one. VMware has published their own VMmark benchmar, but, because it includes in its testing the performance of Linux running on more than one core and because Hyper-V does not support Linux beyond one core, there is no way to run that test across both virtualization technologies. This lack of standard benchmarks leaves it up to a given site to run tests to determine performance differences. I expect that VMware is more efficient in terms of CPU and memory use, but have not yet proved that via testing.

Pricing of software can be complicated, and virtualization solutions are no exception. Table 1 compares the flavors of VMware vSphere and their list prices.

Specification	Standard	Advanced	Enterprise	Enterprise Plus
Cores per CPU	Up to 6	12	12	12
Virtual cores (per guest)	4	4	4	8
RAM	256GB	256GB	256GB	Unlimited
Failover	0	16	16	16
Consolidation	Hypervisor, agent, thin provisioning, update manager, VCB	Hypervisor, agent, thin provisioning, update manager, VCB	Hypervisor, agent, thin provisioning, update manager, VCB	Hypervisor, agent, thin provisioning, update manager, VCB
Availability	HA	HA, live migration, fault tolerance, vShield zones, data recovery	HA, live migration, fault tolerance, vShield zones, data recovery	HA, live migration, fault tolerance, vShield zones, data recovery
Automated resource management			DRS, DPM, storage vMotion	DRS, DPM, storage vMotion
Simplified operations				3rd-party multipathing, distributed switch, host profiles
Cost	$795 per processor	$2,245 per processor	$2,875 per processor	$3,495 per processor

TABLE 1: VMWARE VSPHERE COMPARISONS

To these product costs must be added any operating system licenses and application licenses. Also needed is a license for one or more copies of vCenter Server: the "Standard" version has no host limits, can link to other vCenter Servers for consolidated management, and includes "Orchestrator," a VMware automation tool. It costs $4,995. The more limited "Foundation" version costs $1,495 and is limited to 3 ESX hosts. ESXi itself can be run free of charge, but optional maintenance can add $495 per year to its cost. Of course many sites execute a site license agreement with VMware, greatly reducing the per-processor cost. In general, a site license for vSphere or any operating systems is not an unlimited license; rather, it is a discount based on a volume purchase of licenses. A "true up" occurs periodically in which the number of instances in use is calculated and the total cost to the site tallied.

Microsoft Hyper-V R2

Microsoft Hyper-V R2 is at the time of this writing in "release candidate" form in Windows Server 2008 R2. Microsoft claims that half of its infrastructure is currently virtualized (via Hyper-V presumably), so clearly Microsoft feels that it is ready for production use. But because it is in release candidate form, it is difficult to draw conclusions about its use in the field, production deployments, and even final features and performance.

Hyper-V is Microsoft's virtualization layer, the technology that allows virtual machines to run within Windows, just as ESX and ESXi are VMware's. Hyper-V is simply a feature of Windows Server. Technically it is a "type 2" hypervisor, as virtual machines run under a host operating system and not just a hypervisor. However, this line is blurry, as ESX also includes a Linux component in its host operating system. According to Microsoft, they provide a thin type 1 hypervisor layer that runs alongside the full Windows Server software. In their implementation, device drivers have low latency access to the hardware, and therefore type 1-like performance.

Hyper-V is part of the complete virtualization solution from Microsoft. The associated management component is Microsoft System Center, the software generally used by Microsoft infrastructure shops to manage their Windows Server deployments. A new add-in to SC is SCVMM—Microsoft System Center Virtual Machine Manager. SCVMM is able to manage not only Hyper-V-hosted guests but also Virtual Server, VMware Server, and VMware ESX and GSX guests. SVCMM works well in conjunction with the other standard components of System Center, such as the Configuration Manager and Operations Manager. This tight integration is a boon to sites that already make use of those other tools.

SCVMM has a host of features, making it a fairly complete manager in Microsoft environments. For example, it will intelligently place VMs onto the hosts with the most available resources, based on the resource needs of the VMs in question. Also included are physical-to-virtual (P-to-V) tools to take a physical server and generate a virtual machine image from it, and a virtual-to-physical (V-to-P) tool that does the reverse. P-to-V is the way most sites generate their first VMs, capturing existing systems and virtualizing them. The utility of V-to-P should not be overlooked, however. It can be useful for debugging, to test whether virtualization is causing the problem or whether it is virtualization independent. It can also be useful for performance testing. Finally, it ensures that even if an application has been virtualized, there is a back-out plan if that virtualization results in insufficiency.

Other useful features include the full scriptability of SCVMM actions via the standard PowerShell tools. Scripting enables repeatability and transportability—for example, a library of scripts that create and manage virtual machines can be copied between sites to allow uniformity and administration efficiency.

A Hyper-V cluster provides high-availability functionality by restarting VMs on other cluster nodes if a node fails. Hyper-V R1 has a "Quick Motion" feature that allows a VM to be moved between cluster hosts, but it lacks Vmotion's ability to do the move "instantly" (in less than a second). Because the move can take several seconds, network connections to the VM can fail during the move with resulting impact to production uptime. Quick Motion greatly diminishes an administrator's ability to manage resource use in a Hyper-V cluster. A running VM cannot be moved to another server seamlessly. If a server needs maintenance, for example, moving the VMs to another server is a downtime event. Hyper-V R2 has a feature called Live Migration that should address this issue and put it on a par with vMotion.

The features available to Windows administrators depend on the version of Windows being used. The available versions include Web, Foundation, Standard, Enterprise, and Datacenter. There is also an Itanium version that does not support Hyper-V, as well as an HPC version. Fundamentally, Enterprise, Datacenter, and Standard can include Hyper-V, but there are also versions of those operating system flavors that do not include it. A Server Core version of Enterprise, Datacenter, and Standard includes all the functionality but without the GUI. This version is intended for headless servers, decreasing the size of the installation and installation time.

Table 2 shows the Windows Server 2008 R2 versions, features, and limits.

Specification	Standard	Enterprise	Datacenter
X86 sockets (up to 32 cores)	4	8	32
X64 sockets (up to 64 cores)	4	8	64
X86 RAM	4GB	64GB	64GB
X64 RAM	32GB	2TB	2TB
Failover cluster nodes	0	16	16
# client access licenses (CALs) included	5	25	0
Cross-file replication (DFS-R)	No	Yes	Yes
Virtual Image Use Rights	1	4	Unlimited
Cost	$999 per host	$3,999 per host	$2,999 per processor

TABLE 2: WINDOWS SERVER 2009 R2 COMPARISONS

Again, application licenses must be added to these costs, as well as the cost for non-Windows guests and any Windows guests beyond the number granted with the OS license.

"Virtual Image Use Rights" determines how many Windows Server guest virtual machines can be created when the given operating system is the host. Unlimited guests are allowed, but only a limited number of Windows Server guests are granted in the license. Windows Server 2008 Standard Edition can have one Windows Server guest VM, Enterprise can have four Windows Server guests, and Datacenter is limited only by available resources. The Windows Server license includes the use of Windows Server as a guest under Hyper-V on that system.

You can use the various flavors of Windows as guests, depending on licensing terms.

- Windows 2008 without Hyper-V can be a guest and can use 1, 2, or 4 virtual CPUs.
- Windows 2003 can use 1 or 2 virtual CPUs.
- Windows 2000 can use 1 virtual CPU.
- SUSE Enterprise Linux can use 1 virtual CPU.
- Windows Vista can use 1 or 2 virtual CPUs.
- Windows XP can use 1 virtual CPU (although Windows XP Professional with SP3 and XP Professional x64 Edition can use 2 virtual CPUs).

Note that Red Hat and Microsoft have announced a joint support agreement. RHEL will be supported as a guest within Hyper-V, and Windows Server 2008 will be supported within RHEL guest VMs. As of this writing, neither of those options is available for production use.

There is no Microsoft equivalent of ESXi—rather, Windows is installed as well as Hyper-V, with Windows being the virtual machine manager (VMM). The minimum installation of Windows Server Core plus Hyper-V takes 2.6GB of disk space. The more complete Windows Server releases take even more space. ESXi takes 70–100MB of disk space.

A Hyper-V VM consists of a configuration file, the image file (in VHD format), saved state files, and differencing disks (AVHDs). Hyper-V supports full snapshot functions, including creation, deletion, and merging. Merging is needed if snapshots that depend on other snapshots are deleted. Snapshots are just block differences. Each snapshot refers to the previous snapshot and just records differences. If a snapshot is deleted, other snapshots may depend on some blocks in that snapshot, and those blocks must be merged into the remaining snapshots. However, merging snapshots is only possible when a virtual machine is halted. The other snapshot commands may be done on live VMs.

Comparison

Table 3 compares all the major features and resource limits of vSphere 4 and Hyper-V.

Aspect	**VMware vSphere 4**	**Microsoft Hyper-V R2**
Host		
CPUs supported	Recent AMD, Intel	Recent AMD, Intel
# CPU cores supported	64	64
Memory supported	1TB	2TB
I/O devices supported	IDE, SCSI, SAS, SATA, FC, 1Gb and 10Gb Ethernet, iSCSI, NFS, FCOE, Infiniband	IDE, SCSI, SAS, SATA, FC, 1Gb and 10Gb Ethernet, iSCSI, CIFS, FCOE, Infiniband
Memory optimization	Over-commit, transparent page sharing, ballooning, large-memory pages	Dynamic memory allocation
Platform support	Fewer vendors	More vendors
Supported storage of guest VMs	Direct, SAN, NAS, iSCSI	Direct, SAN, iSCSI
Number of nodes in a cluster	32 nodes if < 40 VMs per node	16
Guest		
Operating systems supported	Asainux, CentOS, Debian, FreeBSD, OS/2, Solaris 10, SCO OpenServer, SCO Unixware, Windows Server, RHEL, SUSE, MS-DOS, Netware	Windows Server, Vista, XP, SUSE Linux
Operating systems tools provided |(per OS)	Yes, for most guests	Yes, for most guests
# virtual CPUs supported	8	4
# guests per host	256 running	512 (192 running)
Amount virtual memory	256GB	64GB
Virtual NICs	10	Yes, limit unknown
# of snapshots	32 per VM	50 per VM
Types of guests supported	32-bit, 64-bit, simultaneously	32-bit, 64-bit, simultaneously
Ability to hot-add disk images and external storage	Yes	Virtual SCSI devices only, not IDE
Features		
VM move	Live	Live
Direct I/O	VMDirectPath I/O	—
VM synchronization	With limits (1 vCPU, many features disabled)	No

Directly boot from VM image	Only if ESXi installed	Yes
P to V	Included	Included
V to P	Included	Included
HA via clustering and failover	Yes	Yes
Replication	Integration with 3rd-party storage products	Yes (DFS-R)
Performance monitoring	Yes, vCenter Server	Yes, SC Operations Manager
Network features	Virtual switch, VLAN tagging, Network vMotion, Network traffic shaper, IPv6, CDP, NIC teaming	Standard Windows Server 2008 features
Storage features	Thin provisioning, consumption-based monitoring, reports and topology maps, LUN discovery, adaptive block sizing, storage vMotion	Standard Windows Server 2008 features
Patching of guests	vCenter Update Manager (both running and halted guests, Windows and some Unix)	Standard Windows Server 2008 features for booted Windows guests, Offline Machine Servicing Tool for halted Windows guests
Security	Layer 2 security policies, vShield, VM-safe 3rd party security products	Native firewall, 3rd party security products
Backups	Native via VMware Data Recovery, Support from major vendors	Native, Support from major vendors
Resource management	Yes, many options	Yes, some options
Physical server power on / off as needed	Via VMware DRS, DPM	No
ISV support	Strong	Strong
VM format conversion	VMware workstation, Linux, VHD	VHD, VMDK
Market share (new orders, Q2 2008, IDC)	44%	23%
Performance	VMMark results published (no industry standard benchmark exists)	None published
Cost	See VMware section	Included with some Windows Server 2008 editions, see Hyper-V section

TABLE 3: VSPHERE 4 AND HYPER-V COMPARISONS

The list of functions, features, and limits needs to be compared with the needs of a data center. Applying that filter, it could be the case that, for a given deployment or environment, the two options analyzed here are equivalent. For example, the two offerings are relatively the same for a site needing to virtualize Windows Server 2008 on a host with 8 processors, 64 cores, 256GB of memory, needing 4 vCPUs per guest, 8 guests, live migration, H/A, and storage management. Comparisons of cost should also be considered.

To use VMware to accomplish this task, the list price cost would be $2,245 per socket for vSphere advanced, plus $1,495 for vCenter Server Foundation, plus 8 Windows Server 2008 Standard (no Hyper-V) licenses at $971 each, for a total of $27,223.

To use Windows Server 2008 with Hyper-V to accomplish the same task, the list price cost would be $3,999 for Windows Server 2008 Enterprise (granting 4 guest licenses), plus System Center at $1,497 (although that is already in place at many Windows sites), plus 4 Window Server Standard licenses (for the other 4 guests), for a total of $9,376. Note that this pricing could change if Microsoft changes its licensing terms with the release of Windows Server 2008 RC and SVCMM R2.

There are some similarities and many differences between the features of these two offerings. In many cases, a shortcoming can be redressed by adding a third-party tool to an infrastructure. There are many such tools to chose from, but adding a tool brings with it added complexities, training needs, maintenance efforts, and so on. Also, consider that the virtualization market is very dynamic. Consider that Virtual Iron was an early entry into the virtualization market, but its purchasers were left without any options to expand its use when Oracle purchased the company and decided to terminate sales, even to existing Virtual Iron customers.

Further, datacenter managers, while determining the total cost of virtualizing an environment, need also to consider the impact of virtualization on the entire facility. Virtualization will likely:

- Decrease the number of physical servers.
- Increase the per-physical-server cost.
- Increase the number of OS instances (“virtual server sprawl”).
- Decrease overall power and cooling costs.
- Increase power and cooling needed per rack containing virtualization infrastructure.
- Increase network throughput needed per rack, possibly resulting in the need for 10Gb networking.
- Increase storage load (where virtual machines are stored).

Conclusions

It is likely that hypervisors will be “free.” Whether as a hardware component (the hypervisor that ships in the firmware) or as a software component (a virtual machine engine shipping as a feature of the operating system), the ability to virtualize will be included. Virtualization will therefore be ubiquitous. A free and ubiquitous feature is difficult for application vendors and infrastructure managers to ignore.

It is also likely that IT infrastructure will migrate toward “cloud” architectures in which systems and storage are resources that are trivially allocated and deallocated as needed based on application demand. Some applications do not lend themselves to cloud technologies, including applications that scale vertically, as a server grows, rather than horizontally, across servers. But those applications that can be implemented, monitored, scaled, and managed via cloud technologies will make the move due to those compelling cloud features and the cloud technologies that leverage virtualization. Networking likewise will evolve to allow fast access among all resources, and easier access to resources at remote sites (such as DR sites). Networking vendors will try to design (and sell) “one connection fits all” infrastructure in which one networking wire (or two for redundancy) handles all network and storage traffic.

Virtualization of applications will likely become the default, assuming virtualization vendors continue down the path of unifying the VM format.

Application vendors will commit to the vision, first illuminated by VMware, in which an application and its operating system are pre-installed, preconfigured, and pre-tuned in a virtual machine. That entity would then be the product shipped by the application vendors, and customers would simply take the virtual machine and deploy it on their infrastructure. The only sticking point in this scenario is how operating system vendors license their products. Sun Microsystems and most versions of Linux already allow free download and use of their operating systems, with payment made only if the customer wants to keep the software and get support. It is therefore allowable to ship a virtual machine containing Solaris and the application to the application's customer. Other vendors (with the notable exception of Microsoft) will likely follow suit, to allow their operating system to be bundled by application vendors.

VMware currently has a clear market and functionality lead, but can they maintain this lead in the face of competition from established vendors, both in features and in price? It is likely that they will have to decrease the premium that data centers have to pay, per CPU socket, to have that socket managed by VMware software.

Hyper-V market share will grow once R2 is released, because the addition of the Live Migration feature enables it to solve many more problems, in many more environments. It will also grow because it is freely included with some versions of Windows Server, and because it is a Microsoft-supported product. Its growth into large data centers will be limited by its scant support for other operating systems.

Currently, datacenter management would be well served to analyze which operating systems they are using, and determine from that which virtualization platform to evaluate using. If there are a large number of Windows Server systems, or a majority of the systems are Windows Server, then Hyper-V becomes a tempting direction. However, the newness of Hyper-V R2 dictates that careful testing, including reliability and performance, be done before any final decisions are made. Certainly its lack of support for Solaris and most Linux releases will limit its use in many environments. Also, installation planning should determine which release of Windows Server best suits the environment and how to deploy that version. VMware posted a video (see References) comparing the installation time and effort of VMware ESXi and Windows Server Core to demonstrate how much more work is involved using standard Windows deployment methods.

The costs and complexities of virtualization, from the tools through deployment and management best practices, are detrimental to datacenter managers. However, many sites are determining that the benefits far outweigh these issues. These benefits include reduced hardware footprint, power, and cooling use; improved application management, reliability, and maintainability; and easier application deployment and disaster recovery. The variations in data centers, priorities, applications, and business drivers require each datacenter management team to evaluate the gains and losses for themselves.

More information about VMware vs. Hyper-V is available in a free (registration required) white paper available from http://ctistrategy.com. In this white paper, I expand on the information in this column by providing analysis of why to virtualize, what to virtualize, more feature details, and a set of next steps for datacenter managers to consider. Also at ctistrategy.com is a decision guide that allows determination of the likely best virtualization fit based on site requirements.

Random Tidbits

The Oracle purchase of Sun Microsystems, although approved by shareholders, has not been consummated as of this writing. Certainly Sun will be changing, whether or not the purchase becomes final. Watch for analysis and updated product information in future versions of this column.

REFERENCES

Cio.com, Virtualization Vendors to Watch in 2009: http://www.cio.com/article/478388/_Virtualization_Vendors_to_Watch_in_.

Comparison of hypervisors by VMware: http://www.vmware.com/technology/whyvmware/architectures.html#c132894.

Hyper-V Glossary: http://blogs.msdn.com/virtual_pc_guy/archive/2008/02/25/hyper-v-terminology.aspx.

Computerworld, "Review: Microsoft's System Center Virtual Machine Manager," October 23, 2008.

Microsoft Hyper-V landing page: http://www.microsoft.com/windowsserver2008/en/us/hyperv-r2.aspx.

Microsoft Hyper-V supported guests: http://www.microsoft.com/windowsserver2008/en/us/hyperv-supported-guest-os.aspx.

Microsoft Systems Center pricing: http://www.microsoft.com/systemcenter/en/us/management-suites.aspx.

Microsoft Windows Server 2008 feature comparison: http://www.microsoft.com/windowsserver2008/en/us/compare-specs.aspx.

Microsoft Windows Server 2008 price list: http://www.microsoft.com/windowsserver2008/en/us/pricing.aspx.

Microsoft, "Windows Server 2008 Hyper-V Technical Overview," January 2008.

David Patterson's SC08 keynote: http://www.theregister.co.uk/2008/11/26/patterson_keynote_sc08/.

Red Hat licensing costs: http://www.redhat.com/f/html/partners_us_skulist.html.

TechAlpha, "Ripple Effects from Virtualization," March 11, 2009.

VMware pricing: http://www.vmware.com/files/pdf/vsphere_pricing.pdf.

VMware video of installation time of Windows Server Core and VMware ESXi: http://www.vmware.com/technology/whyvmware/resources/esxi-hyper-v-installation.html.

VMware vSphere configuration guide: http://www.vmware.com/pdf/vsphere4/r40/vsp_40_config_max.pdf.

VMware vSphere 4 datasheet: http://www.vmware.com/products/vsphere/.

vSphere glossary: http://pubs.vmware.com/vsp40_e/wwhelp/wwhimpl/common/html/wwhelp.htm#href=intro/master_glossary.html&single=true.

Windows as a guest operating system: http://technet.microsoft.com/en-us/library/cc794868%28WS.10%29.aspx.

Windows Virtual PC RC: http://blogs.msdn.com/virtual_pc_guy/.

DAVE JOSEPHSEN

iVoyeur: packet-level, per-user network access control and monitoring

Dave Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and is senior systems engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

WE WATCH THE LOGS SCROLL BY, USERS in the offices surrounding us using VPN to access specific ports on the specific servers they need in the server room based on business roles that apply to them. My mind shifts down one level. The gateway device allows them access to the Internet, where they loop back to the public VPN endpoint for this building. PKI and LDAP Bind authenticate them to the VPN endpoint, and it creates packet filter rules for them based on the outcome of LDAP queries, loads their rules to the VPN users' PF anchor, and hands them routes to privileged internal subnets. This is the kind of interaction that makes my frontal lobes buzz. Lots of pieces, lots of layers. Physical, logical, human, abstract; lots of things to understand—no, lots of *ways* to understand, and we do, because we built it.

It all fits so well together that it's hard to see the pieces now—like these unrelated projects were intended to be subsystems of the whole we've created. The best systems I've built work this way; the ones that are going to stick to the wall. I imagine that real scientific discovery feels something like this. When it's done and you step back and look at it, you know that you've found *the* answer to this particular question, and that when you move on it will remain. This answer is right, it is *truth*, and anything else is a compromise…a kludge.

We sit in silence, lost in thought, mulling details; routing, schema extensions, NAT, tunnels. What if . . . no, that's accounted for. But then what if . . . no, the design takes care of that too.

Finally my cohort blinks and shakes his head. "Whose idea was this?"

All I can do is chuckle and shrug. I honestly can't remember. The design has been haunting me for what feels like years, but I can't say for sure it originated in my brain. It was a progression. An artifact of our collective familiarity with these tools, our familiarity with each other, and our daily carpool brainstorms. Having a need for a network access control system probably didn't hurt, but in reality it wasn't much of a catalyst either. Solutions like this build themselves when they are ready to be built, and which of us is to blame isn't important, even if it were answerable. This thing scrolling away before us is a product of our "us-ness," and also, it's *awesome*.

"It's a pretty good idea," he says.

"Yeah, not bad," I reply.

I should pause here for a short disclaimer: The thing I hate and dread about writing implementation articles is sharing my code. Easy there, Captain Open Source, I'm not being proprietary corporate guy. The thing I hate about sharing my code is inviting you, dear reader, into my brain. It's far easier to stay on the English composition side of this equation, where a well-placed semicolon or two might disguise the blathering idiot I truly am (unlikely, but possible). Sharing source code with the readership of *;login:*, on the other hand, is something like showing up naked to a photography convention. It's not something to be done lightly if you value the respect of your peers, and, being painfully aware of that, I wanted to make sure you knew the code herein is mine. My cohort is innocent in that regard, his kimono firmly closed as it were.

So, when did this idea coalesce? I don't know that either, but the implementation started with a fortuitous network redesign. We moved our headquarters in April, which provided us with the opportunity to redesign the network from the ground up. New numbers, new segments, new providers—the whole deal. A big part of the redesign was this VPN-based access control scheme we'd already been kicking around. The idea was that instead of having an "internal" subnet for our employees, we'd have an untrusted segment officially referred to as "public." University administrators are probably pretty familiar with this idea: I've sometimes heard them call it "the heathen zone."

It's assumed that bad things happen in the heathen zone as a matter of course, and that the systems within it should be allowed relatively unfettered (but NATed) access to the Internet and not much else. In our new corporate network, if a heathen wants to use services that don't have public external addresses (POP, printers, etc.), they should do what they'd do at Starbucks—no, not pretend to read while hoping someone will talk to them, but VPN in. When they do that, we can give them access to the exact services on the specific systems they need, and we can tie their traffic to a UID and monitor/log it. The new network was designed with this in mind.

The next step was figuring out an access control scheme (language?). LDAP was the obvious choice as a database, but how to implement it? We knew we wanted a very flexible role-based scheme that would scale and make it easy to optimize for performance during searches by limiting the search base. We also wanted to be able to consolidate a few other LDAP systems into it for things like FTP and mail authentication, and even asset control and machine inventory. I've done a lot of things in my professional career, but getting LDAP right the first time isn't one of them. In fact, I have rarely gotten a design that I like for more than a few weeks. I've also never found a design that I could move from one company to another. In this case, I think on the third or fourth try we got something that stuck.

We modified the schema to add a few objects of our own, for things like networks, servers, and a role object, with a socket-style "grant" attribute that specifies host/service tuples. The easiest way to give you a feel for how it works is to show you what the VPN endpoint does when an employee logs into it.

Step 1. Given a unique UID, look up the user's DN:

```
ldapsearch (uid=dave) dn
```

This yields something like:

```
uid=dave,ou=foo,ou=bar,dc=dbg,dc=com
```

Step 2. Given a user's DN, look up what it's been granted access to:

```
ldapsearch (&(member=uid=dave,ou=foo,ou=bar,dc=dbg,dc=com)
(objectclass=dbgRole)) dbgGrants
```

This returns a list of server/service tuples that look like this:

```
fooserver.dbg.com:login
```

Step 3. For each tuple, resolve the IP address and port number:

```
ldapsearch (&(cn=a.ig05.dc4.dbg.com)(objectclass=dbgNetwork)) dbgAddress
```

One or more IP addresses in CIDR notation may be returned.

At the moment, there is no service object in LDAP that maps the service to a port number. This is because the service definition is arguably relative, given that future consumer programs might use a different port for the same service name or might not want them mapped to TCP port numbers at all, and anyway creating 30,000+ LDAP objects that mostly won't ever be used just seems wrong. At the moment, I think it's better that the consumer application interpret the service name ("login" in this example) for itself. On the VPN gateway we do this with a slightly modified copy of the /etc/services file.

The VPN endpoint runs OpenBSD, with OpenVPN and the PF (Packet Filter) firewall. There are three OpenVPN configuration parameters that make this design possible. The first is actually optional: --auth-user-pass-verify allows us to authenticate the user via LDAP, which saves us from having to issue new certs every time a heathen forgets its password.

The next two go hand-in-hand: --client-connect and --client-disconnect. These allow us to call a script of our choosing when a client connects and/or disconnects, and are pretty much the bailing wire holding this all together. I wrote a shell script I call VPLDPF (VPN-LDAP-PF) that gets the user's UID from OpenVPN as $1, along with a bunch of other interesting variables. VPLDPF's job is to perform the necessary LDAP queries to figure out what hosts/ports the heathen gets access to, translate these into PF rules, and finally load them into PF. The script is available linked under this article at http://www.usenix.org/login/2009-10/.

PF's "anchor" feature makes this sort of automated dynamic firewall configuration safe and easy. Anchors are named sets of filter rules that can be maintained and loaded separately from the main PF rule set. VPLDPF uses LDAP searches to create PF filter rules for every user who logs in and then stores them in a file named after the user in /etc/pfanchors/vpnusers. Once we've told PF that we'll be using an anchor called vpnusers by adding anchor 'vpnusers/*' to /etc/pf.conf, VPLDPF can load, for example, Bob's rule set:

```
pfctl -a vpnusers/bob -f /etc/pfanchors/vpnusers/bob
```

Going into it, I thought the initial population of LDAP was going to be time-consuming, but the "roles" scheme we came up with didn't take much effort,and has made it pretty easy to get very granular permissions on an individual employee basis. How granular? Let's take a look at "Bob," a pretend employee modeled after a real project manager.

Bob's DN is:

```
uid=bob,ou=projectManagement,ou=staff,dc=dbg,dc=com
```

Running an ldapsearch for object class dbgRole with Bob's DN in the member attribute, we find that Bob has three roles assigned to him:

```
dn: cn=employee,ou=roles,dc=dbg,dc=com
dn: cn=HQVPNUser,ou=roles,dc=dbg,dc=com
dn: cn=PM,ou=roles,dc=dbg,dc=com
```

The employee role contains the following dbgGrants attributes:

```
dbgGrants: fileServ1.dbg.com:login          # a fileserver
dbgGrants: fileServ1.dbg.com:http
dbgGrants: fileServ1.dbg.com:https
dbgGrants: mail.dbg.com:http                #an email server
dbgGrants: mail.dbg.com:https
dbgGrants: mail.dbg.com:pop3
dbgGrants: mail.dbg.com:smtp
dbgGrants: mail.dbg.com:xmpp-client         #this is the jabber port
```

The HQVPNUser role contains the following dbgGrants:

```
dbgGrants: ns.hq.dbg.com:domain
```

And the PM role contains the following dbgGrants:

```
dbgGrants: pm.dbg.com:http        # the project management server
```

Using the server name as a CN to resolve the IP address and looking in the services file for the port, VPLDPF created the following PF rules for Bob:

```
pass in inet proto tcp from 10.253.21.10 to 10.21.1.2 port = ssh flags S/SA
    keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.1.2 port = https flags S/SA
    keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.1.2 port = www flags S/SA
    keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.64.101 port = https flags S/
    SA keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.64.101 port = www flags S/
    SA keep state
pass in inet proto udp from 10.253.21.10 to <__automatic_adb98624_0> port
    = domain flags S/SA keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.1.4 port = www flags S/SA
    keep state
```

The first thing to note is that the "login" service was translated to port 22. A different LDAP client program might have used RDP, or "login" might have been used as a requirement in something like nsswitch.conf if we were doing LDAP Auth on a Linux box, for example. This is why we choose to interpret the service name in the app instead of in LDAP.

Next, note the weird-looking PF destination address for the DNS rule: <__automatic_adb98624_0>. This is a dynamic table that was generated for us by PF. The server object whose CN is ns.hq.dbg.com has multiple address attributes associated with it. This caused VPLDPF to generate a slew of PF rules, one per destination address for that server object. When those rules were loaded into PF, PF saw that everything but the destination address was redundant, so it optimized these rules down to a single rule by creating a table for all of ns.hq.dbg.com's destination addresses. If we wanted to see the contents of this table, we could ask PF with the command:

```
pfctl -a vpnusers/bob -t __automatic_adb98624_0 -T show
```

To make it easy to track the current state of things, I wrote another shell script that parses OpenVPN's status log for the currently connected users and runs the pfctl commands necessary to dump the PF details on each of them. This script, called vpninfo.sh and also available linked under this article at http://www.usenix.org/login/2009-10/, gives the following output for Bob:

```
################# bob ######################
localIP: 10.253.21.10, remoteIP: 67.16.87.60:13771
Connected since: Sat Jul 25 17:24:19 2009

PF Rules for user bob
pass in inet proto tcp from 10.253.21.10 to 10.21.1.2 port = ssh flags S/SA
    keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.1.2 port = https flags S/SA
    keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.1.2 port = www flags S/SA
    keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.64.101 port = https flags S/
    SA keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.64.101 port = www flags S/
    SA keep state
pass in inet proto udp from 10.253.21.10 to <__automatic_adb98624_0> port
    = domain flags S/SA keep state
pass in inet proto tcp from 10.253.21.10 to 10.21.1.4 port = www flags S/SA
    keep state
```

PF Dynamic Table Contents for user Bob:

```
#### __automatic_adb98624_0 ###
   10.21.0.1
   10.21.16.1
   10.21.32.1
   10.21.48.1
   96.26.18.66
```

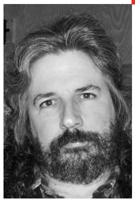
Since users each get their own set of firewall rules, we can associate every packet they send with their username by, for example, tagging their packets with their name or logging them to a special pflog interface. We can monitor and otherwise collect usage information on particular heathens with access to sensitive systems (Holt-Winters forecasting anyone?), and we get the happy side effect of encrypting all traffic in the heathen zone that's destined for privileged networks, whether the protocols in use are encrypted or not. These are the sorts of things that make auditors go all giggly. The impact on our users, most of whom were already used to using VPN from home, was pretty minimal, and I'm far happier with this than any of the real NAC solutions we've tried, though that's arguably apples and oranges.

As far as caveats go, there are two that spring to mind: First, I wish PF had an iptables-style log-prefix feature. That would make auditing far easier (time to delve into PF's source code perhaps). Second, this solution makes it somewhat tricky for systems in the privileged networks to reliably initiate connections to heathen workstations, which may or may not be a problem in your environment. We have a few people who forward their mail from the mail system to the smtp daemon on their workstation by way of a .qmail file, and we're having to find some workarounds for that. Otherwise, it's been all smiles and giggly auditors for us. Whoever's idea this was, it was not bad at all.

Take it easy.

ROBERT G. FERRELL

/dev/random: cloud computing, or, on the origin of specie



Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through.

rgferrell@gmail.com

CLOUD COMPUTING IS BASED ON THE quaint and possibly apocalyptic marketing strategy that casting your precious data out to be masticated by strangers you'll never meet in a vast cesspool of discombobulated CPU cycles with no discernible perimeter is somehow the future of digital processing; moreover, that this is a future devoutly to be wished. I take umbrage at this notion (I have amassed a great store of umbrage over the years; my wife made me move most of it up to the attic) and, in the convoluted fashion of my tribe (*Homo sapiens incoherentus*), will attempt to teach you why this is not the True Path of Digital Enlightenment. I can't tell you what the true path is, admittedly, but I'll know it when I see it. I'm not seeing it here.

In the Beginning (cue wavy flashback accompanied by *Also Sprach Zarathustra*), computing was accomplished in surplus aircraft hangers using relays and vacuum tubes with conductors so expansive you could watch the li'l electrons wend their merry way along the data path. Every so often a couple would stop and make out in the weeds behind a cathode and later a sharp-eyed observer might spot a small family of quarks trailing willy-nilly along after them. In those days you knew where your bits were at any given moment, even if you sometimes had to hail a taxi to get to them.

The next beachhead bravely to be breached was that of solid-state computing. This miraculous manifestation of miniaturization shrank the computational playing field from agriculturally significant to merely mall-sized. No longer would vast tracts of virgin old-growth forest need to be cleared in order to calculate the value of pi to the next digit. This left vast tracts of virgin old-growth forest available to be cleared for strip centers and cardboard-quality housing developments. Lo, there was much rejoicing, and the gnashing of overstuffed wallets filled the crisp morning air.

It is my considered opinion that a significant contributor to the secret underlying the spectacular success of solid-state electronics lies with the labels given to components thereof. "Zener diode" is one such example. I mean, how cool a name is that? Zener, zener, what a weener. Not only do the devices have great names, many of the terms related to solid-state electronics are masterpieces of technopoetry in their own right. “Maximum Reverse Standoff Voltage,” “Parasitic Capacitance,” and “Avalanche Breakdown” top my list. That last one sounds like a chart-busting bluegrass album.

Today the process of electronics shrinkage has reached downright ridiculous proportions. A computer that would have overflowed Yankee Stadium in the fifties now fits comfortably in the end of a ballpoint pen, with room left over for an LED flashlight. What has driven this eternal striving for smaller and faster? Well, money, of course: what else? We just keep pushing the envelope with more and more processing power in a more and more compact package because that’s the marketeer-generated perception of What the Public Wants. At some point the envelope is bound to start pushing back, and then where will we be? I’ll be out in the pool with an amber ale, if anyone cares.

Practical computing left the gate as mainframes with terminals, glided smoothly up the ramp to client-server architectures, then took a sharp, albeit brief, detour into the thin clients cul-de-sac, veering off at last into a somewhat uneasy mix of client-server and P2P. As networking grew in sophistication, we hooked increasingly greater numbers of systems together in increasingly complex recipes, with lots of spicy network appliances for color and texture. Meanwhile, the Internet was gestating in parallel, insinuating its myriad tentacles into our most secret places like some B horror movie monster. At first it was just a novelty communications tool, good for telling co-workers where to meet for lunch and for avoiding actual productivity with IRC and crude CGI scripts. Then social networking came along, and with it an incalculable number of distractions from getting anything resembling work done.

All this time, however, we still had the company-owned LAN/MAN/WAN chugging away doing the corporate computational drudge work while we sacrificed our brain cells to streaming video and “Which Pathogenic Microorganism Are You?” quizzes on Facebook. The Internet may have been costing Corporate America money indirectly due to lost productivity (assuming productivity was in the mission statement to begin with), but it wasn’t putting an active squeeze on accounts payable except as an incidental by-product of network connectivity.

I and numerous others have known (and decried loudly) from the early nineties that the Internet would eventually rule all aspects of civilization. The inflection point toward inevitability came, in my analysis, the day animated .gifs were released upon an unsuspecting and undefended public. At that instant the Web (which is really the only part of the Internet that matters to most users) became a mesmerizing place that could hold the attention of even/especially the semi-literate. Since dynamic systems tend to take the path of least resistance, semi-literacy became the baseline toward which all Web presentations tended and, in reciprocal response, that degenerate intellectual state emerged as the norm. Now we have Flash-only Web sites featuring HD video and 5.1 Surround Sound with no textual content whatever. Hypertext has gone the way of the ponderous Apatosaurus, to be replaced by the amphetaminic Apathosaurus, which feeds exclusively on twelve content-free audiovisuals per minute.

Which brings us, at last, to the Cloud. Clouds, it should be pointed out, are primarily aerial phenomena: insubstantial, ethereal, and yet capable of wreaking great havoc. Once inside one, you may have supreme difficulty finding your way out again. When you do, you may have no idea where you are or how to get back to where you wanted to be. Cloud computing is like UDP, except that it’s your entire data stream you’re entrusting to the vagaries of the meta-network, not just an odd packet or two. We've come full circle: the whole planet has become our server room and, boy, is the air-conditioning bill going to be huge this month. If you think global warming is bad now . . .

Hey, all you trend slave early-adopter-at-any-cost corporate IT departments out there: pack your mission-critical data into a bottle, toss it out into the Humboldt current, and hope it somehow finds the correct destination, is processed to your specifications, and makes it back to you someday without being nibbled on by every fish that swims by along the route. Pay us a lot of money for this privilege. Do it now, or suffer the dire consequences of . . . um . . . not doing it now.

Heck of a business model.

book reviews

ELIZABETH ZWICKY, WITH DAVE JOSEPHSEN AND BRANDON CHING

AUTOMATING SYSTEM ADMINISTRATION WITH PERL, SECOND EDITION

David N. Blank-Edelman

O'Reilly and Associates, 2009. 616 pp.
ISBN 978-0-596-00639-6

This is one of those books that make me wish I had a time machine, so I could go back and give it to my past self. I think back to the hours and hours I spent gritting my teeth and picking through all sorts of documentation, whimpering, "Why will nobody tell me just the basics of this stuff, nicely and clearly? Surely if I had a basic understanding and maybe an example, I could beat this to death with a Perl program smoothly and elegantly," and I hope that the next person stuck in this situation has a copy of this book. This book will tell you (among other things) how to subdue a log file, send mail, query a router's status or a database's contents, create or delete a user, and find out what's going on on a machine, in Perl, on a UNIX-based or Windows operating system. It's a gold mine of information on how to do intermediate system administration tasks in Perl, and it covers exactly the sort of things that are frustrating and time-consuming to figure out for yourself.

There are two classes of things it won't tell you: basics and advanced knowledge. It doesn't give you the basics of either system administration or Perl programming. If you need to know how to name your users, when you should delete a user account, or basics of TCP/IP, you'll need to go elsewhere (and it gives suggestions as to where). Similarly, you should either have written Perl programs before or be an intrepid language-learner. It does give you basics of all sorts of complex topics, like LDAP, SQL, SNMP, and XML, but not details and not advanced knowledge. The goal here is not to make you a skilled database administrator, for instance; it's to equip you to bludgeon a database into giving up its secrets in a competent and workmanlike manner. But there are plenty of references to places with more information.

Along the way, you'll pick up a lot of information about good system administration programming practice and thought patterns. There's a nice balance between instructing the reader on what a clean, elegant solution looks like and how to build it, and instructing the reader on when a dirty, clunky solution is the way to go and how to build it.

SUCCESSFUL LEADERSHIP SKILLS

Ken Lawson

Barron's, 2006. 235 pp.
ISBN 978-0-7641-3246-9

I picked this up because it looked like a non-threatening, easy-to-read introduction to "leadership," which is one of those nebulous concepts that is simultaneously completely bogus management-speak and genuinely incredibly important. And, indeed, this is relatively easy to digest, and it should at least half-way convince you that something real is being discussed. But it's not quite what I was hoping for.

Imagine a course entitled "Successful Leadership Skills"; now imagine you skip the lectures and read just the slides. Also imagine that the presenter is competent but not terribly imaginative, so the slides do not include any pictures or graphs. This is what you'd get. It's a rapid tour of mainstream thinking about leadership, presented almost entirely in numbered lists and written in high-quality business-standard prose. It is as neutral as possible, clearly trying to avoid strong points of view.

There are two situations where this book may be useful to you. First, if you want to jump-start your personal thinking about leadership, you might want to chew over a few pages at a time as a way of clarifying what you think. Since the book is not trying to be particularly persuasive and doesn't bring a lot of rhetorical devices to bear, you are left to bring your own content in. If you don't want to do that, you're unlikely to learn a lot. But if you want a framework you can fill in, it will take you a lot further than a book that's pushing a particular agenda.

Second, if you want to know what management thinks about leadership, it's a concise introduction to mainstream management beliefs. It will teach you the relevant buzzwords, what's supposed to be good, what's supposed to be bad, and how the space divides up. That helps a lot when you're talking crossculturally to management types. If you happen to know what particular philosophy your management subscribes to, you'd probably be better off researching that, but if you don't know or it's just too painful for you to read, this will give you a basic conceptual survey. And it's mostly painless.

ANDROID APPLICATION DEVELOPMENT: PROGRAMMING WITH THE GOOGLE SDK

Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike

O'Reilly, 2009. 334 pp.
ISBN 9780596521479

REVIEWED BY DAVE JOSEPHSEN

I was really excited to hear that there was an O'Reilly Android book out, and I'm equally excited to be able to say that it's exceeding my expectations at every turn.

I had my doubts when I got my hands on it. It is by no means a heavy book—just over 300 pages and yet is co-authored by four people. But true to the adage, this is no book to be judged by its cover. I found it to be well written and densely packed with well organized, clearly conveyed information.

The book has two parts, "Development Kit Walk-Through" and "Programming Topics." I read through all seven chapters of the first part in linear fashion and was introduced to Android's design and the basics of the IDE. If you've ever done mobile Java programming, you're probably aware that learning the intricacies of the development environment is at least one-third of the problem. Most of the trouble you'll have early on centers on cross-compiling, storing and retrieving resources such as strings and graphical sprites, and resolving library references—things that require familiarity with the IDE to get right (and debugging when what you get is wrong).

The authors have a great feel for what you want to know when you're getting started. At least for me, they seemed to have an uncanny knack for providing exactly the right piece of information just at the moment I started to wonder about it, and by the end of the first part I felt I had a pretty good grasp of the architecture. I also had the IDE installed on my MacBook and Linux workstation, and a "Hello World" program running in the emulator on both.

The second part of the book focuses on specific portions of the Android API. Topics include SQLite and content provider access, GUI views and widgets, and the mapping and location API. I've only read portions of the second part of the book, because the first part got me far enough to start work on porting an app I had written for Sidekick over to Android, but the chapters I have read in support of that effort are as well written and informative as those in the first part.

I could probably stop there, but there are a few things I'd like to give kudos to the authors on. The first is that they avoid the jargon-heavy language used by so many authors of Java-related books. They've made an obvious attempt to avoid using heavily Java-centric language, and as someone who only casually programs in Java (and avoids it when he can), I appreciate the effort. Second, as someone who doesn't use Windows at all, it was great to see the extra effort they put into being cross-platform, often providing details for Linux and Mac OS X as well as Windows.

This book isn't a comprehensive tome of everything you'll ever need to know about Android, but it's a fantastic primer and, as a supplement to the official Google documentation, it won't make the migration from desk to shelf anytime soon. Good work, guys.

PRO OPENSOLARIS: A NEW OPEN SOURCE OS FOR LINUX DEVELOPERS AND ADMINISTRATORS

Harry J. Foxwell and Christine Tran

Apress, 2009. 280 pp.
ISBN 978-1430218913

REVIEWED BY BRANDON CHING

Choosing a development environment for either desktop or Web-based application development is generally a trivial thought experiment. Most experienced developers have their preferences and generally don't deviate much unless a new method or tool becomes available that better fits their development needs.

The OpenSolaris operating system is attempting to be that new tool that developers will want to have around. A community-developed and -driven project based on the Solaris 10 code base, OpenSolaris is attempting to lure away the growing cadres of Linux-centric developers and administrators. Touted as a platform for both desktop application and Web development, OpenSolaris is a promising and viable alternative to Linux-based development.

In Pro OpenSolaris, Harry Foxwell and Christine Tran delve into the key features that make OpenSolaris an attractive option for developers and administrators. Topics such as the Service Management Facility, the ZFS file system, and OpenSolaris virtualization are all covered in sufficient detail. While not an exhaustive text on the topic, the book is an excellent introduction and starting point for developers not familiar with OpenSolaris.

The book is broken down into three parts, with nine chapters in all. The first part offers a general introduction to OpenSolaris, including its history, unique benefits over Linux, a walkthrough installation, and general usability coverage. For most experienced Linux developers, the first part of this book will probably not be as valuable as the remaining two. The OpenSolaris installation follows the general Linux installation process with the exception of virtualization options which are addressed in greater detail in Chapter 7. Once installed, the default GNOME desktop environment should be familiar to most.

Part 2, "Working with OpenSolaris," is where the fun really begins. In Chapter 5, Foxwell and Tran introduce the Service Management Facility (SMF) of OpenSolaris. Replacing the familiar /etc/rc* files and methods, the SMF is a service daemon that is responsible for all service management. The authors do an excellent job of introducing the SMF and its associated tools, including feature outlines, screenshots, technical details, and a number of good examples. Each aspect of the SMF is well covered, with detailed explanation and demonstrations. By the end of this chapter, you should be an expert in the SMF.

In Chapter 6, the authors dive into what I feel is the most exciting feature of OpenSolaris: the ZFS file system. As a Web developer, this was my first exposure to the ZFS and, as in the previous chapter, Foxwell and Tran do an excellent job of introducing the technology. The major features of the ZFS are massive addressable space (128 bits), active integrity checking, and, my personal favorite, "nearly unlimited and instantaneous file system snapshots" (p. 103). The authors relate the system snapshots to Apple OS X's Time Machine, but from their descriptions it seems like a more customizable implementation. As with the previous chapter, this one is also full of tool usage details, snapshots, and practical examples.

In Chapter 7, "OpenSolaris and Virtualization," Foxwell and Tran present probably the strongest chapter of the book. Over 40 pages, the authors provide great detail, background, and examples utilizing OpenSolaris-specific virtualization methodologies. Opening with a great general introduction to virtualization, the authors proceed to extensive coverage of OpenSolaris specific zones and zone management, followed by an introduction to the xVM hypervisor. Zones are incredibly powerful ways of managing applications, and the authors stress the use of zones throughout the book.

Part 3 takes you through setting up a development environment in OpenSolaris and an introduction to a few more OpenSolaris-specific features. Chapter 8 walks you through the installation of an Apache, MySQL, PHP (AMP) zone. While a seemingly trivial exercise to Linux natives, the authors cover OpenSolaris-specific considerations, including package management, service administration, and default file locations. The chapter closes with an introduction to the NetBeans IDE and integration of other third-party tools and products such as Subversion.

In the book's final chapter, Foxwell and Tran seem to touch on a number of remaining OpenSolaris tools and features that just don't fit anywhere else. Coverage in this chapter includes DTrace for system analysis, the Tracker utility for metadata file searching, and a few other resources for entertainment and educational pursuits.

Overall, Pro OpenSolaris is a great introduction to the features and tools offered by OpenSolaris. While probably not the most complete guide to implementation in OpenSolaris, the book is definitely of value to both Linux desktop and Web developers, as well as system administrators and information managers. The writing style is technical yet approachable and connects topics nicely. As I mentioned, the chapters on ZFS and virtualization are incredibly strong, and I would certainly recommend this book to anyone interested in a Linux alternative with cutting-edge features and an active community base.

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

FREE SUBSCRIPTION to *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, networks, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/.

DISCOUNTS on registration fees for all USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals: www.usenix.org/membership/specialdisc.html.

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to board@usenix.org.

NOMINATING COMMITTEE FOR 2010 USENIX BOARD OF DIRECTORS

The biennial elections of the USENIX Board of Directors will be held in early 2010. The USENIX Board has appointed Rémy Evard to serve as chair of the Nominating Committee. The composition of this committee and instructions on how to nominate individuals will be sent to USENIX members electronically and published on the USENIX Web site this fall.

SUMMARY OF USENIX BOARD OF DIRECTORS ACTIONS

Ellie Young, Executive Director

Below are some of the actions taken by the USENIX Association Board of Directors in the past year.

Awards

The Best Paper Award at OSDI was renamed in memory of Jay Lepreau.

USENIX is now seeking nominations for its 2009 SAGE Outstanding Achievement Award: please send suggestions to sageawards@usenix.org. We also seek input on the FLAME and STUG awards for 2010: send your suggestions to awards@usenix.org.

Finances

A deficit of $322K was incurred in the USENIX operating budget for 2008, the result of lower revenue from training and conferences in the latter half of the year. The budget for 2009 projects a deficit of $500K overall, in anticipation of reduced revenue from membership and conferences and lower gains on investments.

Member dues were raised by $5 in all categories except corporate membership. Registration fees for conferences and workshops were raised by $10–$15, to partially cover increased direct expenses.

The Board expressed the need to closely monitor the downturn in enrollment in training at USENIX events and to develop new ideas/services/conferences that will increase revenue and expand our outreach.

For cost savings, the following actions were taken: (1) standards activities were suspended in early 2009; (2) some recently hired USENIX staff were laid off in early 2009; (3) remuneration for tutorial instructors was reduced slightly. The USENIX Board expressed their thanks to the remaining staff for working "smart and hard."

Student Grants Program

Program chairs are being encouraged to seek funding from the NSF for support of students to attend conferences and are being asked to help the staff find additional corporate sponsors.

In response to the huge increase in the number of students applying for grants and the reduction in corporate sponsorship this year, USENIX allocated $40K to fund students to attend the 2009 USENIX Security Symposium.

USACO

In line with USENIX's desire to continue support of K–12 computer science–related activities, USENIX continued its support for the USA Computing Olympiad.

Revamp of the USENIX Office Systems

The USENIX Board allocated substantial funds for 2009 and 2010 to revamp the USENIX Web site—including a CMS system and a new design for logos as well as for the site itself—a new event registration system, and a new membership database.

Going Green

It was decided that USENIX would move toward eliminating print for proceedings publication. Conference proceedings continue to be published: that is, they are assigned ISBNs; they are typeset with page numbers and running feet; and the title page, table of contents, and other frontmatter are created and made accessible online. The files are available online by session for attendees before the event (with the exception of any papers that have to be kept private until the event). Once the technical sessions have begun, the complete proceedings are available to everyone, both as a tarball and as individual session files.

Currently, USENIX provides the files on reusable flash drives as an option for attendees. Tutorial materials at LISA '09 will also be available on flash drives. We are looking into providing materials in Kindle format as well.

Conferences

To address a number of questions and issues that have arisen recently, USENIX has developed new guidelines and clarification of the responsibilities of all event organizers: program chairs, steering committees, USENIX Board liaisons, and USENIX staff. The Board approved a clarification of the statement concerning the confidentiality of paper submissions.

In an attempt to identify and acknowledge outstanding papers from USENIX events, program chairs will be encouraged to approach suitable journal editors about inviting select paper authors to submit an expanded version to their journal.

Adam Moskowitz was appointed to serve as program chair for LISA '09. A steering committee was formed to make recommendations about format, direction, and guidance for the future.

OSDI '08 had the highest attendance ever, and the number of co-located workshops doubled. Remzi Arpaci-Dusseau and Brad Chen were appointed to serve as program co-chairs in 2010.

NSDI '09 also had its highest attendance, at 254. Miguel Castro and Alex Snoeren were appointed as program co-chairs for 2010.

FAST '09 had a slight dip in attendance, explicable mainly by reduced corporate travel budgets. Participation in training, the TaPP workshop, and the OpenSolaris Summit was excellent. Kim Keeton and Randal Burns were appointed program co-chairs for 2010.

A proposal for a workshop on Sustainable Information Technology to be co-located with FAST in 2010 was accepted. Erez Zadok and Ethan Miller will serve as co-chairs.

The first Hot Topics in Parallelism workshop was held in March, drawing 92 attendees. David Patterson and Geoff Lowney were appointed as co-chairs for a second workshop in 2010, to be held on the Berkeley campus of the University of California.

After a successful HotOS XII in May, chaired by Armando Fox, Matt Welsh was invited to serve as program chair for the 2011 workshop.

The USENIX Annual Technical Conference will be reconfigured and rebranded in 2010. Co-located events such as the USENIX Annual Tech Refereed Papers, the new USENIX Conference on Web Application Development, HotCloud '10, and a workshop on online social networking are but a few of the exciting programs USENIX will be offering during this third week in June in Boston. The traditional tutorial program will be eliminated. USENIX is soliciting additional topics of interest to developers and programmers. Please contact ellie@usenix.org if you have additional ideas for that week.

The 18th USENIX Security Symposium and co-located workshops, held in Montreal in August, had excellent attendance and were very well received. Ian Goldberg was appointed program chair for USENIX Security in 2010 and David Wagner will chair in 2011. Program co-chairs for EVT/WOTE '10 were approved as well.

SAGE

Three Short Topics booklets were published in 2008. We expect to publish two in 2009: a booklet on monitoring environment, networks, and systems, and a heavily revised Jobs Descriptions booklet, which will contain, for the first time, a Management series of job descriptions. Several Short Topics booklets are now available on Safari Books Online.

In April 2009, the USENIX Board approved the settlement proposal regarding the lawsuits between AH, Inc., LOPSA, and USENIX.

Policy

In response to a general recommendation from the auditors, the Board of Directors approved an official policy on gifts from vendors.

USENIX adopted a policy concerning retention of electronic communications.

Member Benefits

All videos from USENIX events are now immediately available to all USENIX members, as well as to attendees of the event.

Next Meeting

The next in-person USENIX Board of Directors meeting will be held November 2, 2009, in Baltimore, MD, during LISA '09.

conference reports

THANKS TO OUR SUMMARIZERS

2009 USENIX Annual Technical Conference

San Diego, CA
June 14–19, 2009

OPENING REMARKS

Summarized by Rik Farrow

After thanking the program committee and the USENIX staff, co-chairs Geoffrey M. Voelker and Alec Wolman announced the Best Paper awards: "Satori, Enlightened Page Sharing" by Grzegorz Miłoś, Derek G. Murray, Steven Hand, and Michael A. Fetterman, and "Tolerating File-System Mistakes with EnvyFS," by Lakshmi N. Bairavasundaram, Swaminathan Sundararaman, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Next, Alva Couch, Secretary of the USENIX Board of Directors, presented the Software Tools User Group Award to Jean-Loup Gailly and Mark Adler for their work on file compression (see http://www.usenix.org/about/stug.html for details). Gailly said, "I feel I have received more from the OS community than I gave," and Adler then said "Ditto," in a couple of the shortest acceptance speeches ever.

Couch then presented the Lifetime Achievement Award to the late Professor Gerald J. Popek (you can read more at http://www.usenix.org/about/flame.html). Two of his past students, Bruce Waller of HP Labs and Geoff Kuenning of Harvey Mudd College, described Popek's long history in CS research, with many contributions in systems, including the first mention of clusters. Both men also spoke of Popek's dedication to his students. Kuenning explained that Popek had taken a long leave from academia to start Locus Computing, which is why his name disappeared from publications sometime during the '90s.

KEYNOTE ADDRESS

- ***Where Does the Power Go in High-Scale Data Centers?***
  James Hamilton, VP & Distinguished Engineer, Amazon Web Services

  Summarized by Stephen P. Tarzia (starzia@northwestern.edu)

James Hamilton gave a fresh appraisal of electrical power's role as a primary design consideration in data centers. The fundamental issue is that high-scale data centers such as those managed by Amazon and Google are very different from conventional enterprise data centers. Due to management complexity introduced by heterogeneity, people costs dominate the total enterprise datacenter costs. By contrast, a high-scale data center typically has more than 1000 servers per administrator, so people costs are almost negligible. In this keynote, Hamilton outlined the true costs of such data centers as well as their engineering implications.

Hamilton gave a total cost analysis for operating a theoretical 15 megawatt high-scale data center. He showed that servers accounted for 53% of costs, power and cooling infrastructure for 23%, and power usage for 19%. Since server prices are falling, he forecast power-related costs accounting for over half of total costs in the future. However, it is important to note that the majority of power-related costs are due to infrastructure, not utility charges.

To drive cost-cutting efforts, Hamilton advocated measuring Total Power Usage Efficiency (tPUE), the ratio of total facility power to power delivered to server components. His blog, in particular the entry at http://perspectives.mvdirona.com/2009/06/15/PUEAndTotalPowerUsageEfficiencyTPUE.aspx, has more details on tPUE. This measure differs from the traditional metric, PUE, in that it includes energy waste within IT equipment. In particular, motherboard voltage regulation circuits and case fans are often unnecessarily inefficient. He showed all of the steps in the power distribution chain, which has over 90% end-to-end efficiency.

To get the maximum return from the data center's power and cooling infrastructure investment, the operator must run as many servers on top of that infrastructure as possible without overloading it during peak periods. To achieve that, Hamilton suggested a combination of both cooling and server-utilization optimizations.

Regarding cooling, Hamilton first promoted isolating hot and cool air flows and running data centers at much higher temperatures. Hamilton showed that popular server warranties typically cover equipment that is run at up to 95°F, much hotter than a typical data center. Based on this observation, Hamilton proposed using outdoor air instead of AC for cooling. Some worry that airborne particles from outdoors might damage IT equipment, so detailed studies are needed to test this and to evaluate filtration techniques.

Finally, Hamilton discussed resource consumption shaping. This means reducing peak load at the expense of increased trough load. In other words, smooth out the load curve by pushing some of the peak workload into idle times. He suggested following the airline industry's model of overbooking and then shedding excess load when necessary to maximize capacity utilization. Hamilton also suggested using the same load-smoothing approach with links. Further gains can be had by increasing average server utilization, a figure that is typically only around 15%.

Rik Farrow asked why datacenter operators don't have servers custom-built to work most efficiently in their facility. Hamilton responded that the big datacenter operators do work closely with custom design groups in computer manufacturing companies. He also mentioned that big-impeller fans and shared power supplies are typical requests. Can server traffic can be pushed by hours, since this is what would be needed to smooth out daily user cycles? There is lots of work to be done at night, in particular data analysis and data mining. Still, Hamilton acknowledged that operators will have to pay for peak-time responsiveness.

Is humidity an issue in the data center at higher temperatures? Everyone fears humidity, but concrete data is lacking. How do networking costs figure into the total and will ISPs change their pricing model if link utilization increases? WAN costs were not included in Hamilton's analysis, but they are minor: only a few percent. Hamilton was not prepared to comment on ISP pricing. David Petrow asked about the role of server water cooling now and in the future. Hamilton observed that the industry loves density, while floor space costs are negligible, so water cooling is unnecessary. When asked by Dan Klein how to shed light on the right people to promote his agenda, Hamilton suggested focusing on those with the biggest R&D budget.

The final two questions returned to server utilization. When asked for an example of software inefficiency, Hamilton noted that some software inefficiency must be tolerated, such as using high-level languages to increase developer productivity and thus drive innovation. Someone asked how tPUE included the actual work done. Hamilton acknowledged that it does not, but it is valuable since it is generalizable across different applications and industries. He recommended additional industry-specific calculation of work done per dollar.

VIRTUALIZATION

Summarized by John McCullough (jmccullo@cs.ucsd.edu)

- ***Satori: Enlightened Page Sharing***
  Grzegorz Miłoś, Derek G. Murray, and Steven Hand, University of Cambridge Computer Laboratory; Michael A. Fetterman, NVIDIA Corporation

  Awarded Best Paper!

Miłoś described a system to leverage page sharing without the overheads of VMM page scanning. Memory can be one of the most limited resources in virtual machines, and in the common situation of homogeneous virtual machines there can be a large amount of redundant data. Typical approaches involve the VMM scanning all pages, creating fingerprints, and then initiating page sharing. This is a heavyweight operation whose periodicity is limited. Miłoś observed that many shared pieces of data arise from I/O devices and that by instrumenting the virtual I/O devices we can capture that page sharing and avoid periodic scanning. An additional benefit of this approach is that when using copy-on-write disk images, VMs can bypass the disk read and share the data if it is resident in memory elsewhere. Satori implements I/O-based page sharing behavior in the Xen hypervisor.

While typical page sharing approaches release pages into a global pool, Satori credits fractions of the freed pages to the VMs participating in the sharing. These credits can be taken from a type of inverted-balloon driver, but to prepare for share-breaking the VM must maintain a list of volatile pages that can be evicted at any time. These pages can typically be used for additional page cache. The system performs well in general, with less than 1% overhead for random reads and meta-benchmarks. However, for sequential reads there is a 35% slowdown due to hypercall overheads. Miłoś believes this overhead can be alleviated through a shared memory approach. Satori outperforms VMware's fastest—yet still infrequent—similarity scans, even though Satori cannot perform sharing for pages not loaded through I/O, including the kernel, which is pre-loaded by the hypervisor.

An audience member asked whether the VMs can steal memory from other machines by reading extra shared data. Miłoś responded that the machines cannot gain any additional memory this way. Do the costs of sharing and detection outweigh the potentially short duration of the potential sharing? The aggregate of the short-lived sharing opportunities can still provide a great benefit. Does it make more sense to explicitly share the page cache? The goal is to provide the benefits of page sharing with minimal modification to the guest operating system. Transcendent memory implements the shared page-cache behavior.

- ***vNUMA: A Virtual Shared-Memory Multiprocessor***
  Matthew Chapman, The University of New South Wales and NICTA; Gernot Heiser, The University of New South Wales, NICTA, and Open Kernel Labs

Chapman observed that when you need more computational power than a single processor, you typically turn to a shared memory multiprocessor or a cluster of workstations. Large shared memory multiprocessing systems are often very expensive, and workstation clusters are often awkward to program. Typical approaches that join a workstation cluster into a single machine image use language-specific middleware or narrowly supported distributed operating systems. Chapman proposed vNUMA, where a virtual machine monitor presents an unmodified operating system with a single machine image spanning a workstation cluster.

vNUMA addresses a number of challenges in faithfully reproducing the SMP programming environment. Unlike many distributed shared memory systems, all data in an SMP system is shared, including locks, and read-modify-write and memory-fence behavior must be respected. Because no particular write invalidation technique is performant across all access patterns, vNUMA uses an adaptive protocol selecting among three approaches for particular memory pages. In one update case, trap-emulation is required, but a simple write-invalidate is used in most cases. Chapman showed that vNUMA can out-perform the distributed shared memory library, Treadmarks, across compute-intensive HPC benchmarks and that vNUMA performs comparably to distcc for compilation. However, for I/O intensive database workloads, vNUMA performs poorly. Overall, vNUMA provides a single system image for free with good computation performance.

An audience member asked how devices are handled. Chapman responded that the work focused primarily on memory behavior rather than devices. In the current implementation, network and disk I/O are routed through node zero. Future work could introduce striping across nodes and improve performance.

- ***ShadowNet: A Platform for Rapid and Safe Network Evolution***
  Xu Chen and Z. Morley Mao, University of Michigan; Jacobus Van der Merwe, AT&T Labs—Research

Chen observed that alternative configurations on carrier-grade networks can have negative effects. However, existing modeling and emulation testbeds cannot get the same fidelity and hardware implementation as the production network. Chen proposed ShadowNet, a system that provides a network that is connected to but separate from the production network. This provides an environment in between the lab and the production environment and allows multiple service trials to run simultaneously, sharing the same physical resources but in isolation.

ShadowNet is implemented on top of Juniper-based virtual routers, which are, in turn, attached to a ShadowNet node hosting virtual machines. Each virtual router provides the full functionality of the original routers, while connected to each other and VM instances via a variety of connectivity options, keeping traffic isolated and routing updates regulated. At the experimental level, ShadowNet provides configuration management across experimental configurations. Chen demonstrated that ShadowNet is able to get the desired bandwidth allocations, although the virtual routers have some interaction with the other routing elements under high load. Chen also demonstrated that ShadowNet can achieve failover to an alternate configuration.

An audience member observed that in PlanetLab-style deployments it is very easy to add nodes and asked how feasible it is to add new nodes in ShadowNet. Chen noted that the controller takes care of adding the nodes and that they should be easy to add. Do any cloud vendors provide similar systems? Most cloud infrastructures only provide the virtual hosts, and ShadowNet has richer networking support.

INVITED TALK

- ***Teaching Computer Science in the Cloud***
  David J. Malan, Harvard University

Summarized by Matthew Renzelmann (mjr@cs.wisc.edu)

Professor David Malan presented his work on reinvigorating Harvard's introductory computer science course, CS 50, in an effort to increase enrollment in the university's computer science program. Enrollment figures for the last decade and a half showed a significant decline after the dot-com bubble burst in early 2000. Malan suspects that this decline stemmed from misconceptions about computer science and the relatively uninteresting nature of many introductory programming projects (e.g., writing programs with a command-line interface vs. a GUI).

To make the course more interesting, Malan discussed using more languages than just C (e.g., PHP) and providing students with frameworks to write more sophisticated programs. These frameworks also serve to acquaint students with reading code. In addition, Malan emphasized the importance of assigning programming projects that solve more interesting problems, such as implementing a Vigenère cipher with arrays or a competition to come up with the fastest spelling checker.

After outlining his approach to teaching the course, Malan began discussing the role of cloud computing. Malan's goal was to acquire a set of machines with unfettered root access, which he could then configure for the students in the course. Although his group examined the possibility of operating their own cluster, they concluded that because of limited space, power, and cooling, it would be easier to offload everything to Amazon's EC2 service. In Malan's experience, launching a group of virtual machines on EC2 was much easier than setting up the infrastructure themselves.

Observed benefits of using Amazon's EC2 cloud infrastructure for course work were numerous. The number of virtual machines assigned to the cloud was scalable, and it was easy to start additional virtual machines during periods of high activity, such as the night before an assignment was due. The students found that using the virtual machines was straightforward because access was available through the host name cloud.cs50.net. This single host would pseudo-randomly assign each user to one of the cloud's virtual machines.

Using Amazon's EC2 also involved some costs. Malan estimated a cost of $15/student for the semester, or $5000 in all, but believed that additional work on his part could drive this cost down to $2000–$2500. Bandwidth was a particular concern, because it can be expensive. Learning EC2's idiosyncrasies was also troublesome; in the past, the department's IT staff took care of infrastructure issues, but with EC2, the onus was on Malan and his staff to keep things running smoothly.

One audience member asked whether the term "sprites" was a spoof or pun after Osterhout's Sprite research. The question was in reference to Malan's use of MIT's Scratch programming environment, which used objects called sprites. Malan replied that the name was entirely courtesy of MIT's Media Lab. Someone else pointed out that Malan's results showed an increase in course enrollment during the first week, but it wasn't clear whether these students were doing any better later in the course. Malan responded that there was not yet enough data to answer definitively, but that there has been an uptick in the number of students selecting computer science as a major.

NETWORKING

Summarized by John McCullough (jmccullo@cs.ucsd.edu)

- ***Design and Implementation of TCP Data Probes for Reliable and Metric-Rich Network Path Monitoring***
  Xiapu Luo, Edmond W.W. Chan, and Rocky K.C. Chang, The Hong Kong Polytechnic University, Hong Kong

Luo observed that Internet measurement can be very challenging. ICMP packets frequently have different behavior and can only measure limited metrics. He introduced OneProbe, which enables the measurement of TCP-based applications' specific behavior and can capture RTT, directional packet loss, and packet reordering. The current incarnation operates over HTTP, and the approach should be extensible to additional TCP-based applications.

OneProbe operates using a pair of probing packets. By observing the sequence and acknowledgment numbers in TCP packets and distinguishing TCP data packets and TCP control packets through packets' payload size, the responses can be classified into one of 18 cases to determine reordering and loss on both forward path and reverse path. Thus, OneProbe is able to achieve more expressive measurements against almost any Web server and provide more accurate results than httping. Luo showed results of latency measurements for the Web servers of the 2008 Olympic Games: they were able to observe diurnal RTT and loss behavior, and a significant difference between OneProbe and ICMP echo result on some paths. See http://www.oneprobe.org for more information.

An audience member asked how the RTT tests can be accurate for forward and reverse paths while using TCP. Another audience member inquired about the requirements on the application protocol. Luo answered that servers must send back some data packets and that clients need to be able to send data back to the server.

- ***StrobeLight: Lightweight Availability Mapping and Anomaly Detection***
  James W. Mickens, John R. Douceur, and William J. Bolosky, Microsoft Research; Brian D. Noble, University of Michigan

Mickens observed that we typically like to know the status of hosts in our networks. This can sometimes be achieved using distributed systems or other monitoring mechanisms, but it requires host modifications and can sometimes lead to scalability concerns. Mickens introduced StrobeLight, a system targeted to measuring networks of a few hundred thousand hosts. StrobeLight simply sends ICMP probes to every host on the network every 30 seconds, providing fine-grained fingerprints for availability data. This data can be used to guide choices in building multicast trees, task allocation, or identifying misbehaving networks or network hosts.

StrobeLight was designed to be simple and unintrusive without requiring infinite scaling. It operates by extracting the list of hosts from the DNS server and pinging each of them. The availability data is used to construct a per-subnet bit-vector where each position represents the availability of each host. Using a similarity metric, Mickens shows that most subnets do not change in character over time unless there is an anomaly. In one case, this assisted in identifying subnets that lost connectivity, and it can be used in general to identify routing anomalies such as BGP hijacking. Using an external wide-area prober, Mickens demonstrates that the fingerprints are generally similar and that in simulation they were effective in identifying hijacking attempts.

An audience member asked whether the hijacker could prevent detection by mimicking the host availability of the target network. Mickens replied that you'd be relatively powerless if the attacker could duplicate the availability profile, but that access to the network availability can be restricted to designated probers. Overall, the task is challenging, but StrobeLight is a simple first cut.

- ***Hashing Round-down Prefixes for Rapid Packet Classification***
  Fong Pong, Broadcom Corp.; Nian-Feng Tzeng, Center for Advanced Computer Studies, University of Louisiana at Lafayette

Pong established the need for fast packet classification that is both dynamic and compact. Typical approaches use either decision trees or hash tables. Decision trees can be tall and take a while to traverse, and the addition or deletion of a rule can necessitate a reconstruction of the entire tree. Hash-table approaches often require many probes to determine the correct prefix length and, in some cases, use supplementary decision trees to select the correct prefix length. Instead of storing a single address and mask pair at each entry in the hash table, HaRP (hashing round-down prefixes) lumps groups of prefixes into sets that can be searched in parallel. This reduces the number of hash lookups and reduces memory requirements.

Because HaRP probes all prefix-group buckets and because prefixes are transitive, it is possible to load-balance the hash table by placing shorter prefixes in longer buckets. HaRP also allows for either source IP hashing or destination IP hashing. Pong demonstrated results for six rule sets: three from practice and three artificially enlarged. Although HaRP does not hash-load-balance quite as well with many short prefixes, it enables a compact representation that can fit in cache and overall achieves approximately a 5x speedup. Data structure updates are much quicker than the several minutes that can be required for a large decision tree.

An audience member questioned the applicability of rapidly changing firewall rules. Pong pointed out that in VPN settings there can be frequent creation and deletion of firewall rules as clients enter and leave the system. Does the order of rule insertion affect the layout and hash-table load? Their first cut of choosing the first fit has worked well, and he also notes that finding the optimal fit is an NP problem.

FILE AND STORAGE SYSTEMS

Summarized by Alex Rasmussen (alexras@acm.org)

- ***Tolerating File-System Mistakes with EnvyFS***
  Lakshmi N. Bairavasundaram, NetApp, Inc.; Swaminathan Sundararaman, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

  Awarded Best Paper!

Swaminathan Sundararaman presented EnvyFS, a file system based on N-version programming that is designed to tolerate silent file system failures. Sundararaman argued that modern file systems are very complex and that this complexity, combined with the increasing depth of the stack between the file system and the disk (imposed by virtualization, networked file systems, etc.), admits the possibility of a wide range of failures, including so-called "fail-silent" failures in which the file system doesn't detect an error and continues to work on the wrong data, causing data corruption and returning bad data to the user. EnvyFS copes with fail-silent failures through N-version programming, which involves executing several different functionally equivalent programs (in this case, file systems) and using majority consensus to agree on the federated program's output. To reduce the storage and performance overheads imposed by using several child file systems at once, the authors created a single-instance store called SubSIST that de-duplicates data while retaining most of the reliability benefits the N-version file system provides.

Sundararaman presented several examples of individual fail-silent errors that resulted in corruption in the ext3 file system, but that EnvyFS (using ext3, JFS, and ReiserFS as its child file systems) is able to tolerate. In one case, EnvyFS masked an ext3 error that would otherwise have caused a kernel panic.

One audience member wondered whether EnvyFS assumes that all file systems have the same block size and how EnvyFS would deal with an extent-oriented file system. Sundararaman replied that EnvyFS assumes 4 KB blocks and that, if extent-oriented file systems wrote at non-block-aligned offsets, a more sophisticated scheme such as fingerprinting would have to be used to identify duplicate blocks. Had the file system authors had been informed of the bugs uncovered during the evaluation? Yes, they had. Did correlated failures relate to file systems having copied code from one another? They had not noticed any instances of code copying, but different file systems can be chosen to minimize the presence of duplicate code if such code is observed.

- ***Decentralized Deduplication in SAN Cluster File Systems***
  Austin T. Clements, MIT CSAIL; Irfan Ahmad, Murali Vilayannur, and Jinyuan Li, VMware, Inc.

Austin Clements presented a new method of de-duplication in storage area networks (SANs). De-duplication prevents duplicate data from being stored on disk by tracking the locations of written blocks in an index and bypassing writes to disk if the block to be written is already in the index. Clements asserted that classical methods of de-duplication do not work well in a decentralized setting due to cache coherence problems, the need for coordinated allocation of disk space for new blocks, loss of disk locality on individual disks, and a shared index structure to which access must be coordinated using locks.

To solve these problems, the authors have developed DeDe, which breaks de-duplication into three stages: write monitoring, local de-duplication, and cross-host de-duplication. DeDe can de-duplicate live storage devices out-of-band and in large batches. The system is designed to minimize contention on the shared index and communication between hosts, is resilient to stale index information, and improves access to unique blocks by allowing them to be mutable and to remain sequential on disk.

The authors evaluated DeDe on a corporate virtual desktop infrastructure and found that it was able to compress 1.3 TB of non-zero data to 237 GB while using only 2.7 GB of disk space for its data structures and causing no additional I/O overhead.

An audience member wanted to know how effective DeDe would be if de-duplication were done at the file level as opposed to the block level. DeDe (and de-duplication in general) would not be as effective in this case, since the opportunity for savings decreases as the size of the unit of replication increases. Might DeDe cause fragmentation and interfere with linear read-ahead? While any de-duplication system has these issues to some extent, DeDe suffers less from these problems, because it keeps blocks in their sequential location on disk whenever possible and permits in-place updates to blocks without duplicates. Would certain kinds of access patterns lead to poor performance with a de-duplication system that uses fixed-size chunking, as DeDe does? Many systems in this space use variable-size Rabin fingerprinting to overcome this issue, but Clements speculated that such fingerprinting is unlikely to be worth the performance penalty of managing variable-size blocks in a live, shared file system. What might happen when a lot of duplicate data is injected into the system suddenly, such as when all VMs in the network are patched in rapid succession? Increased duplication would trigger de-duplication more frequently, but such temporary increases in duplicate data are hard to deal with in general and some extra storage space must be allocated to deal with this eventuality.

- ***FlexFS: A Flexible Flash File System for MLC NAND Flash Memory***
  Sungjin Lee, Keonsoo Ha, Kangwon Zhang, and Jihong Kim, Seoul National University, Korea; Junghwan Kim, Samsung Electronics, Korea

Today's NAND flash memory comes in two main varieties: SLC (single-level cell) and MLC (multi-level cell). SLC has higher performance and lasts longer than MLC, but MLC has higher capacity. However, MLC flash memory can be programmed dynamically as either MLC or SLC through use of a special writing method. Sungjin Lee described FlexFS, a new file system that combines the performance of SLC flash with the capacity of MLC flash. It does this by managing disk blocks as three separate pools for SLC, MLC, and free blocks. Blocks are dynamically allocated and migrated between regions in the background. New data is written to the SLC region and blocks are migrated to the MLC region in the background as the SLC region becomes full. FlexFS also takes advantage of idle time to generate free blocks for the SLC region and avoids migrating "hot" (recently referenced) pages. In addition, FlexFS's wear manager controls the rate at which erase operations occur, to maximize the device's lifetime.

An audience member wanted to know if FlexFS, which was targeted at mobile systems, could be applied to large disks, where capacity is less of an issue, and to environments where the system could take advantage of write caching. Lee responded that FlexFS could certainly be extended to support such environments.

- ***Layering in Provenance Systems***
  Kiran-Kumar Muniswamy-Reddy, Uri Braun, David A. Holland, Peter Macko, Diana Maclean, Daniel Margo, Margo Seltzer, and Robin Smogor, Harvard School of Engineering and Applied Sciences

Provenance is metadata that describes the history of an object. Such data is useful for scientific reproducibility, business compliance, and security. Previously, the authors constructed PASS, which observes file system calls to infer relationships between objects. However, if an application such as a Web browser also tracks provenance, no method exists to link the provenance tracked by the application with that tracked by the kernel. Kiran-Kumar Muniswamy-Reddy discussed the Disclosed Provenance API (DPAPI), through which software that tracks provenance can disclose that provenance to lower layers of the software stack in a secure, modular way. DPAPI can pass abstract provenance-containing objects between programs through use of opaque handles and has functions to associate provenance with reads and writes, thus ensuring that provenance is consistent with the data it describes. Muniswamy-Reddy then described the use of DPAPI in Kepler (a provenance-aware workflow engine) that links the Web browser and the Python interpreter. He concluded with some lessons learned by the authors in writing DPAPI. Among these lessons learned are that it is not easy to make applications provenance-aware and that making platforms provenance-aware does not necessarily provide provenance awareness to all applications running on that platform.

One audience member wondered whether there was some notion of nested provenance, where, for example, each tab tracks its provenance and the browser tracks the "meta-provenance" of the collection of tabs. Muniswamy-Reddy replied that the browser knows where each URL came from and the chain of URLs the user viewed in the past and so can do some logical separation within itself, but that the clarity of the separation isn't clear. He mentioned that they are looking at Google Chrome as a better target for DPAPI than Firefox, due to Chrome's use of a separate process per tab. Why is the file system the right location at which to focus provenance tracking, since the application is so much more aware of what is actually being done at a high level? The file system is a single point with which all processes must eventually interact and, while its provenance is incomplete, is a piece of the larger provenance puzzle; DPAPI helps to integrate it with the rest of the software stack.

INVITED TALK

- ***Project SunSPOT***
  Roger Meike, Sun Microsystems

Summarized by Alex Rasmussen (arasmuss@cs.ucsd.edu)

Roger Meike from Sun Labs provided an overview of SunSPOT, Sun's research and development platform for studying the entire embedded systems software and hardware stack as part of its Internet of Things Actualized initiative. SunSPOTs run Java, come equipped with a variety of sensors and affectors, and can communicate with one another wirelessly. Additionally, the SunSPOT platform comes with a large number of libraries, and the device itself is modular, allowing users to install custom or preconfigured sensor boards to fit their application.

The majority of the talk focused on projects that have used SunSPOTs. Meike gave examples ranging over several domains, including toys, research-oriented sensor networks for environmental monitoring, autonomous robots, and art installations. He also discussed some work that has been done to build a community around the SunSPOT platform; Meike believes that allowing the SunSPOT community to largely integrate itself into existing social networks (through use of the #spaught Twitter tag, for example), rather than creating a domain-specific social network, has helped the community grow beyond a core group of SunSPOT enthusiasts.

When asked about the weirdest thing anyone has ever done with a SunSPOT, Meike replied that they once taught a SunSPOT Morse code and created a translator that would receive Morse code sent wirelessly by one SunSPOT and translate it into semaphore.

Meike provided a number of pointers to more information about the SunSPOT platform. sensor.network.com provides information about various SunSPOT installations. See http://sunspotworld.com and http://spots.dev.java.net for more information about the platform and existing applications.

POSTER SESSION

Summarized by Chris Frost (chris@frostnet.net) and Rik Farrow (rik@usenix.org)

- ***SPROV: A Library for Secure Provenance***
  Ragib Hasan, University of Illinois at Urbana-Champaign; Radu Sion, Stony Brook University; Marianne Winslett, University of Illinois at Urbana-Champaign

SPROV, an application-layer library for secure provenance, intercepts file-system system calls and logs file modifications and other lineage information. Cryptographic components of the provenance chain allow verification of the integrity of these provenance records at any point in the future. This capability allows one to verify the edit history of a document. For most common real-life workloads, SPROV imposes runtime overheads of 1–13%. For more information see http://www.usenix.org/publications/login/2009-06/openpdfs/hasan.pdf and http://tinyurl.com/secprov.

- ***Towards a Formally Verifiable Multiprocessor Microkernel***
  Michael von Tessin, NICTA, University of New South Wales

Michael von Tessin presented his work on formal verification of the seL4 microkernel. He is working to extend the completed proofs for the uniprocessor version and make them work in a multiprocessor setup. To reduce complexity introduced by concurrency, he identified two orthogonal approaches: The first is to use one big lock around the kernel to reduce parallelism. The second is to reduce sharing by having a multikernel architecture.

- ***Sonar-Based Measurement of User Attention***
  Stephen P. Tarzia, Northwestern University; Robert P. Dick, University of Michigan/Northwestern University; Peter A. Dinda and Gokhan Memik, Northwestern University

Stephen Tarzia explained how they are using sonar to monitor user activity. Unlike typical activity monitors, such as mouse or keyboard event monitoring, sonar can detect if there is a person sitting in front of a keyboard. The sonar data is easy to analyze and will be used in power management, such as screen dimming.

- ***Including the Network View in Application Response Time Diagnostics using Netflow***
  Jochen Kögel, University of Stuttgart

Jochen Kögel presented a use of router-provided flow-level data, Netflow, to diagnose network issues and their impact on application response time in global enterprise networks. Today Netflow data is only used for reporting, accounting, and security, in part because of its incompleteness caused by hardware logging limitations. However, Jochen showed how network round-trip times can be separated from server response times, how packet loss can be traced to particular network segments, and how one-way network delays can be measured with Netflow data.

- ***Dynamic Resource Management Through Transparent Interaction Monitoring***
  Igor Crk, Mingsong Bi, and Chris Gniady, University of Arizona

In this work, the presenters include context while monitoring user behavior. Simply monitoring user events, such as mouse and keyboard events, doesn't provide enough information to predict when a hard drive or network interface should be suspended or awakened, or the CPU run at a slower clock rate. Using the mouse to open a File dialog suggests that the hard disk should be spun up in anticipation of a read or write. This work builds on their 2008 USENIX Annual Technical Conference paper (http://www.usenix.org/event/usenix08/tech/full_papers/crk/crk_html/index.html).

- ***Software Configuration by the Masses***
  Wei Zheng, Ricardo Bianchini, and Thu D. Nguyen, Rutgers University

This poster focused on early research trying to help new users configure our increasingly flexible software systems. They plan to collect existing users' configurations and the corresponding effects (i.e., performance metrics) to automatically recommend configurations for new deployments. They hope this will help new deployments by determining the sequence of configurations to try whose values are most likely to achieve the target performance on the new deployment in a descending order. They also estimate the number of experiments for a target performance to enable explicit tradeoff between performance target and configuration tuning. Open questions include how to obtain existing configurations, how varied configurations are, how to combine existing configurations with expert data, and how to deal with software evolution. For more information see http://vivo.cs.rutgers.edu/.

- ***FlexFS: A Flexible Flash File System for MLC NAND Flash Memory***
  Sungjin Lee, Keonsoo Ha, Kangwon Zhang, and Jihong Kim, Seoul National University; Junghwan Kim, Samsung Electronics

There exist two types of NAND flash in today's products: SLC is fast and supports a large number of block erases, and MLC supports large capacities. FlexFS is a file system for embedded mobile systems that can use MLC hardware to provide the benefits of both MLC and SLC flash by treating the hardware as MLC or SLC on a per flash-block basis. FlexFS provides the larger capacity of MLC flash to end users, but strives to write as much data as possible to SLC flash blocks to maximize I/O performance. FlexFS also provides a mechanism that mitigates the poor wear characteristics of MLC flash. Their paper was presented during USENIX Annual Tech '09 (see above).

- ***Zephyr: Efficient Incremental Reprogramming of Sensor Nodes using Function Call Indirections and Difference Computation***
  Rajesh Krishna Panta, Saurabh Bagchi, and Samuel P. Midkiff, Purdue University

Zephyr reduces the size of software updates for sensor nodes with a goal of improving battery life. By updating using a modified rsync to reduce the amount of program data required for patching, and using function call indirection, Zephyr requires much less energy for both network and flash storage. Panta also presented on Zephyr during the conference.

DISTRIBUTED SYSTEMS

Summarized by Kiran-Kumar Muniswamy-Reddy (kiran@eecs.harvard.edu)

- ***Object Storage on CRAQ: High-Throughput Chain Replication for Read-Mostly Workloads***
  Jeff Terrace and Michael J. Freedman, Princeton University

Internet storage providers have started providing object storage with eventual consistency semantics. Eventual consistency, said Jeff Terrace, is difficult to program, as it can return stale data on reads to users. The traditional strong consistency is easy to program but hard to scale. This work introduces CRAQ (chain replication with apportioned queries), a storage system that provides strong consistency while also ensuring high scale and availability. CRAQ is an improvement over the chain replication (CR) method. CR organizes all nodes storing an object in a chain, with the head of the chain processing writes and the tail of the chain processing reads. On a write, the head propagates modifications along the chain and acknowledges the write to users once the write has propagated to the tail. On a read, the tail returns the value it has stored for the object, thus ensuring strong consistency. Since all reads are served by the tail node in CR, the tail can be a potential hotspot. CRAQ improves on CR by taking advantage of the fact that all the nodes on the chain have replicas of the data and can serve read requests. CRAQ uses the following scheme to ensure strong consistency in the event that reads are issued while an update is propagating through the chain. Each node that has dirty data (i.e., data that has not yet been propagated to the tail) queries the tail for the current version number of the data and returns that version of the data to the user. The overhead on the tail in CRAQ is smaller than in CR, due to the fact that the tail has to reply with metadata, as opposed to the whole object in CR.

CRAQ can also provide eventual consistency if it is sufficient for the applications, reducing the number of operations across data centers compared to CR. Since one can look up objects from any node in the chain in CRAQ, one can look up objects from the nodes in the local data center, whereas in CR, one has to send the request to the data center that has the tail node. Users can also configure CRAQ in a variety of ways. For example, to ensure datacenter diversity, users can specify the data centers to be used for a chain and the chain size in each data center. Terrace presented evaluation results comparing CRAQ and CR on Emulab: the results confirm that CRAQ scales better than CR.

One audience member commented that Hadoop could really use this scheme for appends and requested that the authors consider contributing this to the Hadoop project. Sourav Bagchi asked what happens if two separate writes originated for the same object in two data centers. Terrace replied that there is a statically defined master data center that coordinates the writes. How might the system change if the operations were persistent as opposed to in-memory (as it is now)? The system would change but the protocol would still be effective.

- ***Census: Location-Aware Membership Management for Large-Scale Distributed Systems***
  James Cowling, Dan R.K. Ports, Barbara Liskov, and Raluca Ada Popa, MIT CSAIL; Abhijeet Gaikwad, École Centrale Paris

Dan Ports presented Census, a membership service designed to work in wide area locality-aware large-scale systems—hence, a platform for building large-scale distributed systems that have to deal with constant churn. Census divides time into epochs and provides consistent membership views to all nodes in a single epoch. Many of the previous membership services were restrictive in that they provided only partial views of system membership, whereas Census provides stronger consistent semantics, thus making applications easier to develop. The basic approach of Census is to designate one node as a leader. The other nodes then report any membership changes to the leader, and the leader aggregates these changes and multicasts the updated membership to members. In the next epoch, members update their membership views.

In order to reduce load on the leader, Census divides the nodes into a hierarchical structure based on network coordinate locality of the nodes. The hierarchical structure is constructed by exploiting the membership knowledge of the system. Since there is a consistent membership view, nodes can reconstruct the tree on the fly, and there is no protocol overhead even during churn. For very large networks, nodes are grouped into regions according to their network coordinates. For such networks, Census makes an exception, and the membership knowledge of nodes is restricted to the nodes in their region. Each node has a summary of membership in other regions. Census uses standard state replication techniques for fault tolerance and can optionally deal with Byzantine faults. An evaluation of Census shows that it imposes low bandwidth overhead per node, reacts quickly to churn, and scales well.

One audience member questioned whether it is possible for branches to occur due to two nodes having different views of the membership tree. Ports replied that there can be temporal inconsistencies, but the nodes can use the version number in each epoch to resolve inconsistencies. Does Census require nodes to store old versions of the views? It helps to have a few recent membership views. How does Virtual Synchrony compare with Census? Ports replied that Virtual Synchrony was more rigorous, but it is similar to their scheme. Does the duration of the epoch affect the scalability? If the epoch is small, their scheme can have slightly higher overhead, but it does not affect scale.

- ***Veracity: Practical Secure Network Coordinates via Vote-based Agreements***
  Micah Sherr, Matt Blaze, and Boon Thau Loo, University of Pennsylvania

Network coordinates (NC) are a decentralized mechanism to estimate approximate network distances between hosts without performing a pairwise measurement. However, NC systems are easy to manipulate. If 10% of the nodes are malicious, there is a 4.9x decrease in accuracy, and if 30% of the nodes are malicious, there is an 11x decrease. Micah Sherr presented Veracity, a security protection layer for NC systems. It differs from existing solutions in that it assumes no triangular invariants, is fully distributed (other schemes assume *a priori* trusted nodes), supports dynamic neighborsets, and does not assume temporal locality.

Participants in Veracity are either publishers or investigators. An investigator is a node that wants to use the publisher's coordinate to update its own. When a publisher returns its coordinate to the investigator, the coordinate is verified by a set of verification nodes (a deterministic set of peers of the node) before the investigator uses it. A malicious node that tries to publish incorrect coordinates will fail this step. After verification, the investigator updates its own coordinate based on the publisher coordinate and the RTT between it and the publisher. The publisher can delay its response to the investigator's probe and induce an error in the investigator's coordinate computation. In the second step, the investigator updates its coordinate only if the new coordinate results in an error below a threshold when computed against a random set of peers. Veracity is implemented by modifying the Vivaldi implementation that is packaged with Bamboo. The authors demonstrated the effectiveness of Veracity under a variety of attacks.

John Dunagan commended Sherr for the thoroughness of their evaluation, but wondered if random delay is the worst an attacker can do. Sherr replied that they looked up all attacks in the literature and came up with some of their own attacks. Further, Sherr agreed that it is hard to assert that Veracity is resilient against all attacks, so they are trying to formalize and verify their system. Could jitter on the WAN affect Veracity? NC systems handle many of these issues. Veracity doesn't distinguish between malice and temporary effects. Veracity does allow users to tweak knobs so that they can handle corner cases in network behavior.

KERNEL DEVELOPMENT

Summarized by Ragib Hasan (rhasan@uiuc.edu)

- ***Decaf: Moving Device Drivers to a Modern Language***
  Matthew J. Renzelmann and Michael M. Swift, University of Wisconsin—Madison

Matthew Renzelmann presented Decaf, a tool for moving device drivers from the kernel to user space. Driver programming and debugging are difficult tasks, and complications can lead to driver unreliability. Renzelmann argued that writing drivers in type-safe high-level languages such as Java can alleviate these problems, but may introduce performance degradation. Decaf solves this by moving most of the driver functionality to user-mode code written in Java, with a only small amount of code running inside the kernel. Decaf provides a migration path for porting existing kernel drivers to user mode. This also makes writing patches easy, to evolve drivers over time.

Decaf builds on the authors' previous work on microdrivers, allowing one to write drivers from scratch and migrate existing drivers. The programmer annotates legacy drivers and then uses the tool DriverSlicer to split the driver code into a nucleus (which runs in kernel mode) and a user-mode library. The developer can then migrate code from the driver library to the Java Decaf driver one function at a time. For example, in porting the ENS1371 sound card driver, Decaf uses Jeannie to allow C and Java code to be mixed in the same file. Complex Java/C transfers are handled using XPC. The authors evaluated Decaf by migrating five existing drivers, showing that porting most of the functionality to user-mode Java code can still provide reasonably good performance.

A member of the audience asked if the authors had studied the memory overhead. Renzelmann replied that there is roughly a 3x memory overhead. Since there was no Java code invoked during the benchmarks, how can the authors be sure that Decaf will correctly handle bugs in the driver? The Java code did run during driver initialization, and the only code left in the kernel mode in C is for faster performance. To a question about refactoring, Renzelmann mentioned that the object-oriented features of Java allowed reduction of the code size for the e1000 driver by 6.5 KB. To a question about the observed bug distributions, Renzelmann referred to their earlier microdrivers study, where they found bugs to be uniformly distributed between kernel and user mode. Alan Thai inquired about performance optimization, and Renzelmann said that they did not use any multi-threading or other optimizations. An audience member who recently discovered a bug in the e1000 driver asked if the authors performed any detailed bug analysis. They had not. How much performance degradation occurred by rewriting the C driver code into Java? Performance loss was not substantial, largely because most of the overhead is in control and in data transfer between user and kernel modes.

- ***Rump File Systems: Kernel Code Reborn***
  Antti Kantee, Helsinki University of Technology

Antti Kantee presented his work on reusing kernel code in user space. A large portion of the kernel code can be run in the user mode with no modifications. Reusing kernel code in user-space applications saves reimplementation time. The Rump File system runs on NetBSD, where it allows different file system codes as user-space processes. The author defined a Rump as a "runnable userspace meta program," i.e., a user-space program which runs kernel code, and a framework that allows this. The Rump kernel runs inside the host OS kernel. The author's goal was to make kernel development simpler. Currently, kernel developers need to use user-mode OS, virtual machines, or emulators to develop and debug kernel code. By allowing unmodified kernel code to be run from user space, debugging and development become easier.

Rump works by using as much kernel code directly as possible. Rump has two modes: a mounted server mode, which is transparent to the applications but where mount privileges are required, and an application library mode, where the application needs explicit modifications but can run with no special privileges. Kantee gave an example scenario in which a corrupt file system on a USB stick can cause a system crash or have exploits. This can be avoided by mounting the device as a Rump file system in user space, thereby isolating the damage to a user-mode process. Rump also makes kernel debugging easier, as various debuggers can be used to give even non-experts control over the debugging process. Kantee talked about a Google Summer of Code project that implemented an application suite providing mtools-like functionality for all file systems supported by Rump. He also showed that Rump is maintainable, with only a small number of Rump breakage commits in the NetBSD repository.

Is the buffer management layer still kept inside the kernel? Kantee answered that Rump uses double buffering, with both the kernel and the application maintaining its own buffer. However, the double buffering is a temporary workaround, which Kantee plans to fix. Is it obvious which part of the interface to port and which one to rewrite? It is not obvious—it's mostly gut feeling.

- ***CiAO: An Aspect-Oriented Operating-System Family for Resource-Constrained Embedded Systems***
  Daniel Lohmann, Wanja Hofer, and Wolfgang Schröder-Preikschat, FAU Erlangen—Nuremberg; Jochen Streicher and Olaf Spinczyk, TU Dortmund

Wolfgang Schröder-Preikschat presented their paper on using aspect-oriented programming in embedded operating systems. Embedded operating systems are widely used in different devices, but to handle different architectures, the code becomes very complex with the use of #ifdef. In the eCos operating system, for example, only two lines of fundamental code requires 34 lines of ifdef blocks, spread over seventeen functions and data structures in four implementation units. This problem, also known as "ifdef hell," makes software complex and difficult to maintain. Schröder-Preikschat argued that aspect-oriented programming (AOP) can solve this by modularizing cross-cutting concerns. After a brief introduction to AOP, the presenter showed how they used AspectC++ (an extension to C++) along with the source-to-source weaver tools to generate normal C++ code from a given aspect specification.

AOP can help here by eliminating the ifdefs. They used this approach in the CiAO system to design configurable embedded-system software. CiAO has loose coupling, visible transitions, and minimal extensions. Important state transitions are captured by a point-cut expression in the aspect. The resulting base system is designed with classes, and most functionality is provided by optional aspects—extension aspects, policy aspects, and upcall aspects. Schröder-Preikschat gave an example of an extension aspect that uses task scheduling. For evaluation, he mentioned their collaboration with Audi and Elektrobit, where they showed that CiAO runs on a number of microcontroller-based systems. He also compared CiAO with OSEK. Finally, Schröder-Preikschat talked about how using aspects for low-level code can break fragile join points and about other issues related to aspect-aspect interdependencies, such as join point traceability and granularity.

Remzi Arpaci-Dusseau from the University of Wisconsin asked whether the ordering of advice procedures has to be considered in the structure definitions. Schröder-Preikschat replied that advice ordering tells in what order the aspects are going to be intermixed in the source code, and so the use of a C pointer may cause some problems. Can aspects be applied at the level of statements? Fine-grained aspect-oriented programming would require using empty functions around statements, but there are not very many cases like that. Sourabh Bagchi from Purdue asked whether using this will be cost-effective, in terms of the effort spent in defining aspects, in real-life scenarios. Configurability of a system will forever be a problem, for example, in the auto industry, which needs very deterministic behavior from their embedded systems.

INVITED TALK

- ***Towards Designing Usable Languages***
  Matthew Jadud, Allegheny College in Meadville, PA, and Christian L. Jacobsen, Untyped Ltd.

  Summarized by Michael von Tessin (mtvt@cse.unsw.edu.au)

Edit, compile . . . edit, compile. This tireless cycle dates back to the 1960s, when the cost of editing and compiling was substantial.

Despite this by now long-standing interaction between human and computer, the observable behavior of novice programmers has only recently been linked with negative affective states. That's a fancy way of saying, "We can detect when students are frustrated." The short-term goal in this study is to support the learner with sensible interventions based on automated observation of their interactions with the compiler and their environment. The longer-term goal is to help provide a human-centered foundation for the design of languages and their environments.

To this end, the language design target is ambitious: parallel languages for robotic control. The authors have built a small virtual machine to support message-passing parallel languages (the Transterpreter) and have begun exploring its use on small, microcontroller-based mobile robotics platforms. They felt this was good engineering: begin by exploring, understanding, and reusing well-tested and formally verified languages with a rich 20-year history. They also thought more people might use the tools if they could play with them on robots made out of shiny yellow plastic. In short, this is a story about people trying to do some cool stuff at the intersection of usability research and the design and implementation of parallel-programming languages.

BlueJ is a development environment used at the University of Kent to teach novice students OO programming in Java. It allows classes, object instances, and invocations to be created/executed via a graphical interface. Skeleton code is automatically created and helps students to start writing code instead of just presenting them with a blank page. BlueJ can be used to trace all compiler invocations, i.e., record the time of invocations and their results (warnings, errors). Their study covers about 42,000 programs in 2000 sessions of 120 students over two years. Overall, 56% of all compiler invocations resulted in a syntax error. The top errors were: unknown variable, missing semicolon, missing bracket, unknown method, app. error, illegal start of expression. They were able to spot compiler error messages that can completely confuse students and frustrate them.

Midway through the talk, the speakers opened the floor to questions. If students know their activities are monitored, does it affect their behavior? They haven't explicitly looked at that, but if monitoring makes them think harder before they hit "compile," that would be an interesting outcome. Someone else asked about variation in traces over the course of a semester. They did see error rates vary and in the types of errors change as students' skills evolved.

When teaching parallelism to students, the speakers continued, they don't want to use an existing sequential language, because the compiler doesn't know how to help students (or even confuses them). Thus, they have chosen occam, which is used a lot at the University of Kent. They want learning parallelism to be fun, but also authentic. "Authentic" in robotics means that although there is a basic sequence—sense → think → act—this is never a strict sequence; you can have multiple inputs (sense), multiple outputs (act), and multiple tasks/calculations (think phases) running in parallel and interconnected.

The authors want the ability to reach out to a community of tinkerers and explorers and the hardware they use in the classroom to be affordable. A good choice is Arduino (http://arduino.cc), which only costs $20 and allows you to buy, download, or build your own software and has a large community behind it. The authors know that designing a new language is hard to do well and to get right. On the other hand, Java was not designed for novice programmers. There were some good examples (Logo and Lego Mindstorms), but they eventually want to get rid of the standard edit/compile cycle.

One speaker said, "In 10 years from now, I don't want my son (now 13 weeks) to learn how to program embedded systems in C. I want tools to be designed for him. So please engage and have a look at baseplate.org and transterpreter.org."

There were some concluding questions. Have you contacted psychologists or other experts in child development? Not yet. That would be future work and very interesting. What motivated you to choose occam? Occam has a long British tradition (Tony Hoare, Bristol) and Kent has a long tradition in that space. Erlang (which is heavily used in telco) has a huge runtime, so we moved away from Lego Mindstorms and started to use occam, which has a very small runtime and memory footprint.

AUTOMATED MANAGEMENT

Summarized by Xu Chen (chenxu@umich.edu)

■ ***Automatically Generating Predicates and Solutions for Configuration Troubleshooting***

Ya-Yunn Su, NEC Laboratories America; Jason Flinn, University of Michigan

Su observed that troubleshooting computing systems is hard. There have been automated troubleshooting tools proposed, but they rely on a given set of predicates that can be used to determine good or bad system states. In this talk, Su proposed methodologies for automatically generating predicates. Existing approaches analyze source code or configuration files, but Su showed that predicates can be extracted from previous user or expert troubleshooting behaviors.

A modified shell is used to record human troubleshooting behavior, including commands executed and resulting kernel-object modifications, while being unobtrusive to the users. The basic assumption is that users usually use repeated commands as predicates, one for recreating the failure and one for evaluating the troubleshooting outcome. The results of these repeated commands should be different, in terms of exit code, screen output, or kernel objects modified. Causal dependencies of different commands are tracked by the modified shell such that the command that solved the problem can be identified, while pruning unrelated commands. To sanitize the user-submitted predicates, they are ranked according to popularity and merged based on the associated state delta.

Su demonstrated through a user study with 12 participants solving four configuration problems that the proposed method can extract correct predicates, with very few false positives (wrong predicates). The false positives are introduced because the users did not perform repeated predicates or did not solve the problem.

The audience raised questions regarding how to pinpoint the exact solution when the user changes a lot of different things. Su emphasized that their methodology currently works at kernel object level (e.g., a file). So the exact change made, such as which line in the file, cannot be determined. How are generated predicates applied to other environments? The generated predicates and solutions are canonicalized so that they can be applied to different users, as shown in their prior work, AutoBash.

■ ***JustRunIt: Experiment-Based Management of Virtualized Data Centers***

Wei Zheng and Ricardo Bianchini, Rutgers University; G. John Janakiraman, Jose Renato Santos, and Yoshio Turner, HP Labs

Managing a data center is hard. To predict system behavior resulting from configuration changes, Zheng proposed an experiment-based approach called JustRunIt. The assumption is that data centers usually run services in tiers, with many instances of VMs for each tier. The basic idea is to create sandboxed VMs to run alongside production VMs so that different parameters can be explored in the sandbox without affecting production services. The sandboxed VMs will be running clones of production VMs. For each tier, an in-proxy and an out-proxy are used to hand over to sandboxed VMs the traffic from production VMs. The management entities usually specify the ranges for different parameters and some time limit for experiments. JustRunIt will try to search the parameter space as much as possible within the time limit. Since the search space is large, interpolation is used if not all combinations are tested. As a simple heuristic, JustRunIt prioritizes the combination of the upper and lower bounds of different parameters.

JustRunIt is implemented in Xen and has been tested on a cluster of machines running multi-tiered Internet services. Evaluation results demonstrate that the overhead of JustRunIt is small. The accuracy of JustRunIt is very good: it can accurately predicate production service behavior in terms of mean response time and throughput.

Zheng presented two usage scenarios in which JustRunIt was used to estimate the impact of hardware upgrades and to derive a better resource allocation strategy in the context of an SLA violation.

Audience members raised several questions. How practical is JustRunIt when the parameter search space is very large? Even though JustRunIt can give an accurate estimate on mean response time, would the variance or exhibited distribution of response time be accurate as well? How does

JustRunIt compare to some greedy approaches in resource management, which, for example, just allocate more VMs if SLA is violated?

- ***vPath: Precise Discovery of Request Processing Paths from Black-Box Observations of Thread and Network Activities***
  Byung Chul Tak, Pennsylvania State University; Chunqiang Tang and Chun Zhang, IBM T.J. Watson Research Center; Sriram Govindan and Bhuvan Urgaonkar, Pennsylvania State University; Rong N. Chang, IBM T.J. Watson Research Center

Tak observed that enterprise services are usually multi-tiered, and a user request usually traverses the system after being processed by a variety of threads that communicate with each other. How each request moves through the system can be characterized by request-processing paths, which Tak tried to discover in his work. There are two types of existing approaches: statistical inference, which is flexible but may not be as accurate; and instrumentation-based, which is accurate but requires source code.

Tak proposed vPath, which resides in a virtual machine monitor and thus is transparent to the application running, yet does not impose too much overhead. vPath identifies the request-processing path by tracking internal and external causality between thread activities within and across machines. The VMM identifies threads and tracks their TCP behavior to identify causalities. This is possible because most commercial applications follow a multi-threaded structure and synchronous communication patterns. vPath seeks to demonstrate that the application model can be exploited to solve the problem of path discovery, presenting a new direction and paradigm.

Implementation-wise, vPath modifies the Xen VMM to intercept some system calls made within each VM. For the related system calls, threads are identified by inspecting the EBP register, while network socket information is delivered by hypercalls. For current support, a guest VM needs to be modified to invoke hypercalls, but in the future such functionality will be merged into the VMM, as Tak pointed out. While delivering accurate processing-path results, vPath exhibited about 6% overhead in increased response time and throughput reduction in a TPC-W test.

There were concerns from the audience about how applicable vPath is to complicated applications whose workload model deviates from vPath's assumptions, and about the benefits vPath could deliver compared to existing inference-based approaches.

SHORT PAPERS

Summarized by Stephen P. Tarzia (starzia@northwestern.edu)

- ***The Restoration of Early UNIX Artifacts***
  Warren Toomey, Bond University

Warren Toomey described efforts by himself and others at the UNIX Heritage Society to restore the first edition of UNIX from 1971. In addition to its historical interest, this case study serves as a lesson in preserving code for use in the distant future. Toomey's story began with the discovery of a printed assembly code listing for the first edition of UNIX. However, much more than the source code was needed. Running the code required several reconstructions, including correcting lines of code, adding peripheral hardware devices to the PDP-11 emulator, and rewriting system utility binaries.

Toomey observed that although software does not physically decay like hardware, it cannot run in a vacuum. The software and hardware environment that it requires is, of course, constantly evolving. This phenomenon is often called "bit rot." The key to preserving code is to preserve the environment as well. This includes keeping contemporary libraries, compilers, configuration files, hardware (or, preferably, a hardware emulator), documentation, anecdotes, and publications.

Toomey reported that a few first-edition UNIX bugs had been discovered during the restoration and had been jokingly posted as security advisories. An attendee asked how to best preserve this kind of restoration work. Toomey encouraged thorough documentation and replication.

- ***Block Management in Solid-State Devices***
  Abhishek Rajimwale, University of Wisconsin, Madison; Vijayan Prabhakaran and John D. Davis, Microsoft Research, Silicon Valley

Abhishek Rajimwale presented an analysis of how Solid State Drive (SSD) characteristics break file systems' longstanding assumptions, and he prescribed appropriate storage-stack changes aimed at improving SSD performance and longevity. In contrast with conventional, spinning magnetic disks, SSDs have low random-access latency, significant background activity (for cleaning and wear-leveling), significant media wear-down, and no seek delay. Additionally, they exhibit write amplification, meaning a small write results in the entire, much larger, block being rewritten.

Rajimwale and colleagues measured characteristics of several different SSD samples, gathered real file-system traces, and modified the SSD module extension for the DiskSim (PDL) simulator. He showed results quantifying the above characteristics and presented three optimizations based on these results. First, the device should merge write requests when possible; second, background maintenance operations should be stalled during high-priority I/O; finally, the SSD should be prevented from cleaning free disk blocks. To implement the above performance and longevity optimizations, Rajimwale called for a richer storage device interface. In particular, he proposed a higher-level interface such as object-based storage, to allow the SSD to manage block-level details itself.

Were there cases when the SSD should provide dynamic status information to the OS? If cells are being switched between multi-level (MLC) and single-level (SLC) modes, that information could be shared with the OS. Do conventional

RAID disk arrays share the same issues? Rajimwale agreed that write amplification has also existed in RAID, but said that the background activity found in RAID (namely, rebuilding) is relatively infrequent; SSD background activity may be continuous. Another attendee suggested letting the OS control low-level block management on the SSD. Rajimwale agreed that OS control is a viable alternative, but suggested that, as SSD devices get more complex, internal control is desired.

- ***Linux Kernel Developer Responses to Static Analysis Bug Reports***
  Philip J. Guo and Dawson Engler, Stanford University

Philip J. Guo presented an analysis of Linux kernel developer responses to bug reports generated by the Coverity static code analysis tool. Their basic goal is to evaluate such tools and to make them more useful by automatically prioritizing the thousands of generated bug reports by correlating bug reports from a single source snapshot to subsequent developer actions in the bug tracker and repository. Such bugs are either ignored or triaged, and the assumption was that developers ignored bugs because they were identified as less important or meaningful. They found correlations between triage rate and several factors such as error type, other static bugs, user-reported bugs, and file age, size, and location.

Guo argued that static analysis is indeed useful. Although static bugs are shallow in nature, he believes that their correlation results show that developers can be led by static bugs to deeper bugs. Quoted reactions from kernel developers support this hypothesis.

The first audience question was whether static-code-analysis tool use results in fewer bugs over time. Guo responded that, since the kernel source is growing, it is difficult to determine such trends. Can a code-complexity metric be used to find deep bugs? Guo supported this idea and pointed out that static analysis bugs often result from code complexity, so these ideas are actually complementary. Had the authors contacted lead kernel developers? Although their developer quotes were anonymized, some were from veterans.

- ***Hardware Execution Throttling for Multi-core Resource Management***
  Xiao Zhang, Sandhya Dwarkadas, and Kai Shen, University of Rochester

Xiao Zhang presented a new software-based multicore resource management mechanism called Hardware Execution Throttling. The general problem is core performance isolation. Adjacent cores typically share a last-level cache, so one core can slow down other cores by overusing the cache. Hardware Execution Throttling cleverly uses two features of Intel Core-series processors: duty cycle modulation (DCM) and cache prefetcher disabling. These settings can be quickly adjusted (within a few hundred CPU cycles) to reduce a core's shared-cache access rate. This fine-grained control is their primary contribution relative to previous mechanisms. Of course, core performance control policy is a separate problem which this work does not address.

Zhang described a fairness metric for concurrent applications. This measures the extent to which all applications are running at the same level of performance. They used a set of standard benchmark applications to favorably compare Hardware Execution Throttling's fairness to previous mechanisms.

An attendee noted that hyper-threading would cause pairs of threads to be throttled together. Zhang agreed and suggested pressuring CPU vendors for more flexible control. Would kernel activity on a throttled core be affected? The kernel could quickly de-throttle that core. Were there cases in which one of the two CPU features worked better for throttling? Zhang didn't have a concrete example, but reported that DCM has a more predictable effect than disabling prefetching. Responding to another question, Zhang said performance was not very sensitive to the particular choice of DCM setting used. Finally, an attendee pointed out that the throttling policy would have to follow a process as it migrated to different cores.

INVITED TALK

- ***The Antikythera Mechanism: Hacking with Gears***
  Diomidis Spinellis, Athens University of Economics and Business

Summarized by Ragib Hasan (rhasan@uiuc.edu)

Diomidis Spinellis presented the history and the functionality of the Antikythera Mechanism, an ancient mechanical computer used for astronomy. Spinellis started with the history of the discovery of the mechanism. In 1900, Greek sponge divers on a fishing trip took shelter from a storm on the island of Antikythera, between the Peloponnese and Crete. One of the divers found some relics when he dived near the island. Subsequent archaeological expeditions found a large number of ancient artifacts, the result of a shipwreck almost 2000 years ago.

Among these was a heavily corroded bronze object consisting of multiple gears. Initially it was thought to be an astronomical toy. Later, in the 1950s, Derek J. de Solla Price, a Yale professor, presented a theory that this mechanism was used to compute the motions of celestial objects. By studying an X-ray of the object, he created a model of how it worked using a combination of many intertwined gears. Price's model was later found to be incorrect, but it formed the basis of more thorough studies published in the journal *Nature* in 2006 and 2008 (see http://www.antikythera-mechanism.gr).

A total of 82 fragments of the mechanism are available. Recently, researchers studied the mechanism using digital radiographs, X-ray tomography, and a 3D lighting model. Inscriptions on the mechanism serve as a user manual. The researchers found that it could calculate days in the Egyptian calendar. The device could also compute the motion and phases of the moon and predict solar and lunar eclipses. As a comparison, Spinellis showed that the equivalent code in BSD's moon phase program is very complex. The Antikythera Mechanism used the Metonic calendar and computed complex astronomical predictions, as well as the year of the Olympic games.

Spinellis created a complete Squeak EToys emulator for showing how the mechanism worked. The emulator is available at http://spinellis.gr/sw/ameso. He demonstrated the working of the mechanism with animations from his emulator. Finally, Spinellis pondered the purpose of the device. He said that we may never know the real purpose of the tool, but it might have had some political and strategic value, since prediction of eclipses was important at that time. This mechanism was too delicate to carry on a ship to aid in navigation. This could also have been an educational tool, or simply something someone (possibly an ancient hacker) built for fun.

A member of the audience asked what was the complexity of the mechanism compared to other machines. Spinellis said that no other such mechanism from that time has been found. Another person remarked that there was a clock built during the Renaissance to simulate the motion of the planets, and asked if this could have done the same thing. Spinellis said experts have posited that, with some gears, it could have calculated planetary positions too. The mechanism was expensive to build just for moon positions, so it probably also computed the paths of others celestial bodies. Did the device have any bugs? The device was designed in a very clever manner, and there are no bugs in the mechanism.

SYSTEM OPTIMIZATION

Summarized by Abhishek Rajimwale (abhi@cs.wisc.edu)

- ***Reducing Seek Overhead with Application-Directed Prefetching***
  Steve VanDeBogart, Christopher Frost, and Eddie Kohler, UCLA

Steve VanDeBogart addressed the problem in prefetching arising from non-sequential accesses by applications in systems using disks. He introduced a new prefetching algorithm which uses applications' knowledge of future accesses and is implemented in the form of a user-space library called "libprefetch." libprefetch provides a convenient application interface, needs little modification to the kernel, and handles resource sharing.

Steve then presented the details of the intuition behind libprefetch. He showed that for seeks above 1 MB there is a very gradual increase in cost; however, reducing seeks larger than 1 MB to less than 32 KB (on average) can result in significant performance gain. He also showed that using larger reorder buffers greatly helps to reduce average seek cost as well as number of disk passes. Next, he explained the libprefetch's interface, which uses a callback mechanism, and also talked about how libprefetch solves the problem of contention in memory by using a TCP-like mechanism (i.e., additive increase and multiplicative decrease). Finally, libprefetch showed an up to 20x improvement in some benchmarks using real applications.

Someone in the audience asked about whether applications need to use "pread" only to be able to use libprefetch. Steve clarified that they intercept read, pread, and other variants. How much gain do the authors expect if they don't use spinning disk and use SSDs (or RAID arrays) on these benchmarks with libprefetch? Disks will still exist as long as SSDs are expensive; for RAID arrays, it's possible to extend this work by pushing information up from RAID arrays about the layout; as far as direct performance gains on SSDs are concerned, there may be some gains. What is the generality of the non-linear relation between seek time and seek distance, and what are the reasons behind it? Although they had limited samples of disks in their results, similar results have been shown in previous works. The reason is due to rotational latency and seeks. Had they tried to use selective joins (queries) to see gains? As long as access patterns are known, performance will improve with libprefetch. For more information see http://libprefetch.cs.ucla.edu.

- ***Fido: Fast Inter-Virtual-Machine Communication for Enterprise Appliances***
  Anton Burtsev, University of Utah; Kiran Srinivasan, Prashanth Radhakrishnan, Lakshmi N. Bairavasundaram, Kaladhar Voruganti, and Garth R. Goodson, NetApp, Inc.

Fido is targeted to enterprise-class server appliances such as NAS, with the aim of addressing the main problem of performance in virtualizing NAS in order to exploit the natural benefits of virtualization. The main insight, Anton Burtsev said, was to use the relaxed trust model in these appliances to design fast inter-VM communication by sharing memory read-only across VMs in the same appliance.

He then detailed how their fast inter-VM communication works using a large pseudo-global virtual address space and mapping the address of all VMs in one single large address space. Transitive zero copy is achieved by mapping the communicated read-only data into this global virtual address space; a shared memory ring is used to pass pointers. Two interfaces are used to access Fido: MMNet (network device interface) and MMBlk (block device interface). Anton then presented some evaluation figures demonstrating that MMNet outperforms XenLoop and Netfront, sometimes also outperforming the monolithic kernel due to inefficiencies in the TCP stack. Further, MMBlk outperforms XenBlk and also the monolithic kernel due to contention in tmpfs and ext3. Finally, Anton presented the case study with NAS in order to give a more realistic performance evaluation. He showed that with Fido, NAS can be virtualized with little performance overhead on micro-benchmarks and TPC-C macro-benchmarks by exploiting pipelined parallelism between VMs and by eliminating copy and page-mapping overheads in the critical path.

An audience member asked what mechanism was used to reclaim memory that is shared read-only with other VMs. Anton replied that they didn't have any special mechanism to reclaim memory; for TCP they reclaim memory voluntarily when the other VM frees it, but for file systems they have to copy out memory in the other VM. He had already acknowledged this limitation in his conclusions. Someone from VMware also expressed concern about using this relaxed trust model in the face of users pushing malicious content into servers. Anton suggested that this model is a required assumption for this work and, at least for the networking stack, is a reasonable assumption. What are the benefits of virtualizing NAS if there is no fault isolation because of the relaxed trust model? VMs are a pragmatic approach with benefits of migration, cleaner hardware support, and better isolation with very little performance overhead. Future work might include implementing some micro-rebooting technique to provide fault isolation.

- ***STOW: A Spatially and Temporally Optimized Write Caching Algorithm***
  Binny S. Gill and Michael Ko, IBM Almaden Research Center; Biplob Debnath, University of Minnesota; Wendy Belluomini, IBM Almaden Research Center

Binny Gill presented a new writ-caching algorithm called STOW. He explained the need for the algorithm by showing that the destaging rate from write caches is important apart from just destaging order. He pointed out that earlier work, including his own WOW algorithm, had only focused on destaging order.

Binny presented the intuition behind the new algorithm in steps. He first pointed out problems with simplistic techniques for destaging, such as destaging as quickly as possible or having a fixed destage threshold. He suggested that destaging with linear thresholding (high and low) is required to control the destage rate, but even with linear thresholding, the destaging occurs in spikes due to the long time spent in sequential and random regions. With this, he introduced the notion of separating random and sequential data streams. However, this leads to low throughputs because mixing sequential and random streams hurts disk throughput by forcing the disk head to service two separate regions instead of one. He further explained that this can be controlled by adding hysteresis to the destages. The last important thing in the algorithm is to adapt the sizes of the sequential and random queues to be responsive to the workload. He then presented a thorough evaluation of his algorithm, comparing it with CSCAN, LRW, and WOW. STOW outperforms all the other algorithms in throughput and response time for both full back-end and partial back-end experiments. STOW gives an average of 18% improvement over the previous best algorithm (WOW), which is substantial because of the slow nature of disk I/O improvements in hardware.

Someone asked why adaptation was required with sequential queues, particularly because there is little temporal locality with sequential writes. Binny replied that this adaptation was required for multiple streams. In order to get maximum throughput, each stream must have some amount of data to say it's sequential, for at least a stripe or two stripes worth of data. So we need to adapt the size of the sequential queue according to the number of concurrent sequential streams. Why did he choose a simple hysteresis rather than some more complex mechanism? He likes to keep things simple so that they are actually used. He acknowledged that there could be some further gain in throughput (around 5% more) by using more complicated mechanisms, but he wouldn't worry about that more than the real applications of his work. Could the technique of using separate queues with hysteresis be used to dynamically adjust the sizes of read and write caches? This was an open and complex problem he hadn't dealt with.

WEB, INTERNET, DATA CENTER

Summarized by Wei Zheng (wzheng@cs.rutgers.edu)

- ***Black-Box Performance Control for High-Volume Non-Interactive Systems***
  Chunqiang Tang, IBM T.J. Watson Research Center; Sunjit Tara, IBM Software Group, Tivoli; Rong N. Chang and Chun Zhang, IBM T.J. Watson Research Center

Chunqiang Tang provided examples of systems that process requests generated by automated software tools, in addition to requests generated by interactive users, e.g., Twitter, WebCrawler, and IT monitoring and management systems. Systems that have non-interactive workloads generally benefit more from high throughput than from short response time.

Tang proposed a general black-box performance control named TCC (throughput-guided concurrency control), which varies the number of event-processing threads to maximize throughput. TCC keeps adding more threads and observes whether throughput increases. After finding peak throughput, TCC decreases the number of threads so long as throughput does not decrease significantly. Tang also described how to measure throughput accurately and efficiently through sampling and noise removal.

TCC was demonstrated to maximize throughput and control resource to near-saturate level by analyzing different event-processing queue models. The control is also evaluated in a real implementation to demonstrate the scalability of TCC and its effectiveness under various bottleneck scenarios. Tang said future work might include applying TCP-style flow control to general distributed systems. Emphasizing that performance control for non-interactive systems is an interesting problem.

Someone asked whether the time to measure throughput is deterministic. Tang said it is dynamically adjusted. Is the increase and decrease of thread number by percentage? Tang said yes. Can TCC deal with multiple pools of threads?

That is a limitation of this approach and should be studied for future work.

- ***Server Workload Analysis for Power Minimization using Consolidation***
  Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari, IBM India Research Lab

Akshat Verma described the characteristics of the workloads collected from the *Fortune* Global 500 over a period of 90 days in 2007. Based on the observation, Verma proposed two new consolidation methodologies, Correlation-Based Placement (CBP) and Peak Clustering-Based Placement (PCP).

The idea behind CBP is to separate positively correlated applications across servers. However, it cannot capture both the body and the tail of the workload distribution. PCP addresses this problem by using two parameters to decide collocation. The first can be mean or percentile for body and the second can be a tail-based metric. All correlated peaks will be separated across active servers, and the off-peak reservation can be equal to the body value. CBP and PCP are compared against peak-based and mode-based sizing approaches. The results show PCP consuming as little power as the mode-based approach while having very few capacity violations across different application suites.

Someone asked how the power is measured in this work. Verma responded that the power was calculated from a model instead of by measurement. Given that dynamic load balancing/consolidation is popular in industry, how much room is left for static consolidation? Even in that situation, PCP will help to decrease the number of migrations. Chunqiang Tang from IBM Research asked if other resources are considered. Verma said no, but believed that PCP can be extended to deal with other resources.

- ***RCB: A Simple and Practical Framework for Real-time Collaborative Browsing***
  Chuan Yue, Zi Chu, and Haining Wang, The College of William and Mary

Chuan Yue presented RCB, a pure browser-based collaborative browsing framework. A co-browsing host starts RCB-Agent, a Web browser extension. Later, co-browsing participants connect to the host using regular Web browsers. Any of them can visit a Web page and interact there, with all Web page contents automatically synchronized between the host and participants.

The authors implemented RCB as a Firefox extension and evaluated it with real-time performance in LAN and WAN settings. The results showed that a Web page can be synchronized between a host and a participant in a reasonable amount of time. RCB worked with Google Maps and Amazon Checkout. User studies indicated that most people like it and tend to continue using it.

The audience responded actively. One person wondered why RCB is better than screen sharing. Yue pointed out that RCB puts less stress on bandwidth and secure assurance. Would a password be transmitted to others? RCB can enforce a policy on the host side. If multiple participants submit changes simultaneously, what will happen to a Web page? The host can write a policy to control what action to perform if multiple changes are received. Will RCB work if one has limited access to the Internet? As long as a participant can connect to the host, RCB should work.

INVITED TALK

- ***A Computer Scientist Looks at the Energy Problem***
  Randy H. Katz, University of California, Berkeley

Summarized by Stephen P. Tarzia (starzia@northwestern.edu)

Randy Katz described how his group and others are applying computing-inspired solutions to the electrical power piece of the global energy problem. First, he described computing's role as a major energy consumer and proposed strategies increasing efficiency. Second, he addressed electrical generation and distribution. He used the Internet as a model for how the power grid might be re-engineered as a more efficient and robust distributed system.

Katz presented an overview of energy sources and sinks in the US. Currently, most electricity is generated from coal. Renewable energy sources have severe limitations, so limiting demand is important. For example, the country with the highest percentage of electricity from renewables is Denmark, with only 28%. Looking at IT energy consumption, Katz referred to the Smart 2020 report, available at http://smart2020.org. Despite the anticipated invention of new efficiency technologies, IT power drain is projected to reach 4% of total energy consumption by the year 2020.

Katz also touched on some of the same themes as the keynote, showing the energy demand breakdown in Internet data centers and describing some possible optimizations. He suggested that containerized data centers (racks of servers built into shipping containers) may be more efficient, since their internal layout and airflow can be highly optimized.

Power Usage Efficiency (PUE) optimization at the data center is already approaching optimality, so future optimizations will be within IT equipment. One of Katz's slogans is that computers must be made to "do nothing well." This goal, also called energy proportionality, means that idle computers should drain nearly zero current; this is currently not common. However, Katz showed that low-power CPUs such as Intel's Atom are more energy proportional. Since average server utilization tends to be low, one might replace each high-power CPU with several slower Atom CPUs. However, other system components such as RAM, disk, etc., currently have high idle power consumption, so optimizing the CPU is not enough.

In the second part of the talk, Katz described his plan for upgrading the power grid. He drew an extended analogy with the telephone network's evolution in the Internet era.

Because weather variations make renewable energy sources such as wind and solar both unreliable and distributed in nature, the need for a power grid upgrade is essential. Like the Internet, an effective renewable-source grid must have local buffers (energy stores) and adaptive routing. Energy storage is tricky. On large scales, one can pump water uphill or compress gasses. On the small scale, for example in homes, batteries would work but are expensive. Anticipatory work such as cooling a building earlier in the day can actually be thought of as a type of energy storage.

Katz proposed building a smart power grid by augmenting the existing grid with Internet-connected Intelligent Power Switches (IPSes). He also emphasized that an intelligent power grid with open access would bring "power to the people" in both senses of the word; it would allow enterprising individuals to contribute excess generated electricity to the grid.

Rik Farrow asked about energy storage, and Katz noted that there are many innovative options for energy storage, including using water temperature differentials. He described storage as an information management problem. Responding to a question about laptop versus desktop computer power usage, Katz noted that power management policies are currently lacking. He would like to see faster transitions from low-power to operating states and back again. Alan Thal suggested cooling data centers by building them underground. Katz responded that people are looking at innovative data center locations and that architects do have a role to play in the optimization process. Another attendee noted that room cooling is often overlooked when calculating the power drain of computers. Katz replied that PUE measurements in data centers include this and that significant energy savings can be had by allowing server rooms to reach higher temperatures (but monitoring them more carefully). As a follow-up, another attendee noted that much hardware lacks good air flow, so we may have to lean on vendors to improve this aspect of their products. Katz agreed that systems packaging is an issue and mentioned that the containerized design brings airflow to the forefront of design.

An attendee asked whether the next optimization step is choosing what we are willing to compute at all. Katz acknowledged this as the ultimate goal and a broader challenge, although it is rarely mentioned. He suggested modeling user desires and behavior and then structuring the entire system to meet those demands.

BUGS AND SOFTWARE UPDATES

Summarized by Michael von Tessin (mtvt@cse.unsw.edu.au)

- ***The Beauty and the Beast: Vulnerabilities in Red Hat's Packages***
  Stephan Neuhaus, Universita degli Studi di Trento, and Thomas Zimmermann, Microsoft Research

Stephan Neuhaus explained that they used Red Hat Security Advisories to determine which Red Hat packages had vulnerabilities reported. The distribution shows that two-thirds of all packages didn't have any vulnerabilities, whereas the kernel had the highest number of vulnerabilities. They asked if there are package properties that correlate with vulnerabilities and found that there were.

If A → B (A depends on B), then A is vulnerable if: (1) A is in an "insecure domain" ("domain" characterized by dependencies, e.g., "Internet browsers," "image manipulation programs"); (2) B is difficult to use securely (e.g., SSL); (3) a fix in B spills over to A (e.g., a change in the API).

Next they created a dependency subtree of packages, with each node having an attached risk equal to the number of packages that depend on this package. They could then find the "beauties" and the "beasts" by comparing this risk between a parent and a child in the dependency subtree. If the risk of the child is significantly higher ($p < 0.01$) than the parent's, that means that the child is the "beast," and vice versa. They then used machine learning (Support Vector Machine, SVM) to predict the future vulnerability of a package. Their model correctly predicted 10 of the 25 packages found to have vulnerabilities over the next eight months. Programmers can use this research to help choose less risky packages to depend on.

An audience member observed that they looked at binary packages instead of source packages, which could give them some strange anomalies when multiple binary packages are generated from one source package (e.g., OpenOffice). Stephen admitted that looking at source packages would make sense as well. Someone wondered whether there was a correlation vs. causation problem here; instead of implying vulnerability, might the correlation instead say something about the carefulness of the programmers in choosing which packages to use? Neuhaus responded that this was a good question, but that they tested the model on real data and were able to successfully predict the future. So there has to be some truth in it. But even if it came down to choice of packages, it would still be a useful outcome, because a programmer would now know which packages a clever programmer prefers to use. For more information see http://research.microsoft.com/projects/esm and http://www.artdecode.de.

- ***Immediate Multi-Threaded Dynamic Software Updates Using Stack Reconstruction***
  Kristis Makris and Rida A. Bazzi, Arizona State University

Kristis Makris said that software updates to patch critical security holes are arriving with increasing frequency and need to be applied as soon as possible. In a live system, downtime should be as limited as possible (milliseconds instead of minutes). This means that the system needs to be updated dynamically while it is running. In order to do DSU (Dynamic Software Update), we need to know when a system is in a state that allows updating and the mapping that maps old data structures to new data structures. But this problem is undecidable, i.e., there is no algorithm that can always find a correct solution.

Many DSU systems allow old and new code to be executed at the same time (and to use adaptors for accessing data structures), making it very difficult for users to determine valid update points. The authors' work had three design goals:

1. Atomic update: The entire state of an application has to be transferred (from old to new representation) in a single step, before which only old code was running and after which only new code is running.

2. Transaction safety: In certain cases, you need to execute a transaction (critical section of code) without allowing updates in between. This means that such code is either fully executed as old code or as new code, but not mixed.

3. Thread safety: If there are multiple threads sharing a state (e.g., in a Web server), an "immediate update" is needed. Immediate means atomic and with bounded delay (no blocking).

UpStare consists of a compiler, a patch generator, and a runtime. UpStare saves stack frames, updates global state, then reconstructs stack frames. Update points can be automatically set or set manually by the user. Thread safety is implemented by forcing all threads to block in case of an update request, safely detecting that they are blocked, and only then performing the update. Overheads during evaluation ranged from 38% to 97% with throughput decreases of none to 26%. Future work involves moving cold code to the end of image (improves cache locality), adding runtime safety checking, using semantic analysis, and updating in-transit data.

The session chair, John Dunagan, asked the only question, wondering about future work that aims to reduce the amount of user involvement. Couldn't the opposite be useful, i.e., to force programmers to annotate their programs sufficiently? Kristis said that would be a very good idea. It would help to automatically generate the patches fully, without user involvement (except for the annotations, of course).

- ***Zephyr: Efficient Incremental Reprogramming of Sensor Nodes using Function Call Indirections and Difference Computation***
  Rajesh Krishna Panta, Saurabh Bagchi, and Samuel P. Midkiff, Purdue University

Rajesh Krishna Panta explained that the goal of this work is to enable software updates on wireless sensors on the fly and "in situ." Because these devices are battery-powered, reprogramming must be fast and energy-efficient. Their approach achieves energy efficiency by sending as small a diff (delta script) as possible to the sensor, which then applies it to the current code to get the updated code. The computationally intensive part of this setup is done on the host (finding the delta script), whereas applying it is straightforward. They used rsync to calculate a delta script between the old and the new binary (on byte-level). Because of some shortcomings, they had to improve rsync quite a lot (e.g., merging superblocks).

This approach only works fine if functions in the new binary have the same addresses as in the old binary; otherwise, all jump addresses will change, making it very difficult to find a small binary diff (delta script). Solutions to this problem include: (1) leaving space after each function to avoid having to shift parts of the image if functions grow, or (2) using position-independent code, which is available only on certain architectures. Their solution uses function call indirections. All calls are performed into a fixed indirection table, with each function having its predefined slot which doesn't change on updates. Thus, only the contents of the table change, but all jump addresses in the binary stay the same. In all benchmarks, Zephyr is multiple times smaller/faster/more efficient than Deluge or Stream. Future work is to remove function-call latency (due to indirection table) by having the loader relocate the binary according to the indirection table.

Someone pointed out that this is very similar to what a dynamic linker has to do (e.g., indirection table). John Dunagan asked how often updates are typically needed in a wireless sensor environment. Rajesh didn't have numbers, but his experience told him that updates occur quite frequently. Most updates are quite small, e.g., because the environment changes and sensors might have to be reprogrammed to behave a little bit differently. Very small bug fixes or very large updates are rare.

CLOSING SESSION

- ***Third Millennium Problem-Solving: Can New Visualization and Collaboration Tools Make a Difference?***
  David Brin, Hugo Award-winning author

Summarized by Rik Farrow (rik@usenix.org)

David Brin introduced himself as an astrophysicist by training who is also a book author and futurist. Brin headed off into his title theme, but quickly took off in several intriguing directions.

Brin explained that the horns depicted on Moses' head in Renaissance paintings weren't really horns but "lamps on his brow." These lamps are, in turn, a metaphor for the frontal lobes of the human brain that allow us to plan for the future and "discover the troubles in front of you before you stumble into the pit." As a futurist, I have no doubt that Brin uses his horns a lot.

Brin, like other futurists, is very interested in the singularity, the point when humans have computer-enhanced intelligence, or strong AI exists. Brin believes that the singularity is approaching within the current generation, due to the acceleration in technological and social advances that started in the 15th century with the development of printing presses and glass lenses. Printing presses democratized knowledge, while glass lenses made it possible to study the solar system—incidentally uncovering the fact that Earth is not the center of the universe.

The 18th century brought with it mass literacy, printed illustrations, and science, or Brin's memory, vision, and attention. The 19th-century version of these three themes were mass education and public libraries, photography and cinema, and global communication. In the 20th century, we got computers and databases, television and mass media, and abstraction and immersion. By sometime in the 21st, we will have a knowledge mesh, omniveillance (stick-on cameras with IPv6 and one-year batteries) and super immersion. The acceleration of technology, including Moore's Law, will bring about the merger and/or replacement of humans with post-humans and/or AI.

Brin told us that Internet millionaires, like his distant cousin Sergey Brin (Google), believe in positive sum games. The world of the future should not rely on scarcity for worth but be a world where everyone gains.

Brin spoke on many other topics, one of the strongest being a plea for CERTs: Community Emergency Response Teams. Brin pointed out that the many of the most effective responders during 911 were members of the local community, and that we need to support training for CERT members as well as develop P2P communication that will stand up during emergencies such as Katrina.

Eventually, Brin slowed down and opened the floor to questions. Matt Blaze strode to the mike and picked out just one of the many controversial points Brin had made, that no online argument has ever been settled. Matt said that he can count "zillions of times I've been personally informed by an online discussion that I never participated in that prevented me from spreading wrong information." Hey, me too, Matt. Brin feinted by suggesting that we should turn portions of the Internet into arenas for ideas with rankings by reputation for the posters. Blaze countered by suggesting that the Internet may have evolved a generation with better bullshit detectors. Brin agreed, saying that he still wanted better tools for discourse.

Stephan Neuhaus disagreed with Brin's point that graduate school has forced many people into very narrow and focused interests and that this was actually harmful. Neuhaus contended that poor countries really needed to build a professional class. Brin said that he thinks the Third World will quickly pass through their own over-professionalization curve.

You can learn more about David Brin and his thoughts on his Web site: http://davidbrin.com/.

Workshop on Hot Topics in Cloud Computing (HotCloud '09)

San Diego, CA
June 15, 2009

Summarized by Alva Couch (couch@eecs.tufts.edu) and Kiran-Kumar Muniswamy-Reddy (kiran@eecs.harvard.edu)

Cloud computing remains a "cloudy concept" for many people. The first USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '09) brought together academic and industry researchers to discuss late-breaking results and current trends in cloud computing. As in other "hot topics" conferences, HotCloud papers defined a problem and discussed a possible solution and preliminary results. Results ranged from performance of specific management strategies to designs for new components of cloud infrastructures. Full papers discussed upcoming research plans in detail, while short papers described an interesting idea worthy of further study. HotCloud '09 included 13 full papers and eight short papers, resulting in a day packed with new ideas and future challenges.

The workshop discussed several distinct kinds of clouds that are distinguished by the kinds of services that they provide to clients:

- Software as a Service (SaaS): clients gain access to specific software functions (e.g., gmail, Google Maps).
- Platform as a Service (PaaS): clients gain access to individual virtual machines: (e.g., Amazon Web Services, Eucalyptus).
- Infrastructure as a Service (IaaS): clients gain access to networks of (perhaps physical) machines (e.g., virtual data centers).

The kind of cloud determines the boundaries between a client's responsibility and the cloud provider's responsibility. In SaaS the client uses the application as an exterior entity. In PaaS the client must load an operating system instance into a virtual machine, while in IaaS the client might have to choose, deploy, and manage provisioning software that in PaaS is part of the service.

Clouds and cloud applications can exhibit (or lack) *elasticity,* the ability to dynamically adapt to changing use patterns by provisioning and decommissioning resources and virtual instances of servers. In SaaS elasticity is completely invisible to the client; in PaaS the client must enable elasticity by providing images of virtual instances suitable for replication; in IaaS the client may be responsible for ensuring elasticity by choosing, deploying, and managing an elasticity application.

One motivation for "pushing an application into a cloud" is to reallocate responsibilities and risks from client to provider. Clouds can be characterized by the kinds of risks the provider assumes:

- **Compute clouds** provide computational power on demand. The provider assumes responsibility for availability and reliability of compute servers.
- **Data clouds** provide data persistence and preservation, where data can include file systems or databases. The provider assumes responsibility for data availability, integrity, and persistence.
- **Service clouds** provide and ensure function of a specific service. The provider assumes all responsibility for providing the service.

Of course, many cloud infrastructures provide all of the above.

Ensuring security and privacy for cloud data is more difficult than ensuring security and privacy in non-cloud infrastructure. Several security and privacy threats repeatedly arose at HotCloud, including:

- **Malicious use of privilege:** The maintainers of the cloud have administrative privilege and thus clandestine access to client data that they do not own.
- **Exploitation of co-location:** Malicious client applications can discover confidential information about other clients whose cloud functions happen to be co-located on the same physical devices, by employing back-channels, including shared use of memory, I/O, cache, or even address translation buffer behavior.
- **Limits of legal protection:** The Stored Communications Act (SCA) provides less legal protection against subpoena for cloud data than for data stored on self-owned hardware.

Thus, cloud clients may assume implicitly that providers are mitigating risks that may be beyond the providers' capabilities to mitigate. Many presenters assumed that all data in a cloud is public, sidestepping these difficulties, while others specifically considered the difficulties of keeping data private.

Finally, there was much discussion and controversy over eventual versus strong consistency in data clouds. In distributed database theory, a database exhibits *strong consistency* if changes to the data store are reflected immediately in subsequent queries, and *eventual consistency* if it is possible that changes will not be reflected in queries until a later time. Data clouds can likewise exhibit either strong or eventual consistency. While financial transactions such as purchases usually require strong consistency (so that the customer sees a purchase record immediately after a purchase), eventual consistency is usually acceptable for the results of a crawler or Web search. But this is a controversial issue: eventual consistency is "fun for computer scientists," but difficult to handle in practice, and leads to bugs in applications.

CLOUD PLATFORMS AND ARCHITECTURES

Full Papers

- ***Open Cirrus™ Cloud Computing Testbed: Federated Data Centers for Open Source Systems and Services Research***
  Roy Campbell, Indranil Gupta, Michael Heath, and Steven Y. Ko, University of Illinois at Urbana-Champaign; Michael Kozuch, Intel Research; Marcel Kunze, KIT, Germany; Thomas Kwan, Yahoo!; Kevin Lai, HP Labs; Hing Yan Lee, IDA, Singapore; Martha Lyons and Dejan Milojicic, HP Labs; David O'Hallaron, Intel Research; Yeng Chai Soh, IDA, Singapore

Open Cirrus is a cloud computing testbed with 11,000 cores, global services, and an open source stack, with nine sites and a planned size of 20 sites. Objectives of Open Cirrus include providing a vendor-neutral testbed for cloud technologies, collecting realistic traces of workload, and exposing the research community to realistic enterprise requirements. Infrastructure for Open Cirrus includes Tashi-provisioning software from Intel, as well as Hadoop for programming. Open Cirrus is intended to serve as a testbed for metrics of success for cloud computing and thus to inform the decision of whether to lease or own cloud infrastructure. As Open Cirrus is an international infrastructure, challenges include issues of privacy and legality. Users of Open Cirrus must develop separate service agreements with each of the nine international sites. Data privacy is difficult to guarantee when private data is hosted at foreign sites.

- ***Nebulas: Using Distributed Voluntary Resources to Build Clouds***
  Abhishek Chandra and Jon Weissman, University of Minnesota

Nebulas are a form of cloud computing based on voluntary cooperation and inspired by the success of edge-node computing infrastructures such as SETI@home. Voluntary, loosely coupled clouds based on an edge-node computing model seem to have several advantages, including an estimated two orders of magnitude cost difference between SETI@home and Amazon Elastic Compute Cloud (EC2). Nebulas, unlike clouds, implement elasticity through use of excess resources on volunteered distributed hosts. This leads to low cost, at the price of lower potential performance and higher volatility due to dynamic variation in resource availability. Challenges include coping with heterogeneity during deployment, fragility and churn, and data privacy. Threats to privacy arise both from privileged users on the volunteered hosts and from back-channels through co-location of Nebula services.

- ***Towards Trusted Cloud Computing***
  Nuno Santos, Krishna P. Gummadi, and Rodrigo Rodrigues, MPI-SWS

Trusted cloud computing refers to a situation in which data in the cloud—both computed and stored—remains private and protected from data leaks. One threat to data privacy is that cloud administrators have privileged access to virtual machine instances but do not own data contained in the instances. The cloud provider must be trusted to provide physical security and to limit physical access to cloud infrastructure. Software support for trust—which is effective only in the presence of physical security for cloud hardware—includes secure booting and remote attestation of state (i.e., some proof that privacy is being maintained). Challenges for trusted cloud computing include building trusted virtual machine monitors (VMMs) based on key infrastructure provided by trusted platform modules ("TPM chips"), and providing facilities for secure service migration without potential exposure of private data.

- ***The Case for Enterprise-Ready Virtual Private Clouds***
  Timothy Wood and Prashant Shenoy, University of Massachusetts Amherst; Alexandre Gerber, K.K. Ramakrishnan, and Jacobus Van der Merwe, AT&T Labs—Research

A virtual private cloud is an "Infrastructure as a Service" (IaaS) cloud mechanism whereby enterprises can augment in-house computing resources by renting remote computation and storage infrastructure transparently, securely, and flexibly. For IaaS to be practical, legacy applications have to be able to execute in the cloud without being specifically aware of where they are executing. Current cloud mechanisms for IaaS are difficult to secure if applications are not aware that they are running in a cloud, including firewall configuration. A virtual private cloud (VPC) establishes secure connections between owned and cloud infrastructure using dynamically configured layer-2 or layer-3 multi-protocol label-switching (MPLS) virtual private networks (VPNs). Advantages of VPCs include no requirement for end-node configuration and ability to transparently migrate existing applications to the cloud. Challenges for VPCs include the need for virtualized routing infrastructure and for a mechanism to make traditionally static VPN allocation dynamic, perhaps through Border Gateway Protocol (BGP) signaling. The audience expressed concern that the enterprise is giving the cloud provider's administrators privileged access to their owned infrastructure via VPC connections, thus increasing the risk of data leaks.

Short Papers

- ***Private Virtual Infrastructure for Cloud Computing***
  F. John Krautheim, University of Maryland, Baltimore County

One way to improve data privacy in a cloud is to utilize public-key cryptography to secure private information within virtual machine instances. A locator bot (lobot) is a virtual cloud appliance that stores an instance's private keys and manages the instance that utilizes those keys, thus allowing applications inside the instance to access encrypted resources. Lobots are created by a Private Virtual Infrastructure Factory (PVI factory). Challenges of creating lobots include how to measure and validate the security of the fabric in which the lobots execute, as well as protecting against object reuse during object shutdown. Private data leakage due to co-location of malicious clients might remain a problem due to persistence of in-memory copies of decrypted data.

- ***Refactoring Human Roles Solves Systems Problems***
  Jeremy Elson and Jon Howell, Microsoft Research

The success of cloud computing depends on decomposing the task of cloud deployment into human roles with clearly defined and minimal interfaces. In the same way that the software industry decoupled the user from the software developer, new roles in cloud implementation have the potential to decouple parts of the cloud implementation process with positive results. The "hardware wrangler" builds the hardware infrastructure for a cloud, while the "software integrator" chooses the software and versions to execute on that hardware. Inappropriate (or perhaps a better term is "over-specified") interaction between cloud client and integrator leads to "DLL hell" in which desired configurations are impossible to deploy, while inappropriate interaction between application developer and integrator can lead to vertical "stovepipe" architecture with minimal reuse. Challenges include limiting interactions between roles so that system administration of the result remains practical.

ELASTIC CLOUDS AND RESOURCE MANAGEMENT

Invited Short Presentation

- ***GENI and Cloud Computing***
  Harry Mussman, BBN Technologies

The Global Environment for Network Innovation (GENI) is an NSF project in support of experiments in network design. While GENI is not itself a cloud infrastructure, GENI encourages cloud researchers to build clouds on top of the GENI infrastructure, which is deeply programmable at a network level to support networking protocols other than the Internet. GENI is being developed by 29 teams, both academic and industrial, and an initial version will be available for initial experiments in 2009 and fully operational by 2010. GENI asks the cloud community to become involved by communicating specific needs for cloud research to the GENI developers.

Full Papers

- ***ElasTraS: An Elastic Transactional Data Store in the Cloud***
  Sudipto Das, Divyakant Agrawal, and Amr El Abbadi, University of California, Santa Barbara

The Elastic Transactional Data Store (ElasTraS) is a data storage mechanism that adds distributed transaction processing capability to a key/value data storage mechanism. Distributed transactional storage is implemented via a hierarchy of transaction managers. "Owning transaction managers" own key/value mappings, while "high-level transaction managers" communicate with the "owning managers," serve as points of contact, and enhance performance through caching. Challenges include optimal (geographical) distribution of "owning" and "high-level" transaction managers.

- ***Reflective Control for an Elastic Cloud Application: An Automated Experiment Workbench***
  Azbayar Demberel, Jeff Chase, and Shivnath Babu, Duke University

A reflective elastic application is a cloud program that can manipulate its own resource requirements based on detailed knowledge of resource availability. Reflective applications can adapt to resource availability, e.g., by deferring computation until resources are more available, and opportunistically exploit excess resources, e.g., by completing deferred computations when resources are more available or cheaper. To understand the needs of reflective applications, the authors created an experimental workbench that can measure effects of various resource allocations on behavior and performance. The output of the workbench is a visualization of the "response surface" that depicts the relationship between input resources and resulting performance. Response surfaces can be efficiently calculated via sampling methods that interpolate response in areas where behavior seems to vary predictably. The audience questioned whether this approach is cost-effective, because of the relatively high cost of experimentation in a production environment.

- ***Toward Cloud-based Collaboration Services***
  David Banks, John S. Erickson, and Michael Rhodes, Hewlett-Packard Labs

Fractal is an open source cloud-based collaboration platform for public information. In Fractal, multiple "tenants" share a common cloud and contribute information that Fractal can coordinate. Fractal streamlines interaction between cloud information spaces through "extensions" that execute whenever data is modified and automatically relate data from different sources. Extensions can create cross-references between spaces, including citation, author, and location lookup, as well as automatic metadata extraction from documents. Extensions are customized for each tenant. While "privacy" is not considered, "content pollution" is a problem; tenants should not be able to alter the behavior of other tenants' content. Challenges include defining the appropriate notion of isolation for tenants, at the physical, virtual, and data levels.

Short Papers

- ***Colocation Games and Their Application to Distributed Resource Management***
  Jorge Londoño, Azer Bestavros, and Shang-Hua Teng, Boston University

The financial feasibility of renting cloud infrastructures can be improved if cloud clients collaborate (or perhaps collude) to share resources. In a market where providers provide fixed-size instances (in memory, storage, and computational speed), co-location by collaboration between cloud clients provides financial benefit. For example, two customers might realize that their applications will "both fit" inside the same virtual instance of some specific service provider. Such co-location can be modeled as a strategic game. The general case of this game has no guarantee of stability, but considering processes (applications) alone leads to a guaranteed (and stable) Nash equilibrium state in which no player can improve personal financial benefit by relocating. The authors propose that because this co-location game has a stable result, this kind of co-location should be supported by location services that help customers find partners, as well as infrastructure to enable migration.

- ***Virtual Putty: Reshaping the Physical Footprint of Virtual Machines***
  Jason Sonnek and Abhishek Chandra, University of Minnesota

Virtual putty refers to a scheme for optimizing the mapping of virtual applications to physical resources. Each physical machine is described in terms of resources and location. Likewise, each virtual instance has a footprint that includes its static resource needs, dynamic resource utilization patterns, and conflicts with other instances. By matching these footprints against one another, one can efficiently utilize physical resources and lower the cost of operations. Challenges include determining parts of the application footprint that are difficult to observe, e.g., dynamic resource utilization. Someone questioned whether the detail in the footprint actually does better than a simple greedy mapping algorithm and asked whether obtaining footprint data might be too expensive to be cost-effective.

- ***Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters***
  Peter Bodík, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, and David Patterson, University of California, Berkeley

Current methods for resource allocation in clouds use simple performance models trained offline, or watermark methods such as increasing resources when a utilization watermark has been met (e.g., "when CPU utilization is greater than 70%, add a core"). It is possible to do better than these methods with a statistical model of behavior, learned dynamically via online experimentation. Based on measurements of end-to-end latency and its variance, a control policy simulator evaluates different policies and tunes model parameters to optimize a reward function. In the case of tuning feedback gain, the model tuning process is shown experimentally to closely approximate optimal behavior.

PANEL

- ***Future Challenges to Cloud Computing***
  Moderator: Amin Vahdat, University of California, San Diego

  Panelists: Garth Gibson, Panasas/Carnegie Mellon University; Stefan Savage, University of California, San Diego; Ben Sigelman, Google; Rich Wolski, University of California, Santa Barbara/Eucalyptus Systems

Garth Gibson, "RAID for Clouds"

Garth Gibson questioned whether we have "the answer" to storage in clouds. He pointed out that common storage methods triplicate every file block, resulting in a 200% storage overhead. He suggested a strategy, called DiskReduce, that replaces duplicates with parity blocks to implement distributed RAID 5. Repair of defective blocks is a background task deferred to times when storage is otherwise idle. The strategy is tuned to perform optimally for realistic file-system contents, where he estimated that 58% of files use eight blocks or less, and 25% of files fit into a single block.

Gibson noted that the true necessary complexity of a storage stack is an unsolved problem and suggested that developing a definitive understanding of complexity in storage is a challenge problem worthy of the Turing Award.

Gibson noted that infrastructure management is now in its third generation. The first generation involved clustering with Beowulf and condor. The second generation introduced virtualization via VMware and Xen. The third generation introduced elasticity. In Gibson's opinion, the fourth generation will reintroduce time-sharing, in service agreements for response time.

Gibson also mentioned several general cloud challenges, including refining the business model, balancing eventual consistency of data against buggy code that needs strong consistency, and a need for testing at scale. We need expensive resources that we can safely "crash and trash."

Stefan Savage, "Are Cloud Privacy and Security Possible?"

Stefan Savage concentrated on security and availability issues in third-party computing. While Infrastructure as a Service (IaaS) clouds leave primary responsibility in the hands of clients, other models of cloud computing assign responsibility for computing and storage to some third party. Implicitly, a cloud client trusts a cloud provider to provide privacy, as well as storage availability, integrity, durability, and retention limits. The cloud provider trusts cloud clients to act in compliance with "acceptable use" policies and to pay promptly and without contest. There is an implicit (and perhaps unfounded) expectation that the cloud provider will monitor clients for appropriate behavior.

Data privacy is a severe problem. A partial solution is "opaque" storage that is encrypted on disk, but key distribution and management remains an unsolved problem. Aside from technical issues, the Stored Communications Act (SCA) grants third-party data less protection than data stored at a first-party site, and it is unclear whether the privacy mechanisms available in clouds are sufficiently strong to satisfy regulations (e.g., HIPAA and PCI). Much less is known when cloud and customer are in different countries.

In a technical sense, Savage noted that data privacy is threatened not only by privileged access, but also by the existence of side-channels through which one customer can determine the transient state of another, e.g., determining transaction volume by observing the timing of cache or memory flushes, or even via the observed behavior of block translation buffers. This gives one customer real-time information on the state of another that can lead to a competitive advantage.

Durability of storage has both technical and legal aspects. How does one "prove" that storage is durable? What happens in case the cloud business fails? In a recent case, a cloud provider deleted 4% of customer information irretrievably, and the customer had no recourse. A year later the company went out of business, and in transferring their data to another company, one-half of all customer information was irretrievably lost with no customer recourse.

Another ambiguity is what is meant by availability. How do you know your provider is a good "steward" of your data? Cloud providers offer "availability zones" but no one knows what they mean. Meanwhile, lack of availability is reimbursed as cost of the service, rather than the cost of the business loss due to lack of availability. There may be a role for the insurance industry in mitigating the risks that arise from this disparity.

Another ambiguity in cloud hosting is the nature of retention. How does a customer know that deleted data is really gone? Supposedly deleted data can be subpoenaed, and the courts have not supported Fifth Amendment rights for encrypted data.

Cloud computing has inherent risks for both client and provider. Clients risk corruption/subversion of VM images, problems of jurisdiction, and inability to verify the privacy of cloud data. For providers, cloud infrastructure is a cyber-criminal's dream world, with plenty of ambiguity and anonymity behind which to hide. What could be more ideal for the cyber-criminal than paying for a huge amount of untraceable computing infrastructure with a stolen credit card?

Ben Sigelman, "The 'Elephant in the Datacenter' and Cloud Monitoring"

Ben Sigelman discussed the problems of monitoring clouds. The "elephant in the data center" is that clouds are actually quite difficult to use. Infrastructure degrades and changes over time, developers move on, and performance of distributed applications is counterintuitive when one understands only the serial version. The failures we observe are only the subset that is visible, making troubleshooting very difficult. These are all evidence that the building blocks we are using for monitoring are wrong. Programming languages haven't adapted. The time spent on seemingly trivial tasks is alarming.

Recent work at Google on monitoring includes distributed "always-on" event tracing, correlated with low-overhead counters for performance monitoring and accounting. Selected events are traced end-to-end, and request-response times can be broken into components and analyzed in detail. Implementing this kind of monitoring requires standardization, including ubiquitous IPC/RPC mechanisms and control-flow libraries. Monitoring is best considered an independent platform in the cloud.

Challenges to cloud monitoring include needs for standardized APIs for monitoring data, as well as ex post facto accounting. Azure/AppEngine-like systems should expose detailed performance info for APIs. For accounting purposes, we do not know the cost of a write until after the write occurs.

Rich Wolski, "The Self-Owned Open-Source Cloud"
Rich Wolski discussed the role of open source in clouds and the relationship between open source self-owned clouds and the current "retail sales" model of cloud service purchase.

Current public clouds are based on a "retail sales" model that quite literally employs the same infrastructure to rent CPUs as to buy DVDs. Public clouds are dependent on customer self-service and a concept of "quality-of-service" that is misnamed a "service-level agreement." Accountability between customer and provider is based on e-commerce. A customer with a problem is treated like a customer who is dissatisfied with a material purchase.

Meanwhile, management models for clouds are just as valid in the self-owned data center as in the cloud, and upcoming challenges in data assimilation from ubiquitous sources, multi-player gaming, and applications for mobile devices require a new level of infrastructure that is present in clouds but not present in current self-owned data centers.

One solution to this problem is the self-owned, open source cloud. Eucalyptus is one of the first enabling technologies for creating one's own clouds. Eucalyptus (an elastic utility computing architecture) is a Linux hosting service that is simple, extensible, commodity-based, and easy for system administrators to install and maintain. Using Eucalyptus, one can emulate first-generation cloud services such as Amazon Advanced Web Services easily and quickly.

Intended uses of Eucalyptus include cloud research, as well as homogenization of existing self-owned IT infrastructure. It is not intended as a replacement for commercial cloud services, but, rather, as an open prototyping environment that enables research and open source development.

Challenges of clouds include federation, privacy, cost, and storage. Federation is a policy mediation problem. "Private" clouds are actually hybrid clouds with both private and public information. Cost of cloud services is increasingly becoming a "first-class" object, in the sense that algorithms are measuring cost and reacting directly. We have not seen "the" cloud storage model yet.

A short discussion followed, in which several questions were raised. Is it even more difficult to have a testbed than to set up a cloud? Panasas never tested hardware at anywhere near the scale that people are purchasing. Cost and incentive models are hard to understand. If you do not believe this, try teaching a cloud computing course to undergraduates. They do not understand that they are spending money until they "see the bill" for what they did during the course. What is the Eucalyptus business model? When one starts a venture-backed company, one bases one's model on serving the enterprise. Eucalyptus will develop and sell customizations that enable enterprise needs.

STORAGE CLOUD AND APPLIANCES

Full Papers

- ***In Search of an API for Scalable File Systems: Under the Table or Above It?***
  Swapnil Patil, Garth A. Gibson, Gregory R. Ganger, Julio Lopez, Milo Polte, Wittawat Tantisiroj, and Lin Xiao, Carnegie Mellon University

Data-intensive scalable computing (DISC) systems, intended to process and store massive data sets, have built their own distributed file systems (e.g., Google File System, Hadoop Distributed File System [HDFS]). By contrast, cluster file systems such as the Parallel Virtual File System (PVFS) have been used to run larger-scale workloads by the High Performance Community (HPC) community for about a decade. The authors explore how to evolve the file system API used by the HPC community so that they can be used for DISC workloads. The authors propose extending traditional cluster file systems to expose block layout to applications, thus allowing applications to co-locate computation with data. The authors built a lightweight shim layer that connects Hadoop and PVFS. Through this shim, they added three functions: read-ahead, co-location of compute with data, and exposing file block layout to applications. Their experiments show that PVFS with the shim layer performs comparably to HDFS. Second, most DISC systems use databases with weaker semantics than traditional databases to store and query metadata. The authors propose a mechanism for using the file system with a filtered directory scan to provide similar functionality.

- ***CloudViews: Communal Data Sharing in Public Clouds***
  Roxana Geambasu, Steven D. Gribble, and Henry M. Levy, University of Washington

Currently, most Web services store and process their data in their own data center. For example, Flickr and Picasa have similar interfaces, but both of them reimplement the software stack from the ground up. With the advent of public cloud services, however, Web services can "rent" themselves to each other, which is made easier by sharing data among co-located services. CloudViews is a storage system that is designed so that services running on a cloud can share data with each other. CloudViews provides database-style views for data sharing between applications. For example, in CloudViews, a Flickr-like service might create a view that shares photos to an automatic photo tagging service but not the ownership information of the photos. The challenges in such a service include providing a scalable protection mechanism, query admission control, and QoS for resource allocation. A member of the audience pointed out that views are good for read-only data and another member asked how CloudViews shares metadata between services. The author replied that both these issues are good material for future research.

- ***Cloud Analytics: Do We Really Need to Reinvent the Storage Stack?***
  Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, and Renu Tewari, IBM Research

MapReduce workloads are generally executed on Internet-scale file systems, such as Google File System (GFS), that do not provide a POSIX interface. The authors explore the suitability of traditional cluster-based file systems for such workloads. In particular, they compare HDFS (an open source implementation of GFS) with IBM's GPFS cluster file system. Compared to GPFS, HDFS provides larger data blocks (on the order of 64MB), allows applications to co-locate computations with data by exposing block locations to applications, and provides data availability in case of node and disk failures.

To verify that they could bridge the gap between HDFS and GPFS, the authors modified GPFS to expose the block location information to MapReduce applications. Second, directly increasing GPFS block size to match that of HDFS is not feasible, as GPFS internally uses block size to perform prefetching. Instead, the authors introduce a new construct called a metablock, which is basically a consecutive set of (smaller) blocks of a file that are allocated on the same disk. The small blocks are used internally by GPFS to perform accounting, prefetching, etc., whereas the larger logical metablock is exposed to MapReduce applications. With these changes, the performance of the modified GPFS and HDFS are comparable. Further, the authors ran experiments to confirm that metablocks do not hurt the performance of GPFS for traditional applications. Thus, clustered file systems, enhanced appropriately, can provide the best of both the traditional applications and MapReduce workloads.

Short Papers

- ***Constructing and Managing Appliances for Cloud Deployments from Repositories of Reusable Components***
  Matthew S. Wilson, rPath, Inc.

The usual way to deploy applications is to start with a base image, install applications, snapshot the image, and then spin up new instances from snapshots. However, these snapshots are hard to move from one provider to another. Automation tools can help, but they require a new setup for each cloud environment. Instead, Matthew Wilson proposes handling software configuration management via a version control system. Dependencies between software components are encoded by grouping components with the components that they require. Once all software is managed and grouped under version control, one can build deployment images from these groups. One member of the audience asked how many companies are using their system. Wilson replied that companies can use their rPath software to do this or can use their rBuilder free online service. About 17,000 projects are using the service, and 50 companies have downloaded rPath. Another audience member asked whether they changed the operating system. Wilson replied that the operating system is changed as little as possible.

- ***Maximizing Efficiency by Trading Storage for Computation***
  Ian F. Adams, Darrell D.E. Long, and Ethan L. Miller, University of California, Santa Cruz; Shankar Pasupathy, NetApp; Mark W. Storer, Pergamum Systems

The authors argue that instead of storing data that is not frequently accessed in the cloud, it can be more cost-efficient to regenerate data on demand. For example, instead of pre-generating various formats of photos (BMP, jpeg, tiff, etc.), it might be more efficient to store photos in the most frequently used format and regenerate other formats on demand. To enable regeneration of data, one needs to record the inputs, processes, and provenance needed to regenerate the data. The decision whether data should be stored or regenerated is determined by cost-benefit analysis. The factors to consider in this analysis include data semantics (i.e., should the exact same data be regenerated or will any data generated by the same process suffice), the cost of regenerating data, and the cost of computing in the cloud in the future.

MAP REDUCE AND CLOUD APPLICATIONS

Full Papers

- ***Mochi: Visual Log-Analysis Based Tools for Debugging Hadoop***
  Jiaqi Tan, Xinghao Pan, Soila Kavulya, Rajeev Gandhi, and Priya Narasimhan, Carnegie Mellon University

Current debugging tools present debugging data at the wrong level of abstraction to be useful in debugging clouds. Mochi, instead, expresses MapReduce program execution in terms of the high-level operations "Map" and "Reduce." It extracts views of node behavior with SALSA, correlates execution traces, and creates a conjoined representation of control and data flow. Control flow consists of the order in which operations are executed, while data flow indicates how the output of one operation is used as an input to others. This conjoined representation is visualized in a number of ways using the R statistics system. The "swimlanes" visualization shows the extent of map and reduce operations in time, so that wedged operations can be detected and addressed. "Realized execution paths" provide a statistical depiction of time spent in each processor state, while data flow depictions show how map and reduce functions relate to one another.

- ***A Common Substrate for Cluster Computing***
  Benjamin Hindman, Andy Konwinski, Matei Zaharia, and Ion Stoica, University of California, Berkeley

NEXUS is a common substrate level that allows several cloud frameworks with differing semantics to co-locate in the same cloud. It can also be used to run several versions of the same framework in one cloud. NEXUS is extremely lightweight and attempts to be a "microkernel" for serving cloud stacks. Performance experiments for a logistic regression machine-learning algorithm show that running Hadoop on top of NEXUS is negligibly slower than running Hadoop alone, but that running the same application on NEXUS alone is several times faster. Since microkernels were not successful, an audience member wondered, why do the authors expect NEXUS to be successful? By the time microkernels were introduced, there were a number of well-established players in the operating systems space, but the cloud is still young and can be changed.

- ***Using Proxies to Accelerate Cloud Applications***
  Jon Weissman and Siddharth Ramakrishnan, University of Minnesota, Twin Cities

A proxy can be utilized to speed up access to cloud services by having superior location or access to relevant resources. In a PlanetLab experiment, proxies were utilized to access 30 commercial Web services. Response times for 70% of queries were improved by proxying, with a 20% performance improvement on average among these. Proxies excel when a cloud application accesses multiple others, which can happen due to specialization of computing infrastructure or data store, distributed data mining, and mash-ups, among others. Open questions include whether to proxy and why, where to optimally locate proxies, and how to select a proxy from those available. The ability of a proxy to cache results or perform local computations has not been explored.

Short Papers

- ***DryadInc: Reusing Work in Large-scale Computations***
  Lucian Popa, University of California, Berkeley; Mihai Budiu, Yuan Yu, and Michael Isard, Microsoft Research, Silicon Valley

A Dryad job is a directed acyclic graph representing data flow in a distributed computation, where each vertex is a computation and each edge represents data flow. A Dryad job or set of jobs often involves redundant calculation of the same result several times. "Identical computation" (IDE) caches and reuses results of repeated computations, while "incremental merging" (MER) employs a user-crafted computation that incorporates new data into the results of a previous computation. The cost-effectiveness of IDE and MER depends on a time/space tradeoff and whether computation time or cache space is more expensive in context.

- ***Towards Optimizing Hadoop Provisioning in the Cloud***
  Karthik Kambatla and Abhinav Pathak, Purdue University; Himabindu Pucha, IBM Research Almaden

Hadoop has hundreds of configurable parameters. Current tools like Hadoop on demand and Cloudera are laborious to use when parameter tuning. One alternative is controlled experimentation. Trying a distributed grep with 1, 4, 8, 16, and 24 map nodes shows diminishing returns after use of 8 map nodes. Thus one can determine an appropriate number of map nodes by direct experimentation. Someone questioned the value of such a method given that such experiments would have to be done "at scale" and expensively in order to guarantee sufficient accuracy.

BSDCan 2009: The Technical BSD Conference

Ottawa, Canada
May 6–9, 2009

Summarized by Royce Williams (royce@tycho.org)

Slides for most of the presentations are available at http://bsdcan.org/2009/.

KEYNOTE ADDRESS

- ***Thinking about Thinking in Code***
  George V. Neville-Neil, Neville-Neil Consulting

In what he described as "a bit of a rant," George Neville-Neil challenged the BSD development community to think about their work in a different way.

Neville-Neil started by attacking the idea that software development is significantly more creative than, for example, automobile manufacturing. He pointed out that there has been little true innovation in graphical user interface design, showing similarities in GUIs ranging from the Xerox PARC user interface through Mac OS X. He discarded traditional explanations such as blaming marketing or that users demand front-end consistency. Even OS internals, he argued, have not substantially changed and do not fundamentally differ among the major families of operating systems. He stated that the languages that we work with truly dictate our work, that features of bad languages (sloppy, unsafe, confusing) lead to code that follows suit, and that making programming languages easier has effectively lowered the quality of code (by lowering the barrier to entry).

In a flurry of frank advice to programmers, Neville-Neil went on to encourage reading good code, working with good programmers (rather than poor ones, which he argues can actually cause your own code to suffer), and refraining from repeatedly reinventing the wheel by recreating low-level constructs (like lists, hashes, and other academic projects). Instead, he suggested reading research papers discriminatingly, exploring unfamiliar code and languages, avoiding too much specialization, and cultivating a willingness to "break things, look stupid, be wrong, learn from others." He warned against hubris, not starting projects, or never finishing them. He pointed out that we all have a finite amount of time to live, so finding people who will tell you when your idea is bad so that you can quickly move on to the next one is very important. In closing, he suggested seeking to reduce complexity, using visualization tools and new data organization methods, and working with safe yet powerful programming languages.

- ***Automating FreeBSD Installations: PXE Booting and install.cfg Demystified***
  Randi Harper, IronPort/Cisco

Randi Harper reviewed some of the common issues with customizing FreeBSD's sysinstall configuration (the install .cfg file) and installing FreeBSD over PXE. A basic walkthrough followed, noting common stumbling blocks along the way (e.g., that trailing whitespace in the install.cfg file can cause problems, and that some variables are case-sensitive). Since install.cfg is not well documented, reading the source was necessary.

Harper covered the entire sysinstall/PXE installation process. Steps included setting up the ISC dhcpd package; configuring supporting services (tftp via inetd, NFS); copying the contents of a FreeBSD installation CD to a staging area; and using mdconfig to mount the included mfsroot image in order to customize the install.cfg file within. Customization options included running in full automatic/unattended mode, specifying a single NIC to use, and optionally specifying packages to add post-install. The current system requires that the NIC type used for installation be known in advance, making it necessary to customize the install.cfg for different hardware families. Harper is working on adding support for a list of multiple NICs, tried in succession, to reduce the number of separate configuration profiles.

During the question period, the topic of how to avoid building the same system twice was raised. Matt Olander of ixSystems asked about the feasibility of knowing the MAC addresses of each system in advance. Harper replied that such asset management is usually already part of large-scale deployments. Olander noted that his environment consists of setting up large groups of systems and then shipping them to the end customer as quickly as possible, making such inventory work infeasible. Discussion followed about keeping a custom text file to incorporate into the dhcpd. conf file to track "state" (the MACs of successfully built systems).

- ***GEOM_SCHED: A Framework for Disk Scheduling within GEOM***
  Luigi Rizzo and Fabio Checconi, University of Pisa

Luigi Rizzo presented GEOM_SCHED, a disk-scheduling framework built on GEOM, the FreeBSD storage abstraction layer. FreeBSD now uses a primitive elevator/C-LOOK-based scheduler. While there has been previous work on disk scheduling in FreeBSD, it has not been committed to the base OS. Rizzo speculated that this might be due to the previous implementations being device-specific and that disk schedulers based on the GEOM framework could make development easier.

In turn, Rizzo examined the merits of each potential location to place a disk scheduler (the disk device, the device driver, and GEOM). He concluded that GEOM is a good option, because it does not require hardware awareness or driver modification, provides a single point of control, and provides for transparent insertion and removal, as well as runtime reconfiguration.

Another design goal was minimal kernel reconfiguration. Since GEOM_SCHED is not included in the base FreeBSD system, the existing implementation dynamically patches g_io_request() to repurpose some unused fields in the structure. It is otherwise implemented entirely outside of the GENERIC kernel as a userland object, a generic kernel module, and one or more kernel modules.

Rizzo went on to cover the GEOM_SCHED API, basic disk-scheduling concepts, and his measurement methodology, especially noting the hazards of measuring disk I/O performance when the caching and read-ahead policies used by drivers and firmware are sometimes not known.

Rizzo's example scheduler was a straightforward implementation of round-robin queues, with anticipation (in which seeks are delayed in case non-seek activity arrives soon after, and then grouped). He presented the results of his testing for various workloads. Even with the slight overhead caused by GEOM, disk performance for multiple greedy readers was significantly improved. He encouraged others to start from this basic framework, applying other algorithms for other workloads. His prototype is available at http:// info.iet.unipi.it/~luigi/FreeBSD/.

Robert Watson asked about the interaction between disk scheduling and process prioritization, referencing previous work that showed that particular I/O patterns (such as an fsck) can suffer in surprising ways when interacting with process scheduling. Rizzo encouraged further research in this area.

- ***Getting Started in Free and Open Source***
  Cat Allman and Leslie Hawthorn, Google

Cat Allman and Leslie Hawthorn took turns presenting ideas in a tag-team fashion. Since there were many in the audience who were decidedly not new to open source, they partially adjusted their talk to address how current members of open source projects can better understand and attract new contributors.

High points included coming as close as possible to "going back to being new" (by vicariously mentoring newcomers); recognizing that thorny problem areas (like bug wrangling) can be opportunities for ways to participate; understanding that FOSS projects are inherently reputation-based economies; designating a "newbie wrangler" (either someone talented in this area or as a rotating responsibility) to protect people from burning out on hand-holding; creating a culture tolerant of failure and mistakes to aid growth; and not assuming that newcomers who make mistakes early will remain permanently clueless.

- ***Updates to the the FreeBSD Problem Reporting System***
  Mark Linimon, Lonesome Dove Computing Services

Mark Linimon, primary "bugmeister" for FreeBSD, proposed an initial conceptual prototype to start work toward a new system of problem reporting (PR) for FreeBSD, working under a grant from the FreeBSD Foundation. In broad terms, lessons learned from the current system will be applied to a temporary prototype to model this new workflow.

Linimon has discovered some specific areas for improvement. Some PR states (e.g., "patched" and "closed") are used consistently, while others ("feedback," "analyzed") are overloaded, which has been confusing enough to throttle PR throughput rates already hampered by resource limitations. Linimon reviewed the workflow categories used by similar frameworks (Bugzilla, Jira, and Trac) and, based on that review, has created distinct stages in the model for triage, submitter coordination, and development work, each of which can be worked by different people with different levels of skill.

As presented, there are other opportunities for improvement. Notifications are too broad, and none of the alternative systems allow developers to limit notifications to specific subsystems of interest or specialty. Current category names were chosen with developers in mind (and can be misunderstood by submitters). Linimon also identified a family of PRs that do not fit easily into the current system, including booting, installation, and performance issues, and proposed a "Usability" category to group them conceptually.

An emerging property of the recent system was that adding tagging support resulted in people using the relevant subsystem man pages as tags. Linimon has added a specific separate field in the prototype that is being populated using the man page names.

Linimon has chosen Jira as the prototype platform (but was careful to note that this does not mean that Jira will be selected). He will be applying these ideas to Jira and seeking feedback. People interested in helping were directed to the FreeBSD wiki's BugBusting page to coordinate.

- ***scrypt: A New Key Derivation Function***
  Colin Percival, Tarsnap

To provide some background, Percival started with an overview of encryption key derivation functions. KDFs are commonly used to hash passwords for secure storage and to generate cryptographic keys. Examples include the classic DES CRYPT, Poul-Henning Kamp's iterated MD5 CRYPT, PBKDF2, and bcrypt.

These (and most other) preceding KDFs have focused on raising the cost of "dollar-hours" by maximizing the amount of CPU time required to run. However, well-funded groups can afford farms of custom dedicated ASICs, each with thousands of cores optimized for specific cryptographic operations.

Percival's tagline for this presentation was "Doing our best to thwart TLAs with ASICs." Percival noted that the cost of an ASIC is roughly matched with its size and that large amounts of RAM can take up a significant amount of ASIC space. He reasoned that functions which both require very large amounts of RAM ("memory-hard" functions) and are not easily broken down to run in parallel ("sequential" functions) would increase the required size (and number) of dedicated ASICs, thereby significantly increasing the corresponding cost.

To enable such functions, Percival introduced a provably sequential memory-hard problem, ROMix, which fills a hash table with pseudo-random values and then accesses them in a pseudo-random order. In the accompanying paper Percival proves that any algorithm that correctly implements ROMix will be sequential memory-hard, but in the talk he left a review of the two-page proof as an exercise for interested listeners.

Percival then presented scrypt itself, which is a combination of PBKDF2, an algorithm that solves a given ROMix problem, HMAC-SHA256, and Daniel J. Bernstein's Salsa20 cipher to carry out the key derivation while also quickly requiring large amounts of RAM. Like other KDFs, scrypt takes parameters that can be used to adjust its costs to run, so the maximum amount of RAM and maximum time in seconds can be varied.

In order to illustrate the difference in strength, Percival estimated some real-world costs. Since entities in the business of brute-force attacks do not publish hardware costs, Percival also presented the assumptions he used for comparing algorithms. He noted that, even if off by orders of magnitude, these estimates nevertheless held constant among the functions and therefore are useful for relative comparisons.

Using the provided numbers and stating the parameters used (for the functions that take them), Percival compared DES CRYPT, MD5 (as a reference point, not useful for actual encryption purposes), MD5 CRYPT, PBKDF2, bcrypt, and scrypt. For example, an 8-character password with good entropy, encrypted with scrypt in .1 seconds (good enough for authentication speeds), will take roughly $4.8M to crack within one year, while bcrypt would cost only $130K. For longer times suitable for encrypting data (around 5 seconds), the 8-character costs jump to $4.3M for bcrypt (3.0s) and $19B for scrypt (3.8s).

While doing research for this work, Percival discovered that OpenSSL uses simple MD5 hashing as its key derivation function, and OpenSSH also uses simple MD5 for keyfile passphrases. This may be of some concern for people who carry their SSH keys on pocket USB devices.

During the question period, Brooks Davis asked about the feasibility of imposing a large per-transaction memory cost on systems used for high-volume authentication or which are subject to authentication floods. Percival replied that the function takes a number of parameters to adjust the CPU and memory cost of each calculation to fit the target platform and work load.

Percival's paper describing scrypt in more detail (including proofs), the full estimate comparison, and a cross-platform BSD-licensed implementation of scrypt are all available at tarsnap.com/scrypt/.

- ***Works in Progress Session (also known as the "lightning round")***
  Chaired by Robert Watson

Colin Percival announced that he is interested in reviving the project to build concurrency-awareness into the FreeBSD rc.d system. Now that multicore systems are the norm, startup times could be significantly improved. Interested contributors are encouraged to contact Colin.

Scott Ullrich of the pfSense Project outlined features of the BSD Installer, a proposed unified installer for all BSDs, and the installer used by Dragonfly BSD and the upcoming version of pfSense. Features include a clear separation between the front end and back end, enabling multiple possible front ends. Recent work is focusing on adding the remaining functionality included in FreeBSD's sysinstall and the PC-BSD installer, but missing from the BSD installer.

Philip Mullis briefly mentioned a new effort to create an independent VoIP peering exchange. More information will eventually be available at nopstn.net.

Zach Loafman of Isilon described kernel fault injection, a new set of APIs used to insert specific user-controlled errors at particular points in FreeBSD code ("failpoints"). The APIs include the ability to assign the errors' probabilities of occurring or a cap on how many times they occur. Isilon is using hundreds of these "failpoints" in production, and Loafman is working to get support for them committed to FreeBSD.

John Baldwin first gave a status report about the upcoming FreeBSD 8.0. New features include virtual network stacks, MIPS support, NFSv4, ECMP (which enables support for kernel awareness of multiple routing tables and default routes), virtual wireless access points, a reworked USB stack, support for 32-bit FreeBSD 8 as a Xen dom-U guest, and improved Linux binary compatibility. FreeBSD 8.0 is scheduled for release at the end of August.

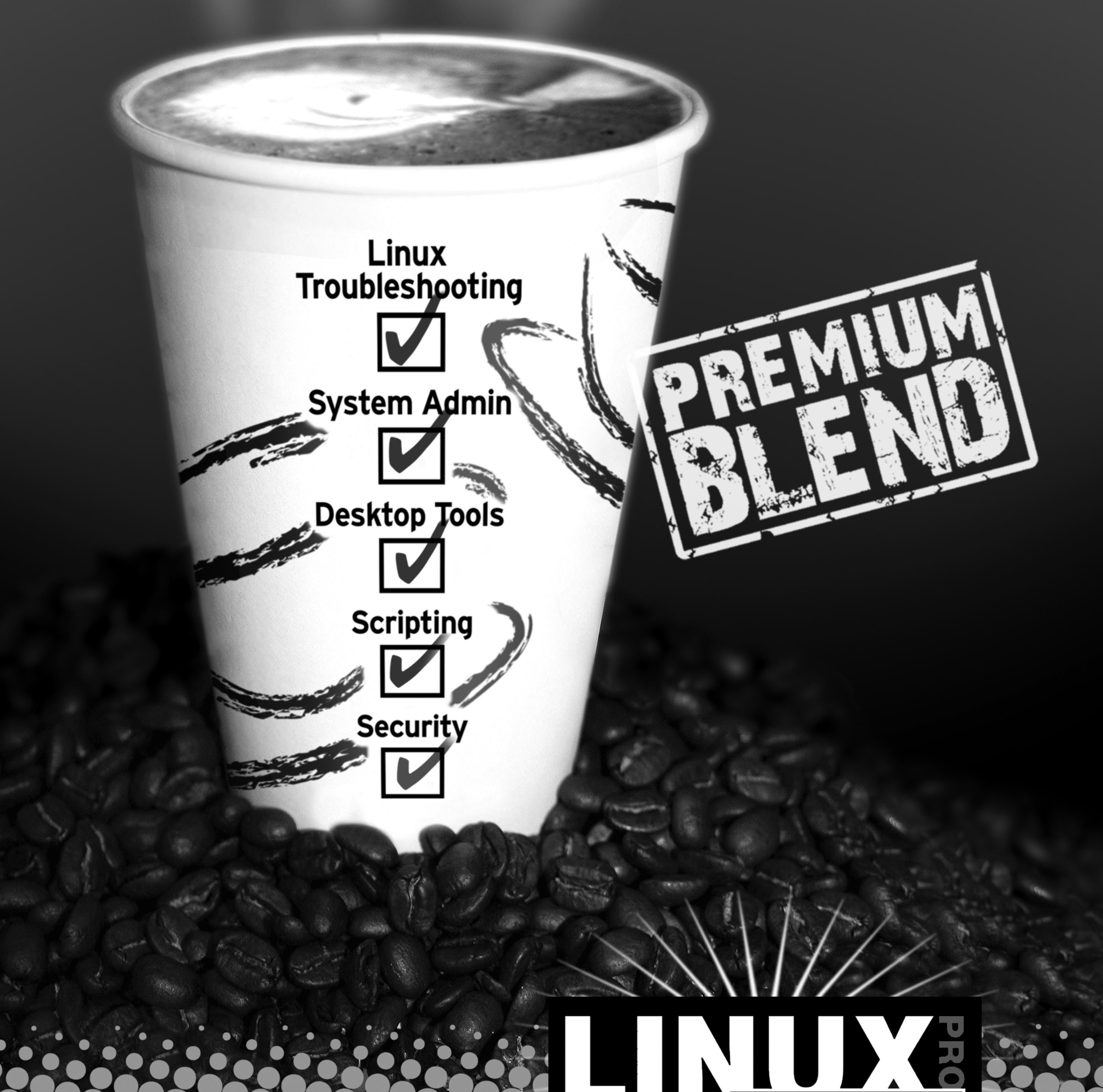
Baldwin also talked briefly about extensions to device mmap() support, largely driven by the memory-mapping needs of modern GPUs. In the amd64 and i386 ports, this will be implemented via PAT (Page Attribute Table, required for good PCI-Express performance) and will pave the way for an Nvidia amd64 driver for FreeBSD.

John Birrell of Juniper Networks described jbuild, a new FreeBSD build system that eliminates multiple layers of redundant dependency calculation, significantly reducing build times. This is accomplished by front-loading the master jbuild process with all dependency information from the build directory.

Doug Rabson presented updates about FreeBSD on Xen. The included XEN and XENHVM kernel configs in FreeBSD-current are the best place to start experimenting with paravirtualization and hardware virtualization, respectively.

Rabson also gave a quick how-to about booting from ZFS and mapped out his planned future ZFS work, including teaching the FreeBSD installer about ZFS.

Warner Losh talked about recent progress with the various flavors of the MIPS port, working on the RMI XLR/XLS, RMI Alchemy, Cavium Octeon1, and Atheros AR71xx/91xx chips, using reference boards supplied by various sources.



LINUX PRO MAGAZINE

If you like the taste of Linux, why not treat yourself to the best?

Linux Pro Magazine delivers real-world solutions for the technical reader. In every issue, you'll find advanced techniques for configuring and securing Linux systems. Learn about the latest tools and discover the secrets of the experts in Linux Pro. Each issue includes a full Linux distribution on DVD.

www.linuxpromagazine.com

USENIX

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

Come to LISA '09 for training and face time with experts in the sysadmin community.

Keynote Address by Werner Vogels, CTO, Amazon.com

Topics include: Virtualization, Solaris, Security, Cloud Computing, and more!

Register Today!

LISA'09

SPONSORED BY USENIX & [sage] IN COOPERATION WITH LOPSA & SNIA

23RD LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

November 1–6, 2009, Baltimore, MD

Discounts Available!

http://www.usenix.org/lisa09/lg