usenix ;login:

OCTOBER 2011 VOL. 36, NO. 5

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION usenix UPCOMING EVENTS

23rd ACM Symposium on Operating Systems Principles (SOSP 2011)

SPONSORED BY ACM SIGOPS IN COOPERATION WITH USENIX

October 23–26, 2011, Cascais, Portugal
http://sosp2011.gsd.inesc-id.pt

ACM Symposium on Computer Human Interaction for Management of Information Technology (CHIMIT 2011)

SPONSORED BY ACM IN ASSOCIATION WITH USENIX

December 4–5, 2011, Boston, MA
http://chimit.acm.org/

25th Large Installation System Administration Conference (LISA '11)

SPONSORED BY USENIX IN COOPERATION WITH LOPSA

December 4–9, 2011, Boston, MA
http://www.usenix.org/lisa11

ACM/IFIP/USENIX 12th International Middleware Conference (Middleware 2011)

SPONSORED BY ACM AND IFIP IN ASSOCIATION WITH USENIX

December 12–16, 2011, Lisbon, Portugal
http://2011.middleware-conference.org/

10th USENIX Conference on File and Storage Technologies (FAST '12)

SPONSORED BY USENIX IN COOPERATION WITH ACM SIGOPS

February 14–17, 2012, San Jose, CA
http://www.usenix.org/fast12

9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12)

SPONSORED BY USENIX IN COOPERATION WITH ACM SIGCOMM AND ACM SIGOPS

April 25–27, 2012, San Jose, CA
http://www.usenix.org/nsdi12

2012 USENIX Federated Conferences Week

June 12–15, 2012, Boston, MA, USA
http://www.usenix.org/fcw12

2012 USENIX Annual Technical Conference (USENIX ATC '12)

June 13–15, 2012, Boston, MA
http://www.usenix.org/atc12
Paper titles and abstracts due January 10, 2012

21st USENIX Security Symposium (USENIX Security '12)

August 6–10, 2012, Bellevue, WA

10th USENIX Symposium on Operating Systems Design and Implementation (OSDI '12)

October 7–10, 2012, Hollywood, CA

26th Large Installation System Administration Conference (LISA '12)

December 9–14, 2012, San Diego, CA

FOR A COMPLETE LIST OF ALL USENIX AND USENIX CO-SPONSORED EVENTS, SEE HTTP://WWW.USENIX.ORG/EVENTS

usenix ;login:

OCTOBER 2011, VOL. 36, NO. 5

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Casey Henderson
Jane-Ellen Long
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.

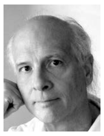
$90 of each member's annual dues is for a subscription to *;login:*. Subscriptions for nonmembers are $125 per year. Periodicals postage paid at Berkeley, CA, and additional offices.

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

OPINION

Musings

RIK FARROW



Rik is the editor of *;login:*.
rik@usenix.org

Once upon a time, Anonymous meant just that. Today, at least for the security conscious, Anonymous means part of the anti-security underground. In particular, Anonymous has been a group of politically aware hackers who have been very good at embarrassing large organizations and corporations by exposing their data.

I mentioned Anonymous (and LulzSec) in my August 2011 column, but a couple of things got me thinking about them again. The first was a short article [1] written by an author I have worked with, Adam Penenberg, about some research on plagiarism published by another friend.

It seems incredible to me that people would dare to plagiarize online materials and call it their own work when all it takes to unmask them is the time and focus necessary to do a bit of searching. Jericho, one of the founders of Attrition.org, began investigating the published work of security charlatans and quickly discovered entire books that had been plagiarized [2].

Adam's article lists other security books where most of the content was copy-and-pasted from online sources. But Adam's article also got me thinking about security. He starts his article by writing about how common it is to copy other people's code, with implicit permission, and add it to your own code. Copying of working code has a long history, and it certainly makes it possible to build your own software faster. The problem with this quick approach is that you may be copying flaws that result in exploitation of your software, or you may quickly build Internet-facing Web applications that you don't understand very well and have failed to secure properly.

And that gets me to the Anonymous connection. A private source had pointed out that the publicized, that is, successful, attacks by Anonymous and LulzSec were using well-known penetration testing tools: Web Application Attack and Audit Framework (W3AF) [3]; sqlmap (for SQL injection of Web apps or databases) [4]; and Nikto, a Web security scanner with roots in the '90s [5]. All of these tools are covered by Snort signatures, implying that noticing that your Internet-facing sites are being probed is easy—if you are paying attention. It also suggests that you could be doing the probing yourself: the tools are free, although you do need to learn how to use them.

None of this is rocket science. Far from it. Just as guns make murder easier, tools like these make hacking much easier. If you don't scan your own mass of Internet-facing code, sooner or later someone will do it for you. I strongly encourage you to do your own scanning, as well as to follow good programming practices [6] and watch for attacks on your servers using IDS or IPS (which could at least block the scanners' IP addresses).

The Lineup

I am cutting my own column short this month, because this issue has the reports from USENIX Federated Conferences Week, and those alone consume a huge portion of the printed version of *;login:*. I hope you appreciate the reports as much as I do, as a good report can tell you a lot more about a paper than the abstract can (or will). By reading reports, you can catch up on current research, or decide to go to the USENIX Web site for the event to watch or listen to a recorded invited talk or panel.

We have several excellent articles in this issue, starting with Greg Burd explaining NoSQL. I am excited about Greg's article, not just because he explains NoSQL clearly, but because he explains why NoSQL became necessary. Greg compares relational databases with NoSQL, as well as contrasting different types of NoSQL databases with each other. As I was reading, it became really obvious to me why Oracle needed to buy Sun, but that's just an aside. Margo Seltzer suggested Greg as an author and also read the near-final draft, which helped in polishing this article.

I contacted Paul Vixie when something he had posted suggested that there are other potential uses for DNSSEC. In the August 2011 *;login:*, Peter Gutmann had pointed out how easy it is to spoof SSH fingerprints, something I wasn't aware of. Paul's post suggested a simple fix for this if you are using DNSSEC. Paul's article describes other real and potential uses for DNSSEC, with the ability to securely use self-signed certificates high on the list.

Don Revelle's article answers questions about virtualization technology that I have been asking for a long time. I have helped publish a handful of virtualization articles in *;login:*, but none that covers the breadth of the technologies, explaining how they differ, the way Don's does. The point of this article is not to get you to change your choice for virtualization, but to understand how the technologies differ.

The final article in this issue explains the thinking behind changing the whois protocol. Andy Newton (ARIN), Dave Piscitello (ICANN), Benedetto Fiorelli (RIPE), and Steve Sheng (ICANN) have written about how the whois protocol, and client software, performs poorly for an Internet that is both international in scope and embraces several regional authorities that have differed in how they present whois data. Whois is supposed to work uniformly today, but it does not. They present a new interface for whois servers that supports both Web browsers and new clients, as well as internationalization.

David Blank-Edelman has focused his steely eyes (actually, they are quite a bit more friendly than that) on performance. David shows how to use two different tools that can help you pinpoint performance issues in your Perl scripts. Fixing them is another issue, but David makes it easy to find where the problem may be.

Dave Josephsen tries out a new approach to his column: he interviews a provider of a hosted monitoring service. Dave asks Theo Schlossnagle about his company's (OmniTI) Circonus service and about the future of monitoring systems in general.

Peter Galvin has both a column and a book review in this issue. In his column, Peter continues on the cloud theme from the August 2011 *;login:*, explaining issues for enterprises considering using some form of cloud.

Robert Ferrell also decided to take a hard look at virtualization for this issue, coming up with a parable about how virtualization has disturbed the firmament.

Elizabeth Zwicky has four book reviews this time, along with two from Sam Stover, one from Trey Darley, and one from Peter Galvin.

Tools and frameworks have it easier than ever to write code—but to secure it. In the best of all possible worlds, tools would produce only secure code. In the real world, you must either audit and assess your own code or take responsibility for your organization facing embarrassment and possibly lawsuits.

References

[1] Adam Penenberg, "When Hacks Attack: The Computer Security Textbook Plagiarism Epidemic," *Fast Company*, July 27, 2011: http://www.fastcompany.com/1769244/plagiarism-professionals.

[2] Example of Jericho's work in uncovering plagiarism: http://attrition.org/errata/charlatan/gregory_evans/spyware_reference_study_guide.html.

[3] W3AF (Web Application Attack and Audit Framework): http://w3af.sourceforge.net.

[4] sqlmap (automatic SQL injection and database takeover tool): http://sqlmap.sourceforge.net/.

[5] Nikto, Web server scanner in Perl: http://cirt.net/nikto2.

[6] A good place to get started is the Open Web Application Security Project: https://www.owasp.org/index.php/Getting_Started.

NoSQL

GREG BURD

Greg Burd is a Developer Advocate for Basho Technologies, makers of Riak. Before Basho, Greg spent nearly ten years as the product manager for Berkeley DB at Sleepycat Software and then at Oracle. Previously, Greg worked for NeXT Computer, Sun Microsystems, and KnowNow. Greg has long been an avid supporter of open source software.
greg@basho.com

Choosing between databases used to boil down to examining the differences between the available commercial and open source relational databases. The term "database" had become synonymous with SQL, and for a while not much else came close to being a viable solution for data storage. But recently there has been a shift in the database landscape. When considering options for data storage, there is a new game in town: NoSQL databases. In this article I'll introduce this new category of databases, examine where they came from and what they are good for, and help you understand whether you, too, should be considering a NoSQL solution in place of, or in addition to, your RDBMS database.

What Is NoSQL?

The only thing that all NoSQL solutions providers generally agree on is that the term "NoSQL" isn't perfect, but it is catchy. Most agree that the "no" stands for "not only"—an admission that the goal is not to reject SQL but, rather, to compensate for the technical limitations shared by the majority of relational database implementations. In fact, NoSQL is more a rejection of a particular software and hardware architecture for databases than of any single technology, language, or product. Relational databases evolved in a different era with different technological constraints, leading to a design that was optimal for the typical deployment prevalent at that time. But times have changed, and that once successful design is now a limitation. You might hear conversations suggesting that a better term for this category is NoRDBMS or half a dozen other labels, but the critical thing to remember is that NoSQL solutions started off with a different set of goals and evolved in a different environment, and so they are operationally different and, arguably, provide better-suited solutions for many of today's data storage problems.

Why NoSQL?

NoSQL databases first started out as in-house solutions to real problems in companies such as Amazon Dynamo [1], Google BigTable [2], LinkedIn Voldemort [3], Twitter FlockDB [4], Facebook Cassandra [5], Yahoo! PNUTS [6], and others. These companies didn't start off by rejecting SQL and relational technologies; they tried them and found that they didn't meet their requirements. In particular, these companies faced three primary issues: unprecedented transaction volumes, expectations of low-latency access to massive datasets, and nearly perfect service availability while operating in an unreliable environment. Initially, companies tried the traditional approach: they added more hardware or upgraded to faster hardware as it became available. When that didn't work, they tried to scale existing relational solutions by simplifying their database schema, de-normalizing the schema, relaxing durability and referential integrity, introducing various query caching layers, separating read-only from write-dedicated replicas, and, finally, data partitioning in an attempt to address these new requirements. Although each of these techniques extended the functionality of existing relational technologies, none fundamentally addressed the core limitations, and they all introduced additional overhead and technical tradeoffs. In other words, these were good band-aids but not cures.

A major influence on the eventual design of NoSQL databases came from a dramatic shift in IT operations. When the majority of relational database technology was designed, the predominant model for hardware deployments involved buying large servers attached to dedicated storage area networks (SANs). Databases were designed with this model in mind: They expected there to be a single machine with the responsibility of managing the consistent state of the database on that system's connected storage. In other words, databases managed local data in files and provided as much concurrent access as possible given the machine's hardware limitations. Replication of data to scale concurrent access across multiple systems was generally unnecessary, as most systems met design goals with a single server and reliability goals with a hot stand-by ready to take over query processing in the event of master failure. Beyond simple failover replication, there were only a few options, and they were all predicated on this same notion of completely consistent centralized data management. Technologies such as two-phase commit and products such as Oracle's RAC were available, but they were hard to manage, very expensive, and scaled to only a handful of machines. Other solutions available included logical SQL statement-level replication, single-master multi-replica log-based replication, and other home-grown approaches, all of which have serious limitations and generally introduce a lot of administrative and technical overhead. In the end, it was the common architecture and design assumptions underlying most relational databases that failed to address the scalability, latency, and availability requirements of many of the largest sites during the massive growth of the Internet.

Given that databases were centralized and generally running on an organization's most expensive hardware containing its most precious information, it made sense to create an organizational structure that required at least a 1:1 ratio of database administrators to database systems to protect and nurture that investment. This, too, was not easy to scale, was costly, and could slow innovation.

A growing number of companies were still hitting the scalability and performance wall even when using the best practices and the most advanced technologies of the time. Database architects had sacrificed many of the most central aspects of a relational database, such as joins and fully consistent data, while introducing many complex and fragile pieces into the operations puzzle. Schema devolved from many interrelated fully expressed tables to something much more like a simple key/value look-up. Deployments of expensive servers were not able to keep up with demand. At this point these companies had taken relational databases so far outside their intended use cases that it was no wonder that they were unable to meet performance requirements. It quickly became clear to them that they could do much better by building something in-house that was tailored to their particular workloads. These in-house custom solutions are the inspiration behind the many NoSQL products we now see on the market.

NoSQL's Foundations

Companies needed a solution that would scale, be resilient, and be operationally efficient. They had been able to scale the Web (HTTP) and dynamic content generation and business logic layers (Application Servers), but the database continued to be the system's bottleneck. Engineers wanted databases to scale like Web servers—simply add more commodity systems and expect things to speed up at a nearly linear rate—but to do that they would have to make a number of tradeoffs. Luckily, due to the large number of compromises made when attempting to scale their existing relational databases, these tradeoffs were not so foreign or distasteful as they might have been.

Consistency, Availability, Partition Tolerance (CAP)

When evaluating NoSQL or other distributed systems, you'll inevitably hear about the "CAP theorem." In 2000 Eric Brewer proposed the idea that in a distributed system you can't continually maintain perfect consistency, availability, and partition tolerance simultaneously. CAP is defined by Wikipedia [7] as:

Consistency: all nodes see the same data at the same time
Availability: a guarantee that every request receives a response about whether it was successful or failed
Partition tolerance: the system continues to operate despite arbitrary message loss

The theorem states that you cannot simultaneously have all three; you must make tradeoffs among them. The CAP theorem is sometimes incorrectly described as a simple design-time decision—"pick any two [when designing a distributed system]"—when in fact the theorem allows for systems to make tradeoffs at run-time to accommodate different requirements. Too often you will hear something like, "We trade consistency (C) for AP," which can be true but is often too broad and exposes a misunderstanding of the constraints imposed by the CAP theorem. Look for systems that talk about CAP tradeoffs relative to operations the product provides rather than relative to the product as a whole.

Relaxing ACID

Anyone familiar with databases will know the acronym ACID, which outlines the fundamental elements of transactions: atomicity, consistency, isolation, and durability. Together, these qualities define the basics of any transaction. As NoSQL solutions developed it became clear that in order to deliver scalability it might be necessary to relax or redefine some of these qualities, in particular consistency and durability. Complete consistency in a distributed environment requires a great deal of communication involving locks, which force systems to wait on each other before proceeding to mutate shared data. Even in cases where multiple systems are generally not operating on the same piece of data, there is a great deal of overhead that prevents systems from scaling.

To address this, most NoSQL solutions choose to relax the notion of complete consistency to something called "eventual consistency." This allows each system to make updates to data and learn of other updates made by other systems within a short period of time, without being totally consistent at all times. As changes are made, tools such as vector clocks are used to provide enough information to reason about the ordering of those changes based on an understanding of the causality of the updates. For the majority of systems, knowing that the latest consistent information will eventually arrive at all nodes is likely to be enough to satisfy design requirements.

Another approach is optimistic concurrency control, using techniques such as multi-version concurrency control (MVCC). Such techniques allow for consistent reading of data in one transaction with concurrent writing in another transaction but do not address write conflicts and can introduce more transaction retries when transactions overlap or are long-running.

Both eventual consistency and MVCC require that programmers think differently about the data they are managing in the application layer. Both introduce the potential that data read in a transaction may not be entirely up-to-date even though it may be consistent.

Achieving durability has long been the bottleneck for database systems. It's easy to understand why writing to disk slows down the database; disk access times are orders of magnitude slower than writes to memory. Most database solutions recognize the potential performance tradeoffs related to durability and offer ways to tune writes to match application requirements, balancing durability against speed. Be careful, as this can mean leaving a small window of opportunity where seemingly committed transactions can be lost under certain failure conditions. For example, some will consider an update durable when it has been written into the memory of some number of systems. If those systems were to lose power, that commit would be lost. Network latencies are much faster than disk latencies, and by having data stored on more than one system the risk of losing information due to system failure is much lower. This definition of durability is very different from what you've come to expect from a relational database, so be cautious. For instance, Redis, MongoDB, HBase, and Riak range from minimally durable to highly durable, in that order. As with any other database, when evaluating NoSQL solutions, be sure to know exactly what constitutes durability for that product and how that impacts your operational requirements.

Data and Access Model

The relational data model with its tables, views, rows, and columns has been very successful and can be used to model most data problems. By using constraints, triggers, fine-grained access control, and other features, developers can create systems that enforce structure and referential integrity and that secure data. These are all good things, but they come at a price. First, there is no overlap in the data representation in SQL databases and in programming languages; each access requires translation to and from the database. Object to relational mapping (ORM) solutions exist to transparently transform, store, and retrieve object graphs into relational databases, and although they work well, they introduce overhead into a process and slow it down further. This is an impedance mismatch that introduces overhead where it is least needed. Second, managing global constraints in a distributed environment is tricky and involves creating barriers (locks) to coordinate changes so that these constraints are met. This introduces network overhead and sometimes can stall progress in the system.

NoSQL solutions have taken a different approach. In fact, NoSQL solutions diverge quite a bit from one another as well as from the RDBMS norm. There are three main data representation camps within NoSQL: document, key/value, and graph. There is still a fairly diverse set of solutions within each of these categories. For instance, Riak, Redis, and Cassandra are all key/value databases, but with Cassandra you'll find a slightly more complex concept, based on Google's BigTable, called "column families," which is very different from the more SimpleDB-like "buckets containing key/value pairs" approach of the other two.

Document-oriented databases store data in formats that are more native to the languages and systems that interact with them. JavaScript Object Notation (JSON) and its binary encoded equivalent, BSON, are used as a simple dictionary/array representation of data. MongoDB stores BSON documents and provides a JSON-encoded query syntax for document retrieval. For all intents and purposes, JSON has replaced most of the expected use cases for XML, and although XML has other advantages (XQuery, typed and verifiable schema), JSON is the de facto Web-native format for data on the wire and now can be stored, indexed, and queried in some NoSQL databases.

Neo4j is a graph-based NoSQL database that stores information about nodes and edges and provides simple, highly optimized interfaces to examine the connectedness of any part of the graph. With the massive growth of social networking sites and the value of understanding relationships between people, ideas, places, etc., this highly specialized subset of data management has received a great deal of attention lately. Look for more competitors in this space as the use cases for social networks continue to grow.

Global constraints are generally not available or very rudimentary. The reasoning for not introducing more than basic constraints is simple: anything that could potentially slow or halt the system violates availability, responsiveness, and latency requirements of these systems. This means that applications using NoSQL solutions will have to build logic above the database that monitors and maintains any additional consistency requirements.

A crucial part for the success of the relational market as a whole has been the relative uniformity of SQL solutions. Certainly there are differences between vendors and a non-trivial penalty when migrating from one SQL solution to another, but at a high level, they all start with the same APIs (ODBC, JDBC, and SQL), and that allows a universe of tools and a population of experts to flourish. NoSQL solutions also diverge when it comes to access, encoding, and interaction with the server. While some NoSQL products provide a HTTP/REST API, others provide simple client libraries or network protocols that use widely available data encoding libraries such as Thrift and Protobufs. Some provide multiple methods of access, and the more successful NoSQL solutions will generally have pre-built integrations with most of the popular languages, frameworks, and tools, so that this is not as big an issue as it may seem.

This diversity is representative of the fact that NoSQL is a broad category where there are no standards such as SQL to unify vendors. Customers should therefore choose carefully—vendor lock-in is a given at this point in the NoSQL market. That said, most leading NoSQL solutions are open source, which does defray some of the risk related to the sponsoring company changing hands or going out of business. In the end, though, moving from one NoSQL solution to another will be a time-consuming mistake that you should try at all costs to avoid.

Distributed Data, Distributed Processing

NoSQL solutions are generally designed to manage large amounts of data, more than you would store on any single system, and so all generally have some notion of partitioning (or sharding) data across the storage found on multiple servers rather than expecting a centrally connected SAN or networked file system. The benefits of doing this transparently are scalability and reliability. The additional reliability comes when partitions overlap, keeping redundant copies of the same data at multiple nodes in the system. Not all NoSQL systems do this. The drawback will be some amount of duplicated data and costs associated with managing consistency across these partitions. In addition it is critical to understand the product's approach to distributing data. Is it a master/replica, master/master, or distributed peers? Where are the single points of failure? Where are there potential bottlenecks? When reviewing the NoSQL solutions it's important not to gloss over the details, and be sure to run extensive in-house tests where you introduce all manner of failure conditions into your testbed. The good NoSQL solutions will prove resilient even when operating in punishing environments, whereas the less mature may lose data or stop operating when too many unforeseen conditions arise.

In addition to distributing data, many NoSQL solutions offer some form of distributed processing, generally based on MapReduce. This can be a powerful tool when used correctly, but again the key when evaluating NoSQL solutions is to dig into the details and understand exactly what a vendor means by "we support MapReduce."

NoSQL in Practice

There are many products that now claim to be part of the NoSQL database market, far too many to mention here or describe in any detail. That said, there are a few with which you should become familiar, as they are likely to become long-term tools everyone uses. Let's examine the leading products broken down by category.

	MongoDB	**CouchDB**	**Riak**	**Redis**	**Voldemort**	**Cassandra**	**HBase**
Language	C++	Erlang	Erlang	C++	Java	Java	Java
License	AGPL	Apache	Apache	BSD	Apache	Apache	Apache
Model	Document	Document	Key/value	Key/value	Key/value	Wide Column	Wide Column
Protocol	BSON	HTTP/REST	HTTP/REST or TCP/ Protobufs	TCP		TCP/Thrift	HTTP/REST or TPC/Thrift
Storage	Memory mapped b-trees	COW-BTree	Pluggable: InnoDB, LevelDB, Bitcask	In memory, snapshot to disk	Pluggable: BDB, MySQL, in-memory	Memtable/ SSTable	HDFS
Inspiration		Dynamo	Dynamo		Dynamo	BigTable, Dynamo	BigTable
Search	Yes	No	Yes	No	No	Yes	Yes
MapReduce	Yes	No	Yes	No	No	Yes	Yes

Here are a few company use cases where NoSQL is a critical component of the data architecture.

Facebook: HBase for Messages

When Facebook decided to expand its messaging services infrastructure [8] to encompass email, text, instant messages, and more, it knew it had to have a distributed fault-tolerant database able to manage petabytes of messages and a write-dominated workload. It needed something to handle their existing service of "over 350 million users sending over 15 billion person-to-person messages per month" and a chat service that "supports over 300 million users who send over 120 billion messages per month," and to allow plenty of room for growth. Although they considered MySQL, a solution with which they have extensive experience [9], they created Cassandra [10]—an open source project combining elements of Google's BigTable [2] and Amazon's Dynamo [1] designs—and Apache HBase, a distributed database that closely mimics Google's BigTable implementation and is tightly integrated with Apache Hadoop and ZooKeeper. Facebook favored the strong consistency model, automatic failover, load-balancing, compression, and MapReduce support of HBase for their new production messaging service.

Craigslist: MongoDB for Archived Postings

Craigslist recently migrated over 2 billion archived postings from its MySQL clusters into a set of replicated MongoDB servers [11]. They still use MySQL for all active postings on their site, but these days when you log in and review old posts that are now expired, you are accessing their new MongoDB-based service. For Craigslist, scalability and reliability were central requirements, but one of the more interesting features they gained by going to MongoDB was schema flexibility. Their MySQL databases were so large that any schema change (ALTER TABLE) would take around two months to complete across the replicated database set. Contrast that with MongoDB, where data is stored as JSON documents with no schema enforcement at all. By separating archived postings from live postings, Craigslist simplified their architecture and made it easier to change their production schema as requirements changed.

Conclusions and Advice

The SQL vs. NoSQL debate will continue, and both sides will benefit from this competition in a market that was stagnant for far too long. I think that as the dust settles it will become evident that NoSQL solutions will work alongside SQL solutions, each doing what they do best. Facebook, Twitter, and many other companies are integrating NoSQL databases into their infrastructure right alongside SQL databases. Each has its strengths and weaknesses; neither will entirely displace the other. Some future SQL databases may start to take on features only found in NoSQL, such as elasticity and an ability to scale out to large amounts of commodity hardware. The demand for SQL will not go away anytime soon, nor will the reality of today's more distributed, virtualized, and commodity-based IT infrastructure. As more companies begin capturing and analyzing an increasing amount of data about their customers, the need for databases that can efficiently manage and analyze that kind of data will only grow.

The key to making a good decision in this market is to remain as objective and open-minded as possible while evaluating these products and talking to their vendors. Recognize that this is a new market, and that right now the market leaders are working hard to cement their positions. Use that to your advantage by asking hard questions and expecting detailed answers. Never decide on a solution without having first put that vendor's product through rigorous testing using your data on your systems. Be sure to interact with and learn more about the vendor; they are all interested in building relationships with you and will generally go out of their way to help you with their product. Spend some time understanding their particular views on eventual consistency, durability, CAP, and replication, and be certain you understand how those tradeoffs will impact your design. Make sure that they support enough tools to meet your needs today and tomorrow. Look for a solid community to bolster your development. In the end, be sure that the solution you pick is going to support you as your product matures. Make sure the vendor has shared their road map and explained how they will help you move from one version to the next, because this area of technology is far from static.

References

[1] Amazon Dynamo: http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html.

[2] Google BigTable: http://labs.google.com/papers/bigtable.html.

[3] LinkedIn Voldemort: http://project-voldemort.com/.

[4] Twitter FlockDB: http://engineering.twitter.com/2010/05/introducing-flockdb.html.

[5] Facebook Cassandra: http://cassandra.apache.org/.

[6] Yahoo! PNUTS: http://www.brianfrankcooper.net/pubs/pnuts.pdf.

[7] CAP theorem: http://en.wikipedia.org/wiki/CAP_theorem.

[8] The Underlying Technology of Messages: https://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919.

[9] MySQL at Facebook: http://www.facebook.com/MySQLatFacebook.

[10] Note on creating Cassandra: https://www.facebook.com/note.php?note_id=24413138919.

[11] Jeremy Zawodny, "Lessons Learned from Migrating 2+ Billion Documents at Craigslist": http://www.10gen.com/presentation/mongosf2011/craigslist.

Other Uses for Secure DNS

PAUL VIXIE

Paul Vixie took over BIND maintainance after Berkeley gave it up in 1989, rewrote it, and then hired other people to rewrite it again. He has recently hired a new team to rewrite it again. Paul is Chairman and Chief Scientist of the Internet Systems Consortium.
vixie@isc.org

Since the mid-1990s I've been a member of a distributed multi-generational team working to secure the Domain Name System. Many ideas and people have come and gone in the decade and a half since this work began, and the current work in progress now being deployed represents a dozen kinds of compromises and band-aids. Yet Secure DNS is being deployed at last, and a market for products and services in this technology is starting to appear. I think this is a good moment to try to remember why securing the DNS seemed like a good idea and to start thinking about other ways to leverage this fundamental change in Internet architecture.

DNS Itself

The original cost justification for deploying DNS itself was that the old SRI-NIC:HOSTS.TXT file was growing (hundreds of kilobytes) and updates to this file were taking a long time (several days) to propagate through the whole Internet. This file just mapped host names ↔ host addresses so that it would be possible to enter or view a host's name even though the underlying Internet architecture worked in terms of binary addresses. To put all this in context, BSD UNIX systems in the 1980s used to pull SRI-NIC:HOSTS.TXT once a week with a cron job, run the "htable" conversion utility to put it into /ETC/HOSTS format, and append a set of local host names showing local host name ↔ address assignments which weren't considered important enough to be worth sending to SRI-NIC.

When Paul Mockapetris designed DNS he gave it a much broader feature set than host name ↔ address translation. For example, DNS made the MX (mail exchanger) record possible, meaning that we could begin to send email to domains rather than to hosts and to have a domain's incoming mail services be provided by more than one host. These were exciting times since this kind of distributed autonomous reliable hierarchical database had never been done before except as proprietary single-vendor standards. Let the record show, however, that the motivation to deploy DNS was not this broader feature set but only the simple expedient of getting rid of the SRI-NIC:HOSTS.TXT file. In other words, the reason DNS was created is broader than the reason DNS was first deployed, and we only have DNS at all today because there was a reason to deploy it in the first place.

Secure DNS

While each member of the distributed and multi-generational team that developed Secure DNS can speak for him or herself as to their individual motives for participating in the effort, I believe that most of us wanted Secure DNS because it would enable a whole new class of distributed applications that could offer enhanced behavior in the presence of crypto-authentic DNS data. We learned early on that the BSD ruserok() function and its ".rhosts" file was a terrible idea, since potential attackers were in direct control of the results of the address → name mapping—the session initiator controlled the IN-ADDR.ARPA data for their own TCP/IP source address. In first-generation DNS, all data is potentially under the indirect control of an attacker since anybody can spoof a DNS response in transit. Since an application that depended on DNS for any of its access or control plane data could be no more secure than DNS itself, no applications were allowed to depend on DNS for sensitive data; as a result, there was no sensitive data in DNS. This made early DNS a distributed hierarchical autonomous reliable database full of non-sensitive data—clearly not as useful as it could be.

But whereas the goal for many of us for Secure DNS was to enable a new class of distributed applications that would be able to depend on crypto-authentic DNS data, there were then no such applications nor any way to build them. Therefore, a short-term expedient was needed, something that would cost-justify the design and deployment of Secure DNS. Most of us realized that the short-term goal had to be to secure the DNS infrastructure itself against medium-value threats such as Web or email redirection. Even though attacks of this kind have never really been common we saw some value in ruling them out altogether. This was for a long time considered too weak a justification for the great and global expense of deploying Secure DNS, but in 2008 Dan Kaminsky showed that spoofing DNS responses was far easier than anybody had thought. After what we called the "2008 Summer of Fear," deployment of Secure DNS finally picked up steam. Note, though, that the justification was still just securing the existing DNS and its existing suite of distributed applications. We knew we couldn't sell Secure DNS based on the vaporous sounding promise of "new applications."

SSHFP

The award for "first DNSSEC-enabled application" goes to Secure Shell (SSH) for which a new record type (SSHFP) for host key fingerprints was created. Secure Shell remembers the host key for every server you've talked to in order to prevent server replacement attacks whereby someone steals a server's network traffic. Under normal conditions, when you talk to a new server Secure Shell will prompt you to verify that server's host key fingerprint just to make sure that the server is what you think it is. Many Secure Shell users do not pay much attention to this prompt and just enter "yes" or click OK or similar without ever reading or verifying the moderately long string of hexadecimal. This creates a security problem whereby users have higher trust in a Secure Shell session than they have any rational justification for, and a server traffic thief can do quite well.

Recent Secure Shell versions now look for an SSHFP record in Secure DNS corresponding to the server's host name. If the result is crypto-authentic in Secure DNS and matches the server's offered key, then Secure Shell need not prompt its user to verify this fingerprint. This may seem like a small thing, especially if it had to carry the full cost of designing and deploying Secure DNS, but it is an example of the kind of things that are possible when we can trust the data we get back from DNS. The full cost of Secure DNS need not be justified by any single new application or new feature, and this fingerprint click-through was a legitimate security concern that could only have been fixed by utilizing a secure global public key infrastructure such as Secure DNS.

X.509 and TLS

Transport Layer Security (TLS) is a way to encrypt TCP/IP session data and possibly also verify the identity of the host or user at the other end of a TCP/IP session. Some sessions start out encrypted (as in HTTPS and IMAPS), in which case it's called Secure Sockets Layer (SSL). Other protocols can switch from clear text to encrypted in a negotiated manner, in which case it's called TLS. As usual in such systems, each side has a persistent host or user key pair (called a "certificate") whose public half is sent to the other side during crypto-negotiation so that the private half can be used for generating secure session keys or signatures. The format of the keying information transmitted during TLS negotiation or SSL startup is called "X.509" and it contains, among other things, a signature on the certificate itself by some outside authority. This signature is used to validate the certificate as belonging to the given host or user. And that's where it all goes off the rails.

If you buy a certificate from an authority known to the other parties to whom you wish to speak securely, they can verify the "certificate authority signature" on your certificate and thus decide to trust the certificate—your certificate—that you're presenting to them. The problem is that the "other end" is usually a Web browser and the maker of that Web browser doesn't necessarily know which certificate authorities they should trust, so pretty much (with a few exceptions) everybody just trusts everybody. Noting the low utility of many certificate authorities, quite a few Web and mail server operators decide to just use a "self-signed certificate," where no certificate authority is involved at all. This results in browser popup messages warning of self-signed certificates which browser operators (that is, end users) usually just click through and ignore. To round things out, some recent incidents have shown lax security or lax verification by certificate authorities such that a lot of certificates out there probably should not have been issued but will nonetheless be universally trusted.

The IETF DANE working group has taken on the task of defining a Secure DNS schema for certificate verification. This will be similar to the Secure Shell SSHFP record, where the operator of the Web or mail server generates a certificate and puts the fingerprint of this certificate into Secure DNS, from where it can be fetched and crypto-authenticated by the other end during TLS negotiation or SSL startup. Some important questions remain, such as whether this will someday enable universal self-signed certificates or whether there will always be a market for "certificate authority" services. What's absolutely certain is that there is value in this approach and that Secure DNS—as the first hierarchical autonomous reliable distributed public key infrastructure—is what's going to make it possible.

User Certificates

The Internet has made connectivity almost universal, but there is nothing like a universal identity system. I don't mean in an Orwellian "big brother" sense, don't worry, I don't want that either. I'm simply noting that passwords don't work well at scale—between one set of people forgetting them and resetting them and another set of people guessing and leaking them, we know that a system with hundreds of millions of passwords is inherently not secure and cannot be made secure. In addition, most of us possess dozens of passwords for different online resources and we either write them down or make them easy to remember or use the same password everywhere or never change them or perhaps all of the above. I cannot imagine a more fruitful electronic crime environment than one in which a billion people do their online buying and selling and banking using passwords.

Happily, Secure DNS will make it possible for any user to create a crypto-authentic anchor for their online identity which could then be the basis for a unified, open, and secure identity system that might in some cases (or on some days or in some places) use passwords, or fingerprint readers, or signature scanners, or near field communications readers, or PIN codes, or challenge and response systems, or whatever those crazy kids in the future will think of. We can't build a system like that without a hierarchical autonomous reliable distributed public key infrastructure. Fortunately, with Secure DNS we now have one of those. In this article, I'm not describing the specifics of what a unified open and secure identity system might look like, merely noting that the first task for the designers of such a system would be to design and deploy something very much like Secure DNS to anchor it all—unless Secure DNS already exists, in which case they can leverage it.

Your Idea Here

DNS in both its original and its new secure form is like a large whiteboard waiting for someone to walk by with a compelling idea. I've told you mine, but I'm actually much more interested in hearing what the rest of the distributed systems community (that is, Internet application developers and creative investors) can think of. Before the Internet the world did not have, and no one really imagined the impact of, universal reachability. Now look. Before Secure DNS the world did not have, and I think no one really imagines the impact of, universal public key infrastructure. Let's find out.

References

RFC 952: "DoD Internet Host Table Specification."

RFC 4255: "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints."

draft-ietf-dane-protocol-10: "Using Secure DNS to Associate Certificates with Domain Names for TLS."

BIND 9.7: "DNSSEC for Humans," http://www.isc.org/.

Hypervisors and Virtual Machines

Implementation Insights on the x86 Architecture

DON REVELLE

Don is a performance engineer and Linux systems/kernel programmer, specializing in high-volume UNIX, Web, virtualization, and TCP/IP networking performance. He has 15 years of UNIX systems engineering and programming experience with a few large high-profile companies in the finance and Internet news media sectors.
don@js-objects.com

Hypervisor and virtualization technology is used to drive cloud computing, server consolidation, clustering, and high availability solutions. The x86 processor line is now the dominant platform for virtualization. Although the x86 processor has a few virtualization challenges, many solutions have been architected. This article's purpose is to briefly explore the principles and operational concepts behind these solutions.

VMware, Microsoft Hyper-V, XEN, KVM, and other Linux-based hypervisors are among the growing number of popular implementations. Most hypervisors use optimized forms of binary translation, paravirtualization, or CPU-supported solutions for full virtualization implementation.

The Challenges of Virtualization and the Popek and Goldberg Principles

Some of the main challenges of x86 virtualization are efficient OS isolation and virtualization overhead. Operating systems expect to execute with unchallenged privilege in order to support their own internal services and the services they offer to user processes. In order to share the x86 CPU with multiple operating systems, an intermediary protocol and a privilege mechanism will be needed. Without an intermediary, CPU partitioning arrangement, or specialized operating system, any attempt to run multiple operating systems would simply bring down the entire system.

Other significant problems of virtualization include execution speed, security, memory management, multiplexing, and isolation of devices such as network cards.

The Popek and Goldberg principles of virtualization [1, 2] define a set of specifications for efficient processor virtualization. The standard suggests the ideal models and expectations of privileged instructions. Ideal instructions should behave in expected ways no matter what the current operating privilege level is and should trap any problems. Not all of the x86 processor instruction set meets the Popek and Goldberg principles, so an intermediary must be used to resolve the issues regarding the problematic small subset of the entire instruction set of the x86 architecture.

The Hypervisor/VMM Abstraction as an Intermediary

Hypervisors supervise and multiplex multiple operating systems by using highly efficient and sophisticated algorithms. The hypervisor is a well-isolated, additional but minimal software layer. The hypervisor must work with minimal overhead and maintain supervisory privileges over the entire machine at all times. The hypervisor seeks to define and enforce strong separation policies and finite boundaries for which operating systems can operate cooperatively.

The x86 processor line uses privilege levels known as rings, and a stand-alone hypervisor takes advantage of this fact. By obtaining privilege ring 0, the highest privilege level, the hypervisor can supervise and delegate all of the system resources. Of course the other operating systems will have to be kicked down to lesser privilege levels if the CPU does not support virtualization internally. X86 processor privilege protections provide the means to completely isolate the hypervisor. Often the hypervisor is referred to as a virtual-machine monitor (VMM), and these terms are used in this article interchangeably.

The x86 processor is dependent on lookup tables for runtime management. Global tables such as the interrupt descriptor vector and the memory segment descriptors are examples of such structures. Controlling access to these tables is a must for any VMM. In a multiplexed OS environment, these and other processor control tables are maintained by the hypervisor only and therefore must be emulated for each virtual context. In particular, it is the hypervisor's role to manage all processor control tables and other processor facilities that cannot be shared in a multiplexed OS environment.

Overview of Virtualization Mechanics

Emulation with Bochs

Bochs [9] is a software emulation of a CPU and the various PC chipset components; it implements the entire processor instruction set and emulates a fetch, decode, and execution cycle the way a physical CPU does. Bochs executes all instructions internally by calling its internal functions to mimic the real ones, which never hit the CPU. The Bochs emulation engine is implemented as a user-space application, and it uses its allocatable memory address space to model a physical memory address space.

This type of translation loses the battle when it comes to execution speed. The additional overhead of nested memory access, opcode parsing, and execution of emulated instructions using procedures in memory results in multiple real processor instructions for each of the emulated instructions. Bochs is an example of a simpler base case of virtualization.

Transition Algorithms and Direct Execution

Translators such as the versions used by VMware's ESX Server [10] and QEMU [11, 12] use intelligent forms of translation. Translators of this type have a huge performance advantage over a simpler interpretive type of emulation such as Bochs. The basic idea is to translate the source of instructions into a cache of instructions that can directly execute on the processor.

Forms of binary translation are intermediary algorithms that are used for processors that do not have virtualization support. As stated earlier, the x86 does not meet the standards provided by Popek and Goldberg. There are a few processor instructions that do not behave in a manner suitable for virtualization. The translation process scrubs and replaces problematic instructions with alternate instructions that will emulate the original.

An overly simplified example follows. While in the fetch and decode phase, the translator comes across the cli instruction. cli is a privileged instruction that disables interrupts on a x86 CPU. The translator can instead replace it with instructions that disable interrupts only on the abstraction that represents the virtual CPU for the virtual machine, not the real CPU. Again, this is a basic conceptual example to help get the idea of translation across.

The instructions that have been translated are cached into blocks that are used for direct execution on a CPU, and at the end of each translated block are instructions that lead back into the hypervisor. Any CPU exception or error that occurs while the translated stream is executing forces the CPU back to the hypervisor. This is critical for security and isolation; the virtualized operating system has no chance of getting control of the CPU, unless the hypervisor itself has been compromised.

Note that VMware has developed a very sophisticated version of binary translation. The algorithm is called Adaptive Binary Translation [3]. In addition, it is a VMM that has features that facilitate mass rollout and management of machines such as virtual-machine migration and memory management. QEMU is itself a user-space program, while ESX is implemented as a kernel. The advantage is that ESX is a hypervisor in the more strict definition which gives it full operational range over the processors.

Para-virtualization with XEN

Para-virtualization under XEN [4, 8] provides a software service interface for replacing privileged CPU operations. Operating systems must be specifically modified to be aware of the services that the XEN hypervisor provides.

To make an OS XEN-aware, the developer has to modify highly complex kernel procedures and replace privileged instructions in the source code with calls to the XEN interface which will emulate the operation in its isolated address space. After the OS is recompiled against the XEN Interface, a directly executable operating system is created. After some administrative setup, XEN can load and schedule this OS for direct processor execution.

The communications gateway that the para-virtualized OS uses to request XEN operations is based on interrupts, recast as hypercalls in XEN terminology. Hypercalls tie the operating system to the hypervisor. When a hypercall is issued, the CPU transfers control to a hypervisor procedure which completes the request in privileged mode. This is the same as the system call interface that system programmers are used to, except the requests are from the kernel to hypervisor. The XEN privileged operations exist in an address space only accessible by XEN, and this addressing method mimics the kernel/process address space split in standard x86_32 processors.

Hypercalls are used for registering guest local trap tables, making memory requests, guest page table manipulation, and other virtual requests. Kernel subsystems such as the slab cache object allocation and threading are not virtualized, but devices and page tables are virtualized. A full list of hypercalls can be viewed in xen.h in the source tree.

XEN, which is a stand-alone bare-bones kernel, maintains ultimate control over the processor as it is the supervisor of the system and sits isolated in ring 0, and the para-virtualized guest OS executes with reduced privileges. While any guest OS is executing on a processor, any processor exceptions or errors are trapped and handled by the hypervisor, thus providing strong isolation and security. For efficiency, XEN allows Linux to directly handle its system calls (0x80h) while it is on the CPU, thus bypassing the XEN layer. This is known as a fast-trap handler.

A XEN para-virtualized guest operating system has a startup and boot-strapping procedure that is different from the stand-alone OS. There is no BIOS available to query for things such as the installed memory size. XEN provides a special table that is mapped into the guest OS address space as a replacement. This is a C structure which is called the "start_info" page. The xen.h include file shows the specifics of the contents of this structure.

It is generally accepted that para-virtualization provides very good performance. The major downside is that the operating system source code must be modified by the maker or a third party. This presents a problem for systems, such as the Microsoft OS product line, which are not open source. Popular open sourced operating systems such as Linux and some versions of the BSD kernels have been successfully para-virtualized to run under XEN without CPU-supported virtualization. Note that XEN does support CPUs with full embedded virtualization.

Full Virtualization with CPU Supported Extensions

AMD, Intel, and others have now embedded virtualization properties directly in the processor. The advantage of this is that any x86-based operating system can execute directly in a virtualized machine context without any binary translation or source code modification. AMD calls its virtualization implementation AMD-V, and Intel calls its implementation Virtualization Technology (VT).

The AMD and Intel virtualization chipsets support the concept of a guest operating system and a new additional privilege mode exclusively for hypervisor use. The hypervisor executes with full authority over the entire machine, while the guest operates fully within its virtual-machine environment without any modification. From the guest OS point of view, there are no new or different privilege levels, and the guest OS is not aware that it is itself a guest. The usual 0/3 ring setup is maintained for the kernel and user processes.

Both AMD and Intel use the key idea of a VMM management table as a data structure for virtual-machine definitions, state, and runtime tracking. This data structure stores guest virtual machine configuration specifics such as machine control bits and processor register settings. Specifically, these tables are known as VMCB (AMD [5]) and VMCS (Intel [6]). These are somewhat large data structures but are well worth reviewing for educational purposes. These structures reveal the many complexities and details of how a CPU sees a virtual machine and shows other details that the VMM needs to manage guests.

The VMCB/VMCS management data structures are also used by the hypervisor to define events to monitor while a guest is executing. "Events" are processor-specific conditions that can be intercepted, such as attempts to access processor control registers and tables. Note that the VMCB/VMCS data structures are only mapped into hypervisor-accessible pages of memory. The guest OS cannot be allowed to access these tables, as this would violate isolation constraints.

The triggering of a monitored event is called a VM-EXIT (virtual-machine exit). When this happens the CPU saves the current executing state in the active VMCB or VMCS and transitions back into the hypervisor context. Here the hypervisor can monitor and fix the issue that caused the exit from the guest. Each specific processor uses model-specific registers to store the location of VMCB/VMCS tables.

Any errors, unexpected problems, or conditions that may require emulation that occur while the guest is executing force a VM-EXIT and switch to the hypervisor host context. This is somewhat similar to the exception/trap transition sequence used by x86 processors running standard operating systems.

Note that XEN, VMware, KVM, and Hyper-V support CPU-extended virtualization.

KVM under Linux

KVM is a very popular Linux-based hypervisor with full virtualization. KVM is a kernel module that brings virtualization directly into the host Linux kernel. The KVM is not completely stand-alone, as it uses the Linux host kernel subsystems. KVM is implemented using CPUs with full virtualization support, along with a QEMU software back-end. QEMU provides device emulation for guest operating systems under KVM control. I/O requests made to virtual devices are intercepted by KVM and queued up to a QEMU instance which is then scheduled for processor time in order to complete the request.

One of the differences with KVM is that it uses the host Linux kernel subsystems. Guest virtual machines are Linux tasks that execute in processor guest mode. KVM supports most operating systems by utilizing CPUs with virtualization support. See the KVM documentation [7] for the supported list of operating systems.

IOMMU (I/O Memory Management Unit)

IOMMU chipsets are becoming mainstream on x86_64 architectures. An IOMMU allows a hypervisor to manage and regulate DMA (direct memory access) from devices. In a stand-alone OS environment, a device can access any system memory address that it has address lines for. The operating system and its devices see a single system memory address space.

But in a multi-OS virtual machine environment, multiple system address spaces are defined by the hypervisor for guest use. Devices, however, still only see a single system address space. The IOMMU creates a virtual system address space for devices and effectively correlates, translates, and sets finite bounds on a device's range of addressable memory.

The IOMMU is positioned in the hierarchy of system buses where it can identify source devices and intercept their memory requests and use its internal IOMMU page tables to allow/deny and translate the requested memory address. The power of the IOMMU allows the hypervisor to assign devices to specific guest operating systems and restrict the devices' memory access to pages in the address space of the guest. This IOMMU isolation and mapping feature is used for PCI-Passthrough.

PCI-Passthrough permits a guest operating system to access a device natively. The guest OS is not aware that it is being redirected by an IOMMU and does not have the access or ability to make modifications to the IOMMU chip. Doing so would open up a huge security hole: for example, OS#2 could program its assigned device to write to a specific page of memory owned by OS#3 or overwrite a page owned by the hypervisor.

An IOMMU is somewhat analogous to the MMU used in x86 CPUs for virtual memory translation. Both IOMMU and MMU use page tables to hold address translation metadata. And, like the x86 CPU MMU, the IOMMU generates exceptions and faults which the hypervisor must catch and resolve.

Intel brands its IOMMU chipset Virtualization Technology for Directed I/O (VT-d), and AMD brands its chipset as I/O Memory Management Unit (IOMMU).

Conclusion

Virtualization innovations are accelerating and overcoming previous efficiency limitations. A good understanding of the internal structure is increasingly vital and will assist in understanding the implications of issues such as performance, scaling, security, and the proper architecting needs of the virtualization layer in your infrastructure.

At this point, full virtualization with CPU support will likely be the main focus of solutions going forward. It provides more flexibility and leverage for hypervisor implementors and more choices for the end user. VMware, XEN, Hyper-V, and KVM support CPUs that have embedded virtualization capabilities.

References

[1] Gerald J. Popek and Robert P. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures," *Communications of the ACM*, vol. 17, no. 7: 412–421.

[2] en.wikipedia.org/wiki/Popek_and_Goldberg_virtualization_requirements.

[3] Keith Adams and Ole Ageson, "A Comparison of Software and Hardware Techniques for x86 Virtualization," 2006: http://www.vmware.com/pdf/asplos235_adams.pdf.

[4] David Chisnall, *The Definitive Guide to the Xen Hypervisor* (Prentice-Hall, 2007).

[5] AMD Developer Manuals: http://developer.amd.com/documentation/guides/Pages/default.aspx.

[6] Intel Processor Developer Manuals: http://www.intel.com/products/processor/manuals/.

[7] KVM: http://www.linux-kvm.org/page/Main_Page.

[8] XEN: http://www.xen.org/.

[9] Bochs: http://bochs.sourceforge.net/.

[10] http://www.vmware.com/virtualization/.

[11] http://wiki.qemu.org/Main_Page.

[12] Fabrice Bellard, "QEMU, a Fast and Portable Dynamic Translator," 2005 USENIX Annual Technical Conference: http://www.usenix.org/events/usenix05/tech/freenix/full_papers/bellard/bellard_html/.

A RESTful Web Service for Internet Name and Address Directory Services

ANDREW NEWTON, DAVE PISCITELLO, BENEDETTO FIORELLI, AND STEVE SHENG

Andy Newton is the Chief Engineer at the American Registry for Internet Numbers, where he oversees system and data architecture and leads multiple teams of software developers.
andy@arin.net

Dave Piscitello is a Senior Security Technologist for ICANN. A 35-year Internet veteran, Dave currently serves on ICANN's Security and Stability Advisory Committee and the Internet Policy Committee of the Anti-Phishing Working Group (APWG).
dave.piscitello@icann.org

Benedetto Fiorelli is a Senior Software Engineer at RIPE NCC. He has contributed to vision, design, and programming of production-grade software for different types of organizations across very different domains. His main areas of expertise are software design, service-oriented architecture, data modeling, and quality software management.
bfiorell@ripe.net

Steve Sheng joined ICANN in 2009 as a Senior Technical Analyst. In this position, he supports projects of ICANN's Security and Stability Advisory Committee (SSAC) and provides research support for projects in the policy department.
steve.sheng@icann.org

Domain name and Internet Protocol address registry operators collect point of contact and other information associated with the delegated administration and use of domain names or IP addresses. Registries and other operators make this information available via the Whois protocol (RFC 3912 [1]) and Web-based interfaces. In this article, we describe how directories based on representational state transfer (REST [2]) Web services could support features that have been identified as desirable or beneficial for domain name and IP address registration data.

By registration data, we refer to the information that registrants provide in order to obtain the right to use a domain name or to have an IP address space allocated for use. For domains in the generic top-level domain space, these data elements are specified in the Registrar Accreditation Agreement and individual registry agreements with ICANN. For IP allocations, each Regional Internet Registry specifies its own set of data elements. In this article, we provide summaries of prototype and early production deployments of RESTful Web services, the current status of standardization efforts, and future plans for this work.

Background

Created in the 1980s, NICNAME/WHOIS began as a service used by Internet operators to identify network administrators. Today, we access registration data via Whois services to identify and contact the registered users of Internet (IP) address allocations and domain names for business matters and on matters related to trademark protection, criminal activities (phishing, spam, botnets), to verify online merchants, and more.

Whois services have evolved in a largely ad hoc manner for over two decades, and not without considerable scrutiny and criticism. Many of the deficiencies and desired additional features are mentioned in reports from ICANN's Security and Stability Advisory Committee [3, 4, 5, 6], in reports of ICANN supporting organizations [7], and by members of the Regional Internet Registry community [8, 9]. Among these are:

Need to support internationalized registration data. The Whois protocol has no standard mechanism for clients and servers to signal a character set. Monolingual users whose language cannot be represented using characters from the US-ASCII7 character set are greatly disadvantaged by this limitation. Unicode and multilingual support is widely available from Web applications. This internationalization is becoming increasingly necessary for registration data. The proliferation of non-standard signaling conventions by registry operators does not scale.

Need to standardize and enhance service. The Whois protocol describes exchanges of queries and responses between a client and a server over a specific TCP port (43). The only constraint the specification imposes on query and message formats is that they must be terminated with ASCII line feed and carriage return characters. The specification does not define standard formats or encodings. It does not have a schema for replies or error messages. The resulting variability across client and server implementations detracts from the quality and usability of Whois "port 43" client as well as Web-based applications. In particular, the lack of uniformity inhibits or adds complexity to machine parsing (automation).

Need for security services. Users or applications access Whois services anonymously, requiring no identity assertion, credentialing, or authentication. Few methods are used to restrict access to Whois servers other than rate limiting based on IP address-level access controls. The lack of authentication mechanisms inhibits adoption of effective user- or group-level access controls, auditing, or privacy measures, features that are typical for directory services [10].

Of these, we believe that developing support for internationalized registration data and seeking uniformity of service are matters that require urgent attention. The security services mentioned, while important, are largely absent from Whois services today, and their inclusion is a matter for policy development.

We posit that the community should consider universal adoption of Unicode and should adopt markup languages and widely accepted data interchange formats. Modern applications benefit from adoption of these instead of free-form text. We further posit that Whois users would benefit from efforts to standardize on signaling and error messages. We recognize that some Whois users view security services as desirable and beneficial, and believe that any framework and protocol considered for a registration data directory service must be sufficiently extensible to support security services should they become requirements.

Why RESTful?

In choosing a RESTful approach, we began by considering first principles for protocols. All protocols have a control part, which defines message formats, security features, signaling, encodings, and errors, and a data part, which is the information that users see and applications process. The control part of the existing Whois protocol specifies little about the data part and would require a virtual rewrite to satisfy the needs we identified earlier. Development of the control part would be encumbered by the need to provide backwards compatibility for the numerous ad hoc extensions various providers of Whois services have adopted in the absence of a formally specified control part in RFC 3912. Today, non-Latin characters are supported differently by several domain name registries. In the extreme, users or applications must know these for each registry they query.

IRIS [11], envisioned by some as a successor to Whois services, has a control part that is specific to the data part (i.e., registry-specific). IRIS also requires its own application-specific transport to operate correctly over TCP or UDP. Whereas a Whois service has very little control part, experience with IRIS has proved its control part is complex and requires sufficient investments in application development for both clients and servers that it was an impractical choice for a widely offered (and, in some circumstances, mandatory) service that typically does not generate revenue.

By juxtaposing Whois against IRIS, we were able to derive a list of desirable criteria for a registration data directory service. The control part should be sufficiently flexible to satisfy domain name and Internet address registry needs. It should support structured, typed data and international encodings. If possible, it should leverage or integrate well with existing (Web) application infrastructures. It should readily accommodate the inclusion of security as mandatory or optional features.

The American Registry for Internet Numbers (ARIN) had been experimenting with Representational State Transfer-based Web services, and their results encouraged the authors and their colleagues to experiment as well. Prototypes developed by ICANN and RIPE further reinforce our belief that a RESTful approach might satisfy the needs we have identified.

Benefits of Representational State Transfer-based Web Services

Representational State Transfer-based Web services define a pattern of usage with HTTP to create, read, update, and delete (CRUD) resources. Resources are addressable as Universal Resource Locators (URLs). The RESTful framework leverages the HyperText Transfer Protocol (HTTP) infrastructure, including caching, referrals, authentication, version control, and secure transport (HTTPS). The programming API accommodates Unicode and numerous markup languages, supports signaling and standard error messages, and runs on top of standard Internet transport protocols.

A RESTful approach allows us to identify structured data types and incorporate these in URL patterns to refer unambiguously to individual resource objects. The existing Whois Web interfaces demonstrate our ability to support a wide client base, including ordinary Web browsers; command-line utilities like curl, wget, and xmllint; and embedded client implementations such as libcurl and various libraries for Perl, PHP, and Java.

RESTful Web services implemented on platforms that support SSL/TLS [12] for other purposes (such as registration services) may be able to leverage this implementation to provide authenticity of origin, transport confidentiality, (sub) authentication, and other security services that have already been implemented around the HTTPS framework.

Benefits of Structured and Typed Registration Data

Several benefits can be derived from establishing data conventions or standards for Web-based queries and responses to registration databases or repositories. First, they allow the signaling or delivery of metadata about the registration data, such as the language of the registration data. Second, they facilitate search mechanisms.

Some consumers of domain name registration data want to be able to search registration data using attributes or object types in much the same way that they can currently perform such searches using, for example, the APNIC Whois service [13]. Today, submission forms typically only accommodate rudimentary query arguments such as <domain name> or <IP address>. Certain communities can benefit from queries for such information as sponsoring registrar or name server information for domain name registrations, or autonomous system numbers, networks, or reverse DNS delegations for Internet address registrations.

Structured, typed data facilitates such queries. Such searches are performed now, often by legal obligation, through ad hoc requests. Providing a standard mechanism for performing such searches could lower the costs to registries and searchers in undertaking this duty. Having a standard, machine-parsable format is especially valuable for performing large-scale data analysis across the registration databases. With appropriate controls to balance privacy interests, such a facility would provide a means of greatly enhancing our understanding of some types of activity on the network.

Experience with RESTful Registration Data Directory Services

Using the prototypes and early production experiences we describe in this section, we demonstrate that adopting a RESTful framework for registration data directory services offers ease of implementation and ease and rapid integration with existing deployments, and, further, that registration data defined using XML offers the opportunity to create flexible submission queries and easily parsable responses using a "universally understood" meta-language that accommodates internationalized registration data requirements for domain name and Internet address registries.

ARIN Whois-RWS

In 2009, ARIN completely rewrote its traditional Whois service as part of a major software re-engineering effort, enhancing this service with a RESTful Web interface, Whois-RWS.

Our Whois-RWS mirrors the major structured data types used by ARIN in URL patterns:

`/rest/poc/XXXX` for points of contact
`/rest/net/XXXX` for networks (IP prefixes)
`/rest/org/XXXX` for organizations
`/rest/asn/XXXX` for autonomous system numbers
`/rest/rdns/XXXX` for reverse DNS delegations

The relationship between these structured types was easy to express using HTTP URLs.

Searches make use of HTTP URL matrix query parameters. Such queries do not utilize a specific identifier to narrow the result set to a single intended item. Multiple query parameters allow the querying party to specify multiple, distinct search inputs. For example, "/rest/pocs;first=John&last=Doe" is a request for points of contacts where the contact's given name is "John" and surname is "Doe".

While XML is the primary output format, the user can direct Whois-RWS to render results in XHTML [14], JSON [15], or plain text by using the HTTP Accept header or by appending a file extension type to the query URL (e.g., .xml, .txt, etc.). The HTTP Accept header is a standards-based method for requesting a specific MIME type. This allows Web browsers to request XHTML or XML styled with CSS automatically so that end users can directly view data in Whois-RWS. When rendered in XML styled with CSS, the output is both easily machine parsable and user friendly.

The ARIN team uses the Relax NG formal schema language [16] to define the XML so that programmers can easily determine what to expect in the response. Extension points are clearly defined in the schemas, which can thus be extended while preserving backwards compatibility. ARIN has already utilized this extensibility when it rolled out new reverse DNS delegation infrastructure to better support DNSSEC. We were able to create Whois-RWS quickly and easily by reusing large software components hitherto developed for ARIN's Web portal system, ARIN Online. The reuse of standard Web software components gave us time to add additional features, such as CIDR query support and a new near-real-time data replication system. Initial prototypes of Whois-RWS were greenlighted in the summer of 2009, and Whois-RWS was released as a production system in July 2010. By this time, customers were using Whois-RWS to augment their own IP address management systems, and one Flash-based application had been written to take advantage of the machine-readable information. As of March 2011, over 40% of Whois queries served come through the RESTful Web interface.

ICANN

The ICANN team began to experiment with a RESTful service for domain name registration data directory service in 2010. Our goals for this pilot were to understand whether a RESTful Web service could support requirements for the submission and display of internationalized domain names and registration data, as the team understood the requirements at that time [17]. Having performed several studies that involved machine parsing and normalization of tens of thousands of registration records collected from a random and large set of registrars [18, 19], we also sought to improve usability of domain name registration data for such purposes by defining standard, extensible formats for the data that would also accommodate future changes. Lastly, we sought to understand the design tradeoffs associated with a RESTful service versus an enhanced Whois for domain registration data, and the approximate cost and complexity of implementation.

Similar to ARIN's implementation, ICANN's prototype service uses the HTTP protocol and conforms to the REST architecture. The client sends its request with the following URL structure:

`/rest/domain/XXXX` for domain name request
`/rest/contact/XXXX` for contact request (by contact ID only)
`/rest/host/XXXX` for host request

The client signals the preferred format using the standard HTTP Accept header. The client can also signal the preferred format by adding a DOS-file-style extension to the resource. The server provides responses in XML, HTML, and plain text format.

The ICANN prototype uses a formal XML schema language so that programmers can easily determine what to expect in the response. The data schema largely reuses the data schema defined in the Extension Provisioning Protocol (EPP) [20, 21, 22, 23]. The ICANN team chose the EPP schema to leverage the existing standards associated with registry-registrar information transfer and to minimize reinvention: ICANN accredited registries and registrars use EPP for their operations, so they are familiar with the schema and may be able to reuse existing software.

The prototype demonstrates that the REST architecture with EPP supports internationalized domain names and registration data in the following way. Queries can be expressed using Internationalized Resource Identifiers (IRIs) [24] and thus accommodate non-ASCII input. EPP data schema accommodate internationalized contact name, organization, and address information represented in unrestricted UTF-8 using the attribute (type=”loc”) from RFC 5733.

The ICANN team continues to experiment with the RESTful service to develop and propose standardized error handling and to find better ways to signal encodings other than UTF-8. We will investigate whether the code base is suitable for further work as an open source project. While we currently use “canned data,” we are considering hosting an experimental service for the IANA registries or certain TLDs that ICANN manages (such as the .INT and .ARPA).

RIPE Database REST API

A large number of organizations and individuals use the RIPE Database. Historical and technical aspects of the Whois protocol, and in particular the Routing Policy Specification Language (RPSL [25]), influence how these clients interact with the RIPE Database.

RPSL specifies much more than routing policies, describing, in practice, more than 20 different types of RPSL objects. It is more a formalization of the way policy records have been stored and exchanged in the existing Whois systems since the late 1980s than a high-level domain language specification. As a result, the policy specification language and its extensions are tightly coupled to the way policies are stored in the existing Whois systems and vice versa; thus any new system that implements the language essentially reproduces a Whois service.

These processes are modeled on a human-centered workflow. They are not optimized for building new services, extending existing ones, or building tools on top of them. Thus, client applications can quickly become overly complex when dealing with RPSL. Many become too expensive to be extended or maintained. The RIPE Database Group observed needs for simpler interfaces and machine-parsable data formats that would simplify the development of services and tools and increase the value of registries by exposing new domain-specific interfaces.

The RIPE Database Group developed a layer of Service Provider Interfaces (SPI) and a data schema simple enough to be agnostic of the underlying registry implementation. The SPI allows composition of domain-specific services and also makes possible real-time interoperation between registries. The RIPE team chose REST because HTTP is the most accessible protocol and the HTTP methods, resource locator protocol, HTTPS, and other features provide a flexible and proven framework for stateless services. For the representation schema we have designed a relaxed attribute-oriented XML Schema. We only apply a structural validation via XML Schema Definition [26]. After some testing we decided to remove any form of attribute or type validation in order to reuse the same schema on different RPSL flavors.

The services support JSON, HTML, and plain text, all derived via XSL [27]. HTML and text transformation demonstrate the transformation powers of XML and how resource navigation can be accomplished using any HTML browser. As is the case in the ARIN implementation, content negotiation is done using HTTP headers or by appending a file extension to the request URL.

The query services can be used on any RPSL-based Whois server or mirror. It is possible to execute the same Lookup or Search request on all the Regional Internet

Registries and return all the responses as a unique set of resources. The query services feature:

- Single-resource lookup service: Given a primary key, a type, and a registry, it always returns one and only one object. It can also be used to identify resources by URL bookmark.
- Resolution of referenced resources: Given a resource, all the attribute values that represent references to other resources contain an xlink anchor that can be followed to navigate and browse networks of resources.
- The client can navigate through any network of resources via xlinks without requiring any stateful information to be stored on the servers. This comes as a benefit of the two previous features.
- Normalization of continuation lines, end-of-line comments, and other RPSL intricacies, and normalization of comma-separated values when they represent references to multiple resources.

CRUD interfaces also can be used on any RPSL-based Whois server or mirror by building adapter modules for the different update mechanisms provided by different registries (given that they adopt the same set of resource types). The CRUD Services split the single overloaded generic update interface provided by Mail Updates and Syncupdates into separate low-level interfaces, each defining a simpler contract, designed for programmatic use rather than for human interaction. The new CRUD Services provide a Delete interface that only requires a primary key, a type, a registry identifier, and one or more passwords. It is the equivalent of a lookup but is executed with the HTTP Delete method. This is exposed only on HTTPS, through a request of the form HTTP DELETE: https://lab.db.ripe.net/whois/delete/test/person/pp16-test?password=123. The server responds with an HTTP Status code indicating success or failure. Failure conditions return unique status codes.

We also prototyped some attribute modification services on top of the CRUD methods. With only one request, we can implement complex update workflow. For example, using one HTTP request it will be possible to execute commands such as:

- Replace all the attributes of a given type with a new set of attributes.
- Remove all the attributes that have a value matching the given regular expression.
- Add this set of attributes after the Nth attribute of type X.

Examples of the lookup and search services can be performed using the following URLs:

http://apps.db.ripe.net/whois/lookup/ripe/organisation/ORG-BME1-RIPE.xml
http://apps.db.ripe.net/whois/lookup/ripe/route/193.6.23.0/24/AS2547.xml
http://apps.db.ripe.net/whois/search.xml?flags=r&source=ripe&source=apnic
&source=afrinic&query-string=AS2547

The technical documentation of the RIPE Database REST API can be found at [28].

Findings and Conclusions

The prototyping and early production experiences support our claim that a RESTful approach is a simple yet elegant solution to the problem set we have identified in this paper. We are able to support internationalized registration data (and, generally, structured and typed data), provide unambiguous signaling, and improve error reporting. We are able to leverage existing client and server infrastructures and provide security services, including transport confidentiality and integrity checking, authentication, and data filtering, in an extensible manner, again with the prospect of being able to leverage implementations and Web infrastructure that makes use of security services today.

Future Work

We intend to continue collaborative experimentation and further development of prototypes. ARIN's production Whois-RWS will provide valuable insight into the features most commonly used. Users may identify additional features or may assist in identifying areas for improvement.

We have requested and received approval from the Internet Engineering Task Force (IETF) Applications area director to present this work to the technical community. We have submitted a draft requirements specification to introduce a series of documents that define the overall problem and some available solutions. The series includes Requirements for Internet Registry Services, descriptions of the ARIN, ICANN, and RIPE Internet Registry Service APIs, and a description of RESTful Whois. Our intent is to publish one of the API specifications as a standards track document and the remainder (including this memo) as informational documents. To participate in discussions about work on this next-generation Whois technology at the IETF, join the Whois-based Extensible Internet Registration Data Service (WEIRDS [29]).

Acknowledgments

The authors recognize the following colleagues for their cooperation or contributions to this joint activity: Joe Abley (ICANN), Francisco Arias (ICANN), Kim Davies (ICANN), Mark Kosters (ARIN), Paul Palse (RIPE), Kaveh Ranjbar (RIPE), Andrew Sullivan, Leo Visgoda (ICANN).

References

[1] L. Daigle, Whois Protocol Specification, RFC 3912, 2004: http://www.rfc-editor.org/rfc/rfc3912.txt.

[2] Roy Fielding, "Architectural Styles and the Design of Network-based Software Architectures," PhD dissertation, University of California, Irvine, 2000: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[3] ICANN Security and Stability Advisory Committee (SSAC), *Whois Recommendation of the Security and Stability Advisory Committee* (SSAC publication No. 003), 2003: retrieved from http://www.icann.org/en/committees/security/sac003.pdf.

[4] ICANN Security and Stability Advisory Committee (SSAC), *Is the Whois Service a Source for Email Addresses for Spammers?* (SSAC publication No. 023), 2007: retrieved from http://www.icann.org/en/committees/security/sac023.pdf.

[5] ICANN Security and Stability Advisory Committee (SSAC), *SSAC Comment to GNSO regarding Whois Studies* (SSAC publication No. 027), 2008: retrieved from http://www.icann.org/en/committees/security/sac027.pdf.

[6] ICANN Security and Stability Advisory Committee (SSAC), *Domain Name Registration Information and Directory Services* (SSAC publication No. 033), 2008: retrieved from http://www.icann.org/en/committees/security/sac033.pdf.

[7] ICANN Generic Names Supporting Organization (GNSO), *Inventory of Whois Service Requirement Final Report* (Marina Del Rey, CA: ICANN), 2010: retrieved October 21, 2010 from http://gnso.icann.org/issues/whois/whois-service-requirements-draft-final-report-31may10-en.pdf.

[8] A. Newton, "Replacing the Whois Protocol: IRIS and the IETF's CRISP Working Group," *IEEE Internet Computing*, vol. 10, no. 4, July–Aug. 2006, pp. 79-84.

[9] Cathy Murphy, "Some RIR Input regarding Whois Deficiencies," presented at IETF 53, 2002: retrieved from http://www.ietf.org/proceedings/53/slides/crisp-1/sld001.htm.

[10] ICANN Security and Stability Advisory Committee (SSAC), *Domain Name Registration Records and Directory Services* (SSAC publication No. 33), 2008: retrieved from http://www.icann.org/en/committees/security/sac033.pdf.

[11] A. Newton and M. Sanz, "IRIS: The Internet Registry Information Service Core Protocol," 2005: http://www.rfc-editor.org/rfc/rfc3981.txt.

[12] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol, Version 1.2," 2008: http://www.rfc-editor.org/rfc/rfc5246.txt.

[13] APNIC Whois Search: http://www.apnic.net/apnic-info/whois_search2.

[14] World Wide Web Consortium, Extensible HyperText Markup Language (2nd ed.), 2002: retrieved from http://www.w3.org/TR/xhtml1/.

[15] JavaScript Object Notation (JSON), 2011: retrieved from http://www.json.org/.

[16] Relax NG, 2011: retrieved from http://relaxng.org/.

[17] ICANN Security and Stability Advisory Committee (SSAC), *Display and Usage of Internationalized Registration Data: Support for Characters from Local Languages or Scripts* (SSAC Publication No. 37), 2003: retrieved from http://www.icann.org/en/committees/security/sac037.pdf.

[18] D. Piscitello and S. Sheng, "Abuse of Domain Name Privacy Protection Services," 2010: retrieved from http://securityskeptic.typepad.com/the-security-skeptic/2010/04/domain-name-privacy-misuse-studies.html.

[19] D. Piscitello and S. Sheng, "Abuse of Domain Name Privacy Protection Services: Act Deux," 2010: retrieved from http://securityskeptic.typepad.com/the-security-skeptic/2010/10/misuse-of-domain-privacy-protection-services-act-deux.html.

[20] S. Hollenbeck, RFC 5730, Extensible Provisioning Protocol (EPP), 2009: http://www.rfc-editor.org/rfc/rfc5730.txt.

[21] S. Hollenbeck, RFC 5731, Extensible Provisioning Protocol (EPP) Domain Name Mapping, 2009: http://www.rfc-editor.org/rfc/rfc5731.txt.

[22] S. Hollenbeck, RFC 5732, Extensible Provisioning Protocol (EPP), Host Mapping, 2009: http://www.rfc-editor.org/rfc/rfc5732.txt.

[23] S. Hollenbeck, RFC 5733, Extensible Provisioning Protocol (EPP) Contact Mapping, 2009: http://www.rfc-editor.org/rfc/rfc5733.txt.

[24] M. Duerst and M. Suigard, RFC 3987, Internationalized Resource Identifiers (IRIs), 2005: http://www.rfc-editor.org/rfc/rfc3987.txt.

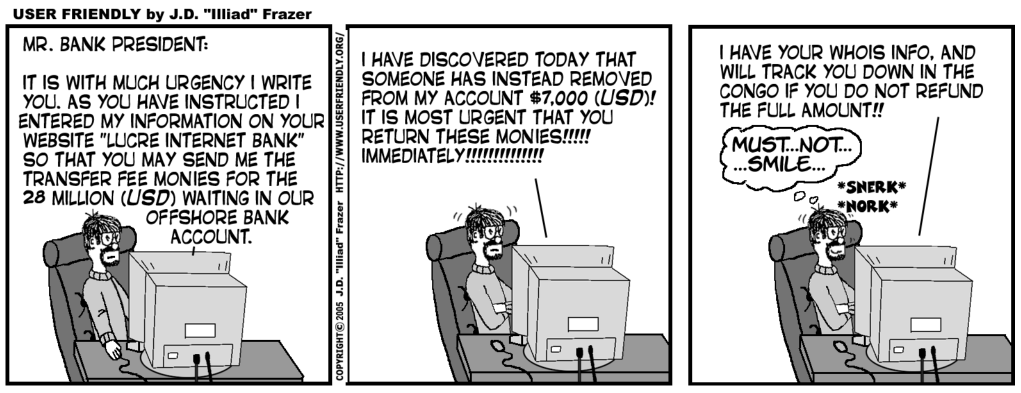
[25] RPSL Reference Guide: http://www.irr.net/docs/rpsl.html.

[26] XML Schema Definition: http://www.w3.org/XML/Schema.

[27] XML Stylesheet Language: http://www.w3.org/Style/XSL/.

[28] Benedetto Fiorelli, RIPE Database REST API: http://labs.ripe.net/ripe-database/database-api/api-documentation.

[29] WEIRDS Info Page: https://www.ietf.org/mailman/listinfo/weirds.


USER FRIENDLY by J.D. "Illiad" Frazer
MR. BANK PRESIDENT: IT IS WITH MUCH URGENCY I WRITE YOU. AS YOU HAVE INSTRUCTED I ENTERED MY INFORMATION ON YOUR WEBSITE "LUCRE INTERNET BANK" SO THAT YOU MAY SEND ME THE TRANSFER FEE MONIES FOR THE 28 MILLION (USD) WAITING IN OUR OFFSHORE BANK ACCOUNT.
COPYRIGHT© 2005 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
I HAVE DISCOVERED TODAY THAT SOMEONE HAS INSTEAD REMOVED FROM MY ACCOUNT $7,000 (USD)! IT IS MOST URGENT THAT YOU RETURN THESE MONIES!!!!! IMMEDIATELY!!!!!!!!!!!!!
I HAVE YOUR WHOIS INFO, AND WILL TRACK YOU DOWN IN THE CONGO IF YOU DO NOT REFUND THE FULL AMOUNT!!
MUST...NOT... ...SMILE...
SNERK
NORK

Practical Perl Tools

Do I Look Better in Profile?

DAVID N. BLANK-EDELMAN

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. David is honored to have been the recipient of the 2009 SAGE Outstanding Achievement Award and to serve on the USENIX Board of Directors beginning in June of 2010.
dnb@ccs.neu.edu

There may come a time in your Perl career when you find yourself asking, "Self..." (because what else would you say?), "Is there a way to make this Perl code run faster?" Many mental health professionals would say that talking to yourself is fine as long as you don't answer yourself, so let me do it on your behalf. When this time comes, you have at least a couple of options.

The first is to buy a faster machine. Unbox it, plug it in, run your code. Done. You may laugh, but that's not a bad solution sometimes. It can actually be a better option than the second one I'll propose: figure out why your code is running slowly and fix it. This option is often more fraught with peril than the first, because in the process of trying to optimize, there are many chances to introduce new bugs into the code. I could quote Knuth at you in an attempt to further scare you away, but then I don't think you'd read the rest of this column.

So how do you find out just where your code lags and how to fix it? The answer to this question in general is one of those lifelong quests. You can strive to even be a better coder, you can become a performance geek, or, heck, you can conduct basic research on how to improve programming in general. I salute those of you who already headed down any of these paths. In this column I'm going to try to help anyone else who is a Perl programmer to take another step or two in this direction.

First, I recommend that you check out Richard Foley's documentation shipped with later versions of Perl (5.10.1+) by typing "perldoc perlperf". The documentation is a little out-of-date, but it is a good start. Some of this column will overlap with the doc, but we're going to spend much more time on the current best practices that have evolved since 2008, when the doc was first written.

Benchmarks

Before we can directly answer the question "Why does my code run slower than I'd like?" I think it is important to bring our legs into lotus position and spend a bit of time meditating on some fundamental questions such as:

- What is "slow"?
- How will I know "slow" when I see it?
- Can I prove something is slow?
- Can I make "slow"?

I realize these all sound a bit more peyote-influenced than Perl-influenced, but bear with me. We need to be able to find a way to time just how fast a piece of Perl code runs. Once we have that, we need to be able to change things and see how that compares to the time it took to run the first version. Perhaps we want to see several different versions of the same code go head-to-head so we can start to get a better sense of what makes for a slow or fast implementation. I realize there is a bit of handwaving in the last statement, because it is certainly possible to speed up some code without having the foggiest idea just how you did it, but let's assume the best for the moment.

The easiest way to start with this stuff is to benchmark your code. The canonical way to do this is to use a module that has shipped with Perl ever since it graduated to version 5: Benchmark.pm. It includes the following routines (as described by its documentation):

timethis: run a chunk of code several times
timethese: run several chunks of code several times
cmpthese: print results of timethese as a comparison chart
timeit: run a chunk of code and see how long it goes
countit: see how many times a chunk of code runs in a given time

Most often you see people using timethis() or timethese() to see how long a piece of code or several pieces take to run many, many times. Modern machines are so fast these days it often requires a huge number of runs of a piece of code to get a good handle on just how fast that code might be (and to eliminate anomalies in the testing environment). Here's an example:

```
use Benchmark;
use Math::Random::ISAAC::XS;
use Math::Random::ISAAC::PP;

my $time = time();

our $xrng = Math::Random::ISAAC::XS->new($time);
our $prng = Math::Random::ISAAC::PP->new($time);

my $count = 10_000_000;
timethese($count, {
        'XS' => '$xrng->rand()',
        'PP' => '$prng->rand()',

      });
```

In this example, we load the two modules that implement the very cool ISAAC pseudo-random number-generator algorithm. The first is a Perl plus C code version, and the second implements it entirely in Perl. We then specify how many times we plan to run the test code ($count). The number 10 million here doesn't have any deep significance. I just started with 1,000 and added zeroes to it until Benchmark.pm stopped complaining about the code not running long enough to get a reliable test (ISAAC is fast). We then look to timethese() to run both the XS and the PP subroutines, generating 10 million random numbers each. The program runs and spits out a very nice result:

```
Benchmark: timing 10000000 iterations of PP, XS...
      PP: 32 wallclock secs (32.58 usr + 0.01 sys = 32.59 CPU)
        @ 306842.59/s (n=10000000)
      XS: 2 wallclock secs ( 2.34 usr + 0.00 sys = 2.34 CPU)
        @ 4273504.27/s (n=10000000)
```

As expected, the XS version is *much* faster. For fun, I ran this sample code with a $count of a hundred million. The difference in speed is even more pronounced:

```
Benchmark: timing 100000000 iterations of PP, XS...
        PP: 321 wallclock secs (320.46 usr + 0.11 sys = 320.57 CPU)
           @ 311944.35/s (n=100000000)
        XS: 23 wallclock secs (23.54 usr + 0.02 sys = 23.56 CPU)
           @ 4244482.17/s (n=100000000)
```

So there you have it, the very basics of how to do benchmarking in Perl. And I do mean "basics." Understanding how to really benchmark code (or anything) is a very detailed and intricate art/science. I bow in the direction of the people who do that for a living.

Profiling

Let's get back to the original question: "Why does my code run slower than I'd like?" We had to talk about benchmarking because it is an important tool for being able to interpret and act upon the results of the process we really want to look at: profiling. Profiling is the process of instrumenting a pile of code to determine just how long each part of that code took to run (and how often it was run). With that data it becomes easier to determine the parts of your code you could change to improve the performance.

The tricky thing with profiling is determining just what "how long/often" actually means. I know this is starting to sound like the contemplative moment from the first section, but semantics here really do matter. For example, do you care how busy you are keeping the machine (raw CPU time) or how long you should expect to be tapping your foot waiting for the job to complete (real time)? A script can take almost no CPU time, but take eons to finish running if it is waiting for data to come in from a slow outside source or if the machine itself is bogged down. Which of these two things you care about at any one time really depends on the circumstances.

If you think that's a trick question, let's continue splitting gigantic, important hairs and ask the following: Do you care how long each specific statement in a program takes to run or how long various parts of your code as a whole (e.g., the subroutines) take? And if you care about the latter, do you want to know the total time for each subroutine, including any calls it made, or do you care only about how long just the code in that routine took? Or maybe you care about all of this? (If you don't care about any of this, see you next column!)

Luckily all of this information is available to you using current Perl tools...well, more precisely, a single tool. Over the years there have been a number of profiling tools written for Perl, but unless you have a good reason outside of the normal use case, there's really only one that you'll want to consider using. Even though there is a profiling tool that ships with Perl (Devel::DProf, which has been deprecated in the latest Perl distributions), the tool of choice here is Devel::NYTProf. Devel::NYTProf is currently maintained by Tim Bunce, a name you might recognize because he's the author of the Perl DBI framework.

Just to set your expectations accordingly, if I were paid by the word to write this column, I wouldn't be very happy with this tool. Yes, you can tweak how it works with various flags, but I've never had to use them. This is one of those tools that just work well right out of the box. I won't have to go into a huge amount of detail on how to get the most out of Devel::NYTProf because it tries to give its all every time you use it. Remember all of the questions above about what sort of information you might want to collect when profiling? Devel::NYTProf provides all of them by default. It's a lovely tool, really.

The first step for using Devel::NYTProf is to run the code you want to profile, but to do so in a way that Devel::NYTProf gets loaded first so it can do its magic. Perl offers a few ways to do this, including:

```
perl -d:NYTProf yourcode.pl       # run that Devel module as the debugger
                                    code
PERL5OPT=-d:NYTProf               # set this environment variable and then
                                    run the code
perl -MDevel::NYTProf yourcode.pl # load the module first
```

Let's actually run Devel::NYTProf on the previous code we used in the benchmarking session:

```
$ perl -d:NYTProf maevebenchy.pl
Benchmark: timing 10000000 iterations of PP, XS...
       PP: 127 wallclock secs (127.21 usr + 0.22 sys = 127.43 CPU)
          @ 78474.46/s (n=10000000)
       XS: 24 wallclock secs (23.12 usr + 0.03 sys = 23.15 CPU)
          @ 431965.44/s (n=10000000)
```

Doesn't look like anything happened. Yes, the run times got slower (running under Devel::NYTProf does extract a bit of a performance penalty) but nothing else was immediately visible. However, if we look at the same directory our script is in we will see a new file there:

```
$ ls -l
total 66704
-rw-r--r-- 1 dnb dnb        343    Jul 25    15:20 maevebenchy.pl
-rw-r--r-- 1 dnb dnb   34147379    Jul 25    15:35 nytprof.out

$ file nytprof.out
nytprof.out: data
```

Devel::NYTProf has created a compressed file of profiling data. To actually use this profiling data, we have to convert it to a more useful format or a report of some sort. The distribution ships with three utilities for this purpose:

nytprofcg: Convert an NYTProf profile into Callgrind format
nytprofcsv: Devel::NYTProf::Reader CSV format implementation
nytprofhtml: Generate reports from Devel::NYTProf data

The first utility lets you put it into a format that the cool KCachgegrind utility can read. This GUI utility builds on Linux and OS X (via MacPorts or Homebrew) and Windows (see http://sourceforge.net/projects/precompiledbin/ for a pre-built version for Windows). It shows you the profiling data, call chains, and other stuff in a very pretty format. The second spits it out in CSV format, which might be useful if you are inclined to further process the data with some other tool. When I use Devel::NYTProf, I almost always use the HTML reports it provides, so let's choose that option:

```
$ nytprofhtml
Reading nytprof.out
```

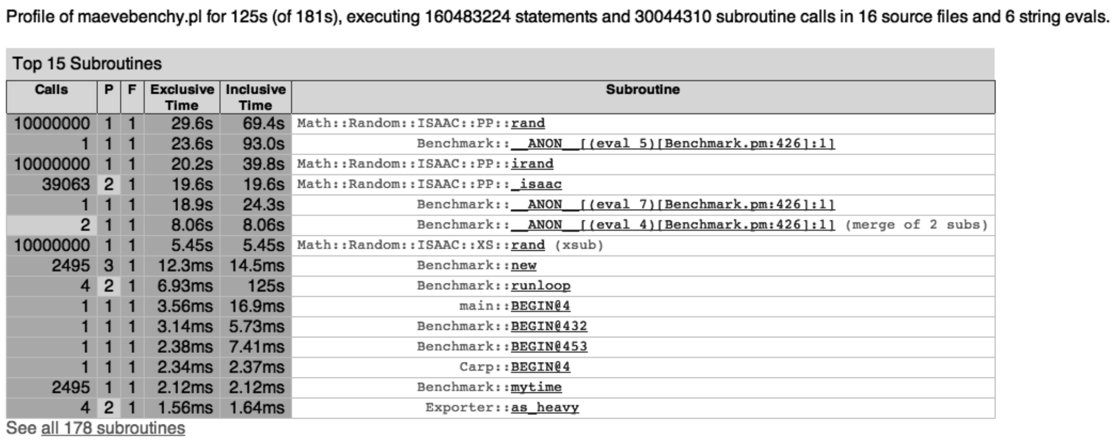
```
Writing sub reports to nytprof directory
   100% ...
Writing block reports to nytprof directory
   100% ...
Writing line reports to nytprof directory
   100% ...
```

If we look now at the directory, we see:

```
$ ls
maevebenchy.pl     nytprof    nytprof.out
```

The "nytprof" entry is a directory with a bunch (in my case 77) HTML and .dot files in it. The .dot files are Graphviz source files (a subject we've talked about in past columns), which you can translate into pretty pictures of call graphs and such using the utilities in that package.

If we open up the index.html file in a browser, we see the Devel::NYTProf top-level report, which includes something like Figure 1 (for space reasons, I've cropped the page so you can see just the first half of the report):



Profile of maevebenchy.pl for 125s (of 181s), executing 160483224 statements and 30044310 subroutine calls in 16 source files and 6 string evals.

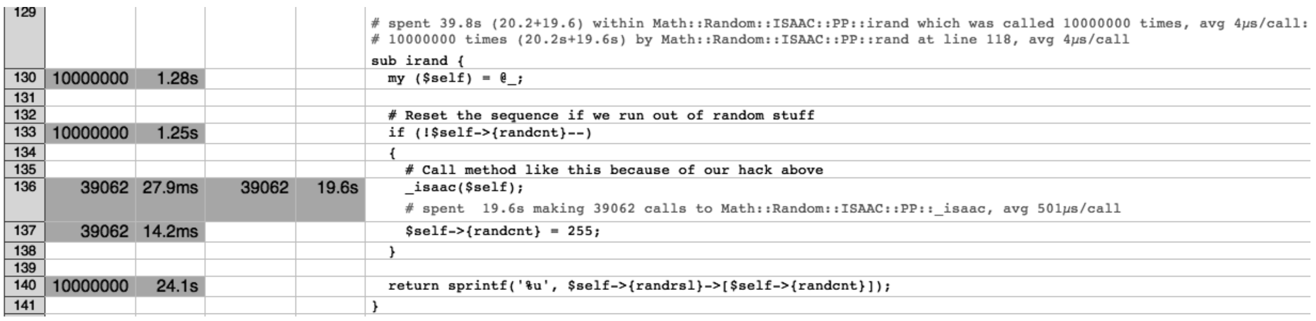
Top 15 Subroutines

Calls	P	F	Exclusive Time	Inclusive Time	Subroutine
10000000	1	1	29.6s	69.4s	Math::Random::ISAAC::PP::rand
1	1	1	23.6s	93.0s	Benchmark::__ANON__[(eval 5)[Benchmark.pm:426]:1]
10000000	1	1	20.2s	39.8s	Math::Random::ISAAC::PP::irand
39063	2	1	19.6s	19.6s	Math::Random::ISAAC::PP::_isaac
1	1	1	18.9s	24.3s	Benchmark::__ANON__[(eval 7)[Benchmark.pm:426]:1]
2	1	1	8.06s	8.06s	Benchmark::__ANON__[(eval 4)[Benchmark.pm:426]:1] (merge of 2 subs)
10000000	1	1	5.45s	5.45s	Math::Random::ISAAC::XS::rand (xsub)
2495	3	1	12.3ms	14.5ms	Benchmark::new
4	2	1	6.93ms	125s	Benchmark::runloop
1	1	1	3.56ms	16.9ms	main::BEGIN@4
1	1	1	3.14ms	5.73ms	Benchmark::BEGIN@432
1	1	1	2.38ms	7.41ms	Benchmark::BEGIN@453
1	1	1	2.34ms	2.37ms	Carp::BEGIN@4
2495	1	1	2.12ms	2.12ms	Benchmark::mytime
4	2	1	1.56ms	1.64ms	Exporter::as_heavy

See all 178 subroutines

Figure 1: Part of the index page from the HTML report generated by nytprofhtml

If we click on one of the subroutines, we can drill down into it and see more detail like that found in Figure 2 (again cropped for size).



129					`# spent 39.8s (20.2+19.6) within Math::Random::ISAAC::PP::irand which was called 10000000 times, avg 4µs/call:`
					`# 10000000 times (20.2s+19.6s) by Math::Random::ISAAC::PP::rand at line 118, avg 4µs/call`
					`sub irand {`

130	10000000	1.28s			`  my ($self) = @_;`
131					
132					`  # Reset the sequence if we run out of random stuff`
133	10000000	1.25s			`  if (!$self->{randcnt}--)`
134					`  {`
135					`    # Call method like this because of our hack above`

136	39062	27.9ms	39062	19.6s	`    _isaac($self);`
					`    # spent  19.6s making 39062 calls to Math::Random::ISAAC::PP::_isaac, avg 501µs/call`

137	39062	14.2ms			`    $self->{randcnt} = 255;`
138					`  }`
139					
140	10000000	24.1s			`  return sprintf('%u', $self->{randrsl}->[$self->{randcnt}]);`
141					`}`

Figure 2: Drilling down into the Devel::NYTProf report

As you can see, Devel::NYTProf is giving us a ton of data about what is running and how long it takes. We can click on lots of links in the reports to look deeper into what it has found. Unfortunately, in some ways, this was not the best code to profile, because it is highly artificial. We are intentionally running two specific subroutines 10 million times (via timethese() in Benchmark.pm), so it is no surprise that they dominate the profiling results.

If we were to simplify the code to just this:

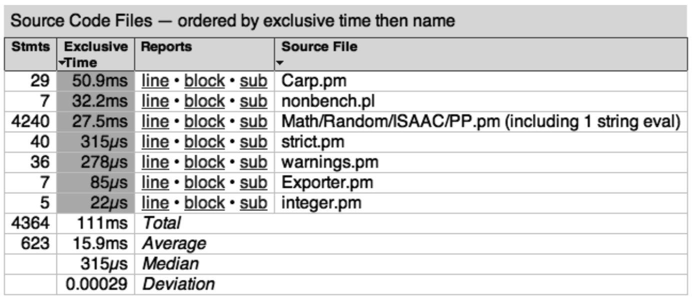
```
use strict;

use Math::Random::ISAAC::PP;

my $time = time();

our $prng = Math::Random::ISAAC::PP->new($time);
print $prng->rand();
```

and run Devel::NYTProf on it, we might get a better sense of what parts of the code are taking up time. Figure 3 shows a shot of the result, cropped this time to show the second half of the index page.



Source Code Files — ordered by exclusive time then name

Stmts	Exclusive Time	Reports	Source File
29	50.9ms	line • block • sub	Carp.pm
7	32.2ms	line • block • sub	nonbench.pl
4240	27.5ms	line • block • sub	Math/Random/ISAAC/PP.pm (including 1 string eval)
40	315µs	line • block • sub	strict.pm
36	278µs	line • block • sub	warnings.pm
7	85µs	line • block • sub	Exporter.pm
5	22µs	line • block • sub	integer.pm
4364	111ms	*Total*	
623	15.9ms	*Average*	
	315µs	*Median*	
	0.00029	*Deviation*	

Figure 3: A more informative Devel::NYTProf result

I'd encourage you to run this on your own code. You'll get a much better sense of what it is doing. Once you have that understanding, you can begin to improve it. For more information on Devel::NYTProf and how to improve code using it, I highly recommend Bunce's talk on v4 of Devel::NYTProf, a screencast of which can be found here: http://blip.tv/timbunce/devel-nytprof-v4-oscon-201007-3932242.

Take care, and I'll see you next time.

Galvin's All Things Enterprise

The State of the Cloud, Part 2

PETER BAER GALVIN

Peter Baer Galvin is the CTO 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* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter is also a Lecturer at Boston University and Senior Contributor to *BYTE*. Peter blogs at http://www.galvin.info and tweets as "PeterGalvin". pbg@cptech.com

In the previous edition of Galvin's All Things Enterprise, cloud was the center of attention. In spite of the over-hyping and frequent under-delivery of cloud, it's still an important, new, and evolving area of computing and, therefore, worth some discussion and analysis. The first task was defining cloud computing, and the next was exploring "why cloud"—what does cloud computing bring to the table and why should you care?

This column continues the analysis by giving examples of projects that have successfully used cloud computing. Why did they succeed when others have failed, and what did they gain by using cloud technologies? Of course there are reasons to avoid cloud computing, and those are included as well. The column finishes with a comprehensive list of cloud considerations—what you should consider when determining if a given project should be based on cloud technologies, and whether it would be best implemented in a public cloud, a private cloud, or a hybrid cloud, or built using non-cloud technologies.

Who Is in the Clouds

It seems to me that cloud computing started in the midst of Web 2.0. What at first blush was simple co-location—running an application or an entire business in someone else's datacenter—evolved to running the same on someone else's gear. That hosting model then further evolved into one of running multiple companies' applications within the same infrastructure. Such multi-use required better management tools, monitoring, alerting, and billing. Those became a set of cloud computing tools. Along the way, many Web 2.0 companies did very well by not running their own datacenters.

Take SmugMug as exhibit number one [1]. Their use of Amazon's S3 storage cloud is a case study in how a company can reduce overhead, costs, and complexity [2]. This photo sharing and management site stores the photos via the APIs provided by Amazon, while retaining the customer information and metadata within their own computers. Certainly that's an example of a company that should use the cloud and an application that was ready-made for cloud integration. Necessary cloud attributes (that it's elastic, metered [pay as you grow], shared, and Internet-based) are all present in this case. Of course, SmugMug is one of thousands of Web 2.0 companies that base their computing or their storage on the cloud.

Back during the dot-com boom, companies needed a lot of venture capital, as well as a lot of IT knowledge (either internal or for hire), to move their idea from paper to full-scale execution. Now, the idea still needs IT knowledge, but requires less funding and less investment. Fortunately for companies looking to be in the cloud, there are many providers trying to add to their client list and take their money. The leaders include both tried-and-true companies such as Amazon, Microsoft, IBM, Google, and VMware, and newer companies such as Rackspace, Salesforce.com, Joyent, NetSuite, 3Tera, Terremark, and GoGrid. These companies vary in their offerings, pricing models, and abilities, but all provide IT resources via a pay-as-you-go model.

My Cloud or Your Cloud?

As good as the public cloud model is for some companies, it leaves many other companies wanting. Questions about reliability, security, and performance, as well as regulatory requirements and corporate policies, prevent many companies from utilizing the public cloud products. Given the recent, very public cloud failures [3], companies that depend on their data and computing resources to be available or under their control are choosing not to use the public cloud, or at least not to use it for large swaths of their computing needs.

Such a choice does not mean these companies cannot have public cloud-like features for their projects. Companies still desire the elasticity, manageability, rapid deployment, and even chargeback (or the no-payment-required version known as viewback). What are such companies to do? The solution for them is to use cloud technologies within their own datacenters—private cloud, in the parlance of our times. Sometimes companies want to use private cloud, as well as using public cloud facilities where applicable. This configuration is called hybrid cloud.

Private clouds can look a whole lot like what we used to call "infrastructure," but there are some implementation choices and technologies that can give them cloud-like aspects. Consider one of my clients that had the following problem. Client X had a small, full datacenter. It was traditional in that there were dedicated servers for each application, a small SAN for all important data, a tape library for backups, and a 1 Gb network for interconnection. X was growing, needed to move to a larger datacenter, needed a DR plan beyond just shipping tapes off-site, and needed to move quickly to respond to new business-driven IT initiatives. They chose to use a co-location facility to provide ping, power, and pipe for their racks of equipment. Other improvements included moving to VMware ESX to layer applications across a pool of servers, and using NAS storage to hold their production data as well as the virtual machines. The NAS array also provided them with replication of the data to a second NAS array at a second co-location facility for DR. Moving to a 10 Gb networking interconnect gave them better performance and more room to grow without running out of throughput. The project also involved deploying tools to enable release management, configuration management, capacity management, and change management based on the virtualized environment. Should this project rightly be called a next-generation infrastructure or a private cloud? Both are correct, but because X now has infrastructure-as-a-service (IAAS) and service management for their application deployment, as well as elasticity, I believe it is a private cloud.

As another example, consider client Y. They had an existing business continuance (BC) plan, but that plan failed when it was needed the most—during a disaster. They could not gain access to their normal offices, so declared a disaster and switched over to the disaster recovery (DR) site. Workers started arriving there, and all was well until the number of workers increased. The plan had been tested, but not at the scale of the entire company. The DR infrastructure fell over and work could not proceed. After sorting through the various options, Y decided to upgrade their BC plan and facilities. Rather than have workers go to the BC site, the workers would work remotely, across encrypted tunnels, using a virtual desktop infrastructure (VDI) facility. The applications run within their BC site, but the workers get remote views of their virtual desktops from anywhere that Internet is available. Because of underlying virtualization, production applications are replicated to the BC site, so all apps are kept up-to-date. Internet technologies allow remote access, and by adding more CPU and memory to the BC farm, they can easily scale the facility as needed. Again, this could be labeled with various names, but private cloud is certainly one of them.

Cloud Candidates

Are there certain application and IT initiative aspects that predispose them to be best deployed in a public cloud, private cloud, or left as is on traditional infrastructure? There certainly are trends and success (and failure) stories that show that some projects are better matches for cloud than others. While there are not any absolute rules, a project involving these aspects is probably a good fit for a public cloud:

- Software as a service
- Audio/video/Web conferencing
- Sales/CRM automation
- BC/DR
- Training/demonstration services
- Collaboration
- Email
- Development/test facilities

Other aspects show a tendency to be best left in a private cloud:

- Large data movement
- Sensitive data
- Regulated data
- Complex processes/complex transactions
- Low latency requirements
- Non-x86 applications

Yet other aspects may reveal projects that should be left on existing infrastructure:

- Legacy applications
- Industry-specific applications
- Real-time applications
- Very large (CPU, memory, data) applications

As with cars, your (project's) mileage will vary. Every site is complex, with many decision points, criteria, and experiences. All that will provide guidance on what to place in cloud infrastructure and what to leave as is.

Cloud Considerations

In my experience, it is possible to codify at least some of those "what to run where" decision criteria. The following set of guiding factors can be useful in applying logic to the task of determining how best to run a given application or given facility.

For each of the following technology areas, you should decide whether the area is a factor or not. If an area is a factor, then document why. For example, the operating system might not be a factor because your application can run on any OS, but networking might be a factor because it's 1 Gb and you need to move to 10 Gb for the throughput your application needs. The list includes: operating systems, applications, servers, storage, networking, Internet technologies, virtualization, logging/reporting/analytics, mobile access, seasonal resource use, elasticity/scalability, and any other technology criteria that might be important to your site.

Next is a set of design requirements that could steer the project toward one type of infrastructure or another. This list includes large data movement, non-virtualizable software, low latency, and high customization requirements.

On the financial front, the following areas could be rated in terms of importance, from not important through very important: reducing OpEx, reducing CapEx, licensing cost reduction, ROI requirements, and chargeback/viewback requirements.

Another area to consider is the line of business that the application or facility is destined to support. The LOB again might have importance ratings in areas such as keeping the infrastructure separate from others, required SLA strength, capacity or performance guarantees, the need to control recovery from problems, automation of workflows, and self-service abilities.

In the risk and regulations area, some factors you should consider are the inclusion of validated systems, regulated data, sensitive/proprietary data, regulated systems, HIPAA/SOX or other regulation compliance, corporate security policy requirements, and whether there are strong security needs.

In the final area of project execution, you should think about whether staff members have the skills to design and implement the project within the facility selected, whether they can do so within any time constraints, and whether the team has the knowledge and tools for ongoing monitoring, maintenance, and management of the facility.

Beyond these considerations, don't forget any site-specific, project-specific, or staff-specific requirements or limits on the broad issue of where to run the facility. Experienced IT managers know that beyond those broad decisions, a project succeeds or fails based on the myriad of details it encompasses. Cloud is not a panacea that removes the need for planning and execution. In fact, cloud computing can place more emphasis on management, teamwork, decision-making, and debugging than more standard projects do.

One final note: cloud computing is important and is changing how infrastructure is built and used and how much it costs. That does not mean that cloud computing can solve all problems or is right for all environments or all projects. Sometimes the internal structures of a company or the ways in which roles and responsibilities are divvied up can mean the difference between success and failure of a cloud-centric project. Many companies are finding that between politics and those old structures, much internal change is needed in order for the company to embrace cloud computing.

Tidbits

If you are interested in performance analysis and debugging, especially based on DTrace, you should have a look at the new *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X, and FreeBSD* by Brendan Gregg and Jim Mauro. It's everything you could want in a DTrace book. See my full review in the Book Reviews section of this issue.

If your interests lie more in the direction of ZFS, then you might want to check out my first video publication. This one is based on the Solaris tutorials I've taught many times for USENIX and elsewhere. The official name is "Solaris 10 Administration Workshop LiveLessons (Video Training): File Systems," but it's 90% ZFS, including both theory and hands-on examples of configuring and using it [4].

On another front, I'm pleased to be part of the relaunch of *BYTE*. As a young lad, I spent many an hour poring over the pages of the venerable magazine, delving deeply into technology details of many aspects of computing. *BYTE* is back, and I'm one of the Senior Contributors there. Have a look at http://byte.com and let me know what you think.

References

[1] SmugMug: http://www.smugmug.com.

[2] SmugMug case study: http://aws.amazon.com/solutions/case-studies/smugmug/.

[3] http://www.crn.com/news/cloud/index/cloud-outages-cloud-services-downtime.htm.

[4] Available at http://www.informit.com/store/product.aspx?isbn=0321753003#Lessons and http://my.safaribooksonline.com/video/-/9780321718372.

iVoyeur
Cloud Gazing

DAVE JOSEPHSEN

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

My Mom, an aerospace engineer by trade, married a physicist when I was 13 or so. As a result I spent some time in my youth hanging around some very cool laboratories at places like Honeywell, Hughes, and Raytheon. There is a curious sort of euphoria shared by scientists and engineers in labs like this, places where interesting work is getting done. It's this feeling, I suspect, that would tempt so many of us to take a janitor's position at CERN, if only to be in the building for a while. I'm a little ashamed to admit that I experience the same sort of bliss when I'm at our co-location facility (which is often). A good friend, former coworker, and fellow pipe-smoker runs the NOC there. When I visit, we'll usually sneak out back, behind the generators, and smoke a bowl together.

I enjoy the banter and the opportunity to exchange whatever tobacco mixtures we're experimenting with, but I also relish the walk to the generators, which affords me the opportunity to pass all manner of strange and colorful appliances (NINJABOX); a half-dissected EMC Clarion (what ARE those guys up to?); a huge, distributed commodity storage installation (BLUE LEDs as far as the eye can see!); the battery room (look at the SIZE of those ground wires!); and of course the 10-ton diesel generators. I'll leave it to the reader to imagine a felicity to match a good smoke with a good friend on the concrete foundation of a 10-ton generator.

At any rate, a few weeks ago, on one such walk toward the generators, a new installation drew my curiosity. By our admittedly meager standards, it was quite large. Rack upon rack of 1U commodity boxes of a brand that rhymes with "blooper psycho." Upon being told it was Akamai, it was driven home to me just how much things had changed in the last five or six years. Remember when the guys with money bought Sun or IBM, or at the very least Dell? Now there is no Sun; only Oracle. And, anyway, the really big guys don't even use Oracle. They've all been hacking away on their top secret MapReduce and DHT implementations for years and years. SANs used to come in pretty boxes with fancy labels on them, and whatever happened to "blades"?

What we're seeing in the colo is just the physical manifestation of what we all know to be true. In the past several years the Google Commodity Hardware Model, configuration management engines, machine and storage virtualization, and, most recently, the NoSQL [1] movement have come together in ways that have changed our profession.

New companies of every kind are as likely as not to turn to a cloud provider or two for their server infrastructure needs. At the same time we appear to have accepted the failure of relational database systems to scale beyond a certain point, and most of the high-end database research is going into distributed key-value type systems. The big iron is gone and from its ashes have sprung a bazillion no-name 1U boxes, most of which play host to some 30-odd virtual machines. Did you know that you can install Cisco IOS on VMware now? The 1U sea of blooper-psycho engulfs even the switches before our very eyes!

I see in this future some fascinating ramifications for systems monitoring, but, alas, being a guy who just runs a bunch of Linux boxes by trade, I'm at the sidelines of most of this spectacle. This month, therefore, instead of subjecting you to my own conjecture, I've decided to borrow from the insight of someone closer to the center of the tempest, Theo Schlossnagle. Theo is the author of *Scalable Internet Architectures* (Sams) and is the founder of OmniTI. Among OmniTI's offerings is a product called Circonus, which is a hosted monitoring service. He was nice enough to answer the following questions for me via email:

[Dave] Let's start with a quick rundown on OmniTI: What sorts of problems do you solve for your clients? What sort of scale are we talking about?

[Theo] OmniTI covers a fairly wide gamut of Internet architecture challenges—everything from datacenter selection and network and systems infrastructure design and management up through software engineering and into design and usability. All of these capabilities are squarely focused on Internet architectures that service millions to hundreds of millions of users. I focus mainly on data, concurrency, and distributed systems issues at that scale.

[Dave] Circonus and Reconnoiter: what are they and what deficiencies were they built to satiate?

[Theo] Reconnoiter evolved out of a long life of pain experienced by our operations team while managing large, distributed architectures. I was very unhappy with the state (and philosophy) of existing monitoring tools, so I set about to fix that. After running Reconnoiter for a while, supporting various high-stakes Internet sites, we started Circonus to bring the features of Reconnoiter and the philosophies of our engineering and operations teams to the rest of the world within the ever-popular and vibrant world of SaaS.

A few key differentiators of our approach include separating data collection from data processing, structured and safe fault remediation workflow, and data agnosticism. What's all that mean? Basically, you don't have to collect metrics in one system to tell you if they are good and bad (paging someone) and then go collect them in another systems to do visualization. It means you can't acknowledge an alert while sleep deprived and subsequently forget that you did so. Perhaps most importantly, it means that you can collect data from other parts of the organization (such as productivity information, finance information, sales and marketing data), so that the engineering and operations teams more directly know how their work impacts KPIs across the organization as a whole.

[Dave] Describe your typical customer. What factors, in your experience, contribute to a company deciding to employ a hosted monitoring system? Do most of your clients use hosted monitoring solutions exclusively?

[Theo] Yet again, we've built a product that doesn't target a market vertically. While frustrating on the sales and position side, it is very rewarding to realize a system that is so useful to every data-driven organization on the planet. Our customers are those who want more insight into inputs and outputs in every facet of their organization and believe that the great transparency of that data leads to higher performance and greater accountability. In my opinion, that describes every organization on Earth.

[Dave] Can you talk a little bit about where you see monitoring going in the near future? Do you expect to see hosted monitoring systems replace traditional in-house monitoring, or do they merely augment it?

[Theo] Monitoring systems need to be smarter. For the past 15 years (of my experience), these systems have provided excellent information. It's about time they provide some insight too. The in-house vs. SaaS model is an interesting system. As soon as you stop to think about it, it's pretty obvious. The monitoring system is designed to provide information in good times and in bad. In those bad times, the last place you would ever want your monitoring system located is "in the problem." Off-site, externally hosted monitoring systems are the only right way to do it.

[Dave] This is complicated by issues of data security. So, the big companies that need to control the location of their data will spin up their own, privately managed, external monitoring systems. It will work and look like typical SaaS, but will not be multi-tenancy. I believe the world will go SaaS on this front (although some will use private SaaS).

Do you see cloud customers who are working to eschew an IT infrastructure of their own, turning to hosted services to fulfill their monitoring needs, or rolling their own solutions in the cloud?

[Theo] The only consistent thing I see is people not knowing what to do. I think education is paramount in the world of monitoring. We see a good mix of people trying to launch their own monitoring systems in the cloud alongside the rest of their infrastructure and others leveraging SaaS. I think you just need to ask yourself: am I in the monitoring business?

[Dave] I've heard it said that the cents-per-cycle pricing models used by the cloud providers are incentivizing users away from agent-based monitoring tools and toward external polling engines. Do you think that's true? If it is, does it bode poorly for agent-based systems in general in the future?

[Theo] Agent-based systems and polling systems are both very useful. Any monitoring system that doesn't support both is missing core functionality. I think using the term "agent-based" is misleading here. Instead, think of it as active vs. passive. Is the monitoring system asking the resource a question and receiving an answer? Or is the resource notifying the monitoring system unsolicited? Both are valid, both are needed. I think a monitoring system that only supports one of these methods has a dark future indeed.

[Dave] As sites are getting bigger, we're starting to see more interest in monitoring methodologies that employ statistical sampling (sFlow, for example). Do you guys think it will eventually become infeasible or even meaningless to constantly poll a service on every host in a large environment? What does monitoring for service availability look like for large-scale applications in the future?

[Theo] There are a lots of ways of collecting data, including both sampling and polling. Each is useful in its own context. It's a rather simple question that leads one to the right methodology: is basic statistical information (mean, stddev, cardinality, etc.) over the last bit of time enough to answer my questions? or is that little bit of data misleading? Looking at something like temperature: I can sample that hundreds of times per second. Is knowing only the average and standard deviation of that over the past 60 seconds misleading? The answer to that (in almost every environment) is no. The average is entirely sufficient and I don't need a breakdown of those samples.

On the other hand, let's look at something where the events are highly decoupled, such as database query performance or Web page load time. Thousands of these events happen each second. Can just the average and a few other token statistical aggregates be misleading? Yes. Here, you can do sampling to get an idea of individual events or (what we do in Circonus) collect histograms instead of simple statistical aggregates, which provides much richer insight into the populations that don't follow a normal Gaussian or gamma distribution.

As with all other things, use the right tool for the job.

[Dave] I've noticed several hosted monitoring services that are themselves implemented in EC2 (browsermob, for example). Will you share your thinking on monitoring from within a cloud provider? Are you guys using any cloud services currently?

[Theo] Perhaps I'm a bit old-school, but there's nothing more important than your monitoring systems being up. The reason we don't use *one* provider is for exactly that reason. I want different bandwidth providers, different routes, different SLAs, and different datacenters for my components. The rule of "no single point of failure" is a simple one; just remember that a vendor is a point of failure. This includes some things people rarely think of: disk vendors, equipment suppliers, datacenters, and cloud providers.

[Dave] With the amount of server and network machine virtualization in use at EC2, it isn't difficult to imagine that a good portion of the monitoring traffic destined for it might be redundant and therefore useless. I'm imagining, for a simple example, 20 ICMP packets from four different customers, destined for 20 addresses that all resolve to VHOSTS sharing the same physical NIC. I'm also imagining the apathy a typical cloud customer probably has with regard to their own monitoring overhead. Do you guys anticipate the inter/intra cloud monitoring burden becoming a problem for the service providers? Is there any potential for systems like Circonus to attempt to detect and optimize for redundant or equivalent outbound monitoring traffic?

[Theo] On the active monitoring side (such as ICMP checks or HTTP requests), I don't foresee any undue burden. Circonus already optimizes the high-frequency cases where several people are running the same check for second-to-second monitoring. We do this to limit the burden on our internal systems more than on those being monitored. When comparing the number of packets/requests in/out from duplicitous monitoring versus regular heavy traffic, we find the monitoring burden to be nominal.

[Dave] I really am fascinated by the changes taking place in database technology lately. Key-value and distributed hash table architectures appear to be all the rage, for the simple reason that they scale horizontally. We used to wonder how we were going to get ACID to scale, and now we appear to have thrown it out the window. Most of the distributed data-tier schemes I read about seem to consider only one failure model—that of some number of nodes failing completely. I personally have witnessed all manner of strange quasi-failures in application tier nodes, so I find this worrisome. Do you guys have any experience with these systems? Can you talk about their reliability versus the old ACID approach? Are they good enough for the banks?

[Theo] There's a lot of confusion between "old ACID" and "new distributed" systems. First, both are quite old. Naming one as old implies that it is less viable. Both systems are very useful. The resiliency profile achievable with well-designed distributed systems will always be better than ACID-compliant systems (at least, we theorize that now). ACID systems will continue to provide a simpler mental model within which engineers can operate. Database failure is far less relevant than service failure (business services, that is). I have seen (and designed) many systems where a failure at the ACID database level causing a complete database outage is not noticeable by the user. These systems are not obsolete. That said, I have a distributed systems background; when you need resiliency, technologies like Riak and Voldemort are really what's needed. Are they good enough for banks? Sure. Both make promises and keep them; the failing is usually with engineers who do not understand what those promises really mean. Distributed systems are hard.

[Dave] With traditional relational database systems, it was easy enough to have Nagios perform a query against the database server, but now how are we to monitor for data consistency in a post-ACID world with 500 Voldemort nodes across multiple cloud providers? What does the future of monitoring look like in the context of the data tier?

[Theo] Monitoring is monitoring. You should care about a vast number of implementation-specific metrics within your database system (relational or non-relational). You also should care about its overall function, quality of service, and availability. If you can measure those, so can your monitoring system. This problem is not difficult.

[Dave] It's conceivable that a modern distributed database infrastructure may grow organically to encompass multiple co-location facilities and/or cloud services. How can we ensure that a given performance metric is meaningful when the database might respond with greater latency to different network ingress paths? How do we monitor to discover performance problems like this?

[Theo] Expectations are evaluated by measuring and analyzing these metrics. Today, these metrics and expectations are set by humans. The mechanics of monitoring these things are very straightforward. Deciding what your expectations/SLA/QoS are for your datacenter distributed system is a much more involved human problem. Again, distributed systems are hard. OmniTI can help with that, if you're struggling.

[Dave] Can you talk a little bit about how Circonus is architected? I'm particularly interested in data storage. Where did you hit the ceiling with RRD, and how do you guys intend to scale the storage of the metric data you collect?

[Theo] Circonus architecture is quite complex, as it provides many services, and the system is highly decoupled to support our rapid development model and fault-tolerance requirements. Data storage is a bit easier to discuss.

Most of the data collection in Circonus follows the path of the open source Reconnoiter system all coming to a head around a set of processes called stratcond. stratcond is simply responsible for securely pulling data from the field and ingesting it into a storage system. Internally, the storage system we use here is called "snowth." It is a custom-built database that borrows the distributed principles behind Dynamo and storage structures around time-series data so prevalent in the financial services industry. The system is written in C (with extensibility in Lua) and provides zero downtime on instance failure, seamless node addition and removal, complex data analysis server-side in Lua, and a highly optimized on-disk time series data format that never loses granularity.

[Dave] Can you talk a little bit about OmniTI's direction of combining business intelligence with traditional performance monitoring and fault detection? Any interesting statistical models at work here?

[Theo] Interesting is certainly in the eye of the beholder. Basically, every model we've used successfully has also failed and provided misleading results. Statistical models are very dangerous in the hands of anyone but a highly qualified statistician. Mainly, we use statistical models and their outputs heuristically. Each question deserves its own answer, and one should never assume that one model applies equally well to different questions. It may, but the assumption is usually what leads people to irrational conclusions.

[Dave] It seems to me that if you're in the business of collecting metrics for a large base of users, there is some potential to analyze the aggregated data to detect larger problems like DDoS attacks, BGP convergence issues in the Internet, or failures within a given cloud provider. Do you guys see any potential in leveraging the information you're gathering with Circonus to solve larger problems?

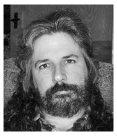
[Theo] That's an interesting question and one that has certainly crossed our minds. We treat our customer's data as sacred, on the retention and integrity side as well as on the privacy side. So we are somewhat limited in what we can do there. We are able to observe these things from the metrics we collect on our own systems (collected by Circonus itself, of course). The recent EC2-east services issues were about as obvious as a nuclear explosion next door. There are already very good tools to detect DDoS and BGP convergence issues, and while those are easy to observe in the data flows in Circonus, I think I'd like to go for something more insightful than that. What? That's the question, isn't it? I'm not sure we have the answer to that yet.

We're actively working on some sophisticated intelligence algorithms that can detect anomalistic points in time-series data in the context of all our data streams with all of the metadata we have attached to each metric. It's a sea of interesting data, but I'd be lying if I said we've made good sense of anything but the surface; we're very excited about our future exploration of this sea.

Did I mention we're hiring data analysts with strong signal processing and mathematics backgrounds?

/dev/random
Virtualization: A Dark Proverb

ROBERT G. FERRELL



Robert G. Ferrell is a fourth-generation Texan, literary techno-geek, and finalist for the 2011 Robert Benchley Society Humor Writing Award.
rgferrell@gmail.com

In the beginning, there was void(). At length the Divine Programmer moved *as yet not established third person singular possessive gender-neutral pronoun* hand over the firmament, causing a Disk Operating System to emerge fully-formed from the swirling darkness and thrumming chaos and lo, the disks did begin to operate. Poorly, slowly, haltingly at first—but verily didst they operate all the same, reveling in their intrinsic diskhood.

The Word was written in stone by a fiery hand, and that Word was: "one machine, one operating system." The faithful took the Word into their hearts; for many years the Word was orthodox and orthodoxy ruled the land.

The first heresy was *dual booting*, and lo, it brought great suffering to the faithful, as it presented a divisive message preached in seductive whispers on false jasmine-scented zephyrs, and thereby was the flock split into twain. While the purity of the Word had been sullied, yet the Word remained in spirit, for only one operating system could be invoked at a time. The flock, though divided, was yet the flock.

It was common knowledge among the faithful that shadowy figures lurked down every dark corridor and within every forbidding crevice strewn with ensnaring webs and belching forth toxic clouds of putrid malady. From deep inside one of these pits of perdition came one fateful day a multi-headed beast of depravity, a hateful hydra of heterogeneity. The beast spread its terrible tendrils across the bosom of the land; the faithful saw it and were sore afraid. They named this terrifying manifestation *vir tual*, which in the Old Language (ALGOL) meant "demon without mercy."

As the affliction spread from enclave to enclave, the faithful were tested as never before. Where once the single operating system model had been the gold standard under which all useful computing activity was performed, now multiple operating systems could be run *simultaneously* on the *same machine*. What devilry was this? When XP, RHEL, Solaris, and BSD could be brought to life at once sharing the same hardware, how could the faithful hope to maintain order in the universe? There was much wailing and gnashing of teeth in the months following the advent of The Beast. The faithful scrambled to discredit the heresy, threatening early adopters with eternal damnation or at best a painful charley horse, but they met with little success. The disease would not be eradicated so easily. It was deeply entrenched, like that black sticky gunk under your refrigerator.

The faithful convened in plenary session (pastries and bottled water provided for paid members only; parking not included), desperate to combat the growing menace which threatened to destroy all that was Good and Holy and Monolithic. They investigated ideas, circulated suggestions, traded PowerPoint presentations, delivered directives, pondered policies, considered consultations, and generally just thought the heck out of the dilemma. No solution presented itself. Finally, the high priestess hung her head and announced, sadly, "We have failed, brothers and sisters. The angry clouds of virtualization have flooded us with tears and darkened our sight with grief. No longer will we float on Cloud Nine when we speak of our processing prowess. The Open Source of evil has clouded computer users' judgment and turned them against us, we who have guided them so long and so well, though fair sky and cloud." At this a small voice in the back piped up. "When life gives you lemons, make lemonade. We have been assailed by these hateful clouds; let us therefore give as we have been given, in like manner." And the assembled brethren saw that this was Good Thinking.

Thus was born a daring plan, a bold initiative, a veritable vehicle of vengeance.

"We will cast the demon virtualization out of the tangible hardware realm and scatter it across the firmament," the high priestess cackled. "We will eradicate determinism from data processing altogether. The enterprise will cease to exist, as will the concept of control over computing resources. User interfaces shall become razor-thin clients—so thin that they will all but disappear in poor light. Users will cast their precious data adrift in digital bottles on a vast sea of virtual machines, hoping against hope that it may be retrieved, processed according to their wishes, and returned to them on the tide, somehow intact and inviolate. They will realize one day that confidentiality and integrity cannot be assured in this model and demand that control of their data be returned to them—and on that day we shall triumph and order be restored." The brethren rubbed their hands together and dreamt of victory, however hollow.

At length their twisted vision did indeed come to pass. The vision took on a life of its own and spread eager, grasping tentacles across the whole of the enterprise computing landscape, but as with every edifice built on a corrupt foundation, eventually the walls began to crack and plaster to crumble, scratching the wainscoting, gouging and leaving ugly smudge marks on the baseboards. Rainwater seeped in, forming mildew-ridden stains and necessitating costly repairs by contractors of dubious licensure.

Alas, the faithful had not reckoned with the immense influence exerted over the collective psyche of users from advertisers employed by the firms who stood to gain most from the abomination that was cloud computing. Little by little they chipped away at the notion that data integrity and confidentiality were desirable, ignoring or actively reviling any who propounded in a contrary manner, until users began blindly to accept their assurances that any data placed in the cloud using their product was "secure." The corporate overlords were then free to use that enormous pool of critical user data for whatever purposes they saw fit. Lo, did profits soar to record levels.

Once the populace were conditioned to accept the first preposterous assertion, succeeding bamboozlement operations became much simpler, even routine, until eventually the idea that they should lie back in their climate-controlled designer capsules and dream of nirvana whilst they served as an organic energy source for the kind, helpful machines that now controlled every aspect of their lives seemed perfectly reasonable.

The Divine Programmer beheld the enslavement of the people and shook *as yet not established third person singular possessive gender-neutral pronoun* head in sorrow and vexation. "Perhaps," *as yet not established third person singular possessive gender-neutral* (I'm getting tired of writing this) *pronoun* declared, "next time I will stop at dinosaurs."

BOOKS

Book Reviews

ELIZABETH ZWICKY, WITH TREY DARLEY, SAM STOVER, AND PETER BAER GALVIN

Beyond Bullet Points, 3rd Edition

Cliff Atkinson

Microsoft Press, 2011. 240 pp.

ISBN 978-0735620520

I am hoping that, someday, the existence of resources about how to create presentations that are not mind-numbing and the existence of which millions of people and the rising tide of presentation hate (mostly directed at PowerPoint) will have a result. So far, what I have gotten is people who make defensive jokes about "death by PowerPoint" and then put up giant heaps of classic PowerPoint slides—but with clip art of people.

This book is aimed squarely at the worst perpetrators of death by PowerPoint, which is probably good for relieving my future suffering, but does mean that I found it had an offputting marketing flavor at times. Oddly, the fact that it is intentionally PowerPoint-specific, and I prefer Keynote, didn't bother me at all, possibly because it leans strongly on features that Keynote implemented before PowerPoint. Users of older versions of PowerPoint are going to be sadly out of luck, however, since the techniques presented are based around both presenter mode (where you see your notes and the slides, the audience sees just the slides) and multiple masters. Keynote users will have to do some translation and do some things by hand that the author provides scripts for, but it doesn't look horribly onerous.

I probably won't adopt the approach wholesale, but I did pick up some information that will change the way I do slides (possibly with the exception of the cases where I was already being intentionally outrageous). I like the emphasis on story, on speaking casually, on consistency of metaphor, on considering the audience, on not cluttering up your slides. I adore the idea that it discusses actual research, with references and everything, and while it probably oversimplifies the neurology, it's not horrifyingly improbable or irrelevant. (This is not as low a bar as it may seem.) Instead of saying "90% of communication is non-verbal!" it cites particular studies that showed specific improvements for slides with relevant pictures.

It also tries hard to teach people who are used to working with words how to think of images and how to find and choose images. Too many books tell you to use pictures but leave you with no idea where to get the things from, which leads to lame clip-art, copyright violation, and desperation. I'm not sure that the advice on image and on metaphor selection are sufficient for people who don't have experience, but those are difficult issues to teach, and at least *Beyond Bullet Points* tries.

If you are looking to learn how to give presentations well, and you're willing to take it seriously, I recommend that you read this book. Even if you don't end up using this system exactly, you will get a good idea of the important issues. The system described is a lot of work, but it's not make-work; doing presentations well just is a lot of work, no matter how you do it.

Me and My Web Shadow: How to Manage Your Reputation Online

Andrew Mayfield

A&C Black Publishers, 2010. 185 pp.

ISBN 9-781-4081-1908-2

This is a book for people who are not terribly technical on managing their Web presence, from the point of view of somebody who is a real netizen. That's not a word I find myself using often, but I don't know how else to talk about somebody who gets the Internet without being technical. He's a marketing guy who's also capable of non-judgmentally advising that buying posts for your blog is probably a really bad idea and that if you want to be part of a community you are going to have to do real work. (He's a UK marketing guy, and I was taken aback initially by his saying that a good reputation required "graft"; that would be hard work, not bribery.)

Although I'm not the target audience, I found some of the advice useful and the book as a whole reassuring and appropriate. It encourages people to view the Internet as a public space where you want to exercise some care and taste and behave like a responsible, contributing human, and it gives you advice on how to do that (including on how to recover from errors). I would certainly offer this book to friends who were thinking maybe they wanted to get some control over this Internet thing and get some use out of it, and feel confident that it would lead them, gently, towards behavior I consider appropriate.

I have some quibbles, of course. It's a bit Google-centric for my taste (note that I'm employed by Yahoo! and may be more sensitive to this than other people). And while I appreciate the thought, I don't really believe that people should change their passwords every six months; I believe they should make them not completely stupid and different everywhere.

Head First Networking

Al Anderson and Ryan Benedetti
O'Reilly and Associates, 2009. 487 pp.
ISBN 978-0-596-52155-4

On the up side, this is a networking book aimed at people teaching you to do actual TCP/IP networking, instead of teaching you to pass a certification exam. It does a good job of walking you through how routers and switches work, and what subnet masks and routing tables actually do, which are difficult concepts for people. It takes you through real problems that network administrators face, emphasizing troubleshooting and network design, which are the hard parts, and leaving the OSI model, which almost never clarifies anything for anybody, to an appendix.

It has one major down side, which is that it has an amazing number of errors. Most of them are just annoying, but some of them have the potential to actually confuse readers; I would only recommend it to relatively resilient learners. For instance, it has a nice example about tracing an intercepted message back to the computer that originated it, but when you get as far as the router, there is a missing explanation, leading to a baffling moment where the problem is solved by an apparent miracle. And it says that routers protect you from MAC address spoofing, which is not precisely wrong but is totally misleading. If there's a router in the path from client to server, the server won't be fooled by MAC spoofing at the client end, but only because the server cannot pay any attention to the MAC in the first place, not because the router is exerting some protective effect.

There is also a design oddity—*Head First Networking* is electron first networking, with quite a lot of discussion of waveforms on cables and how you turn electrical impulses into ones and zeroes before you ever get to the packets. That's a justifiable decision, and there are certainly people for whom it is the best approach. There are also people who are going to tune out somewhere around the multimeter. That's probably a mistake; try skipping ahead to the packets and see if it picks up for you.

While this is never going to be for everyone, if it were fixed, it would be a great resource for many people. It's a much more realistic introduction to networking than most resources.

Hunting Security Bugs

Tom Gallagher, Bryan Jeffries, and Lawrence Landauer
Microsoft Press, 2006. 592 pp.
ISBN 978-0-7356-2187-9

I feel kind of silly recommending a security book from 2006. Five years is a really long time. Unfortunately, the changes in many of the topics the book covers are minimal. XSS? Check. SQL injection? Check. Buffer overflows? Apparently as resilient as cockroaches. Sadly, people are even still running five-year old browsers. Some things are guaranteed to be timeless; canonicalization is good, blacklists are bad. The net result is that *Hunting Security Bugs* is Microsoft-oriented and leaves out some important attacks (XSRF, clickjacking, Flash and PDF vulnerabilities) but still manages to provide a solid introduction to what people get wrong and how you find it.

If you have a significant technical background and are interested in securing software on Microsoft platforms, this is a good place to look. It will show you how to think like a security tester and introduce relevant tools. It also goes through common counter-arguments ("But nobody will make a hostile server!"). It goes into the innards of most things enough to show you why things are vulnerable. It assumes the reader is capable of reading C++ code and has some basic idea what things are bad (it does not explain terms like "escalation of privilege"), but it does not assume much network background.

Pro Puppet

James Turnbull and Jeffrey McCune
Apress, 2011. 318 pp.
ISBN 978-1-4302-3057-1

Perhaps, like me, you have been a mite stymied by the endless debates between proponents of different configuration management tools over the past decade. It seems that config management is destined to be every bit as partisan as choice of editor. If you've been in the field a while you've doubtless noticed there's been a steady uptick in "ssh and a for loop" jokes in the past couple of years. I sense that we passed an inflection point in the past 18 months or so. (Sure, there's still plenty of room for theorists to debate how many angels can dance on the head of a DSL.) But seeing how young people coming to the field seem to view being a sysadmin as passe and insist they are a breed apart—the devop—it seems that the tools themselves are damn near cooked.

If you've been scratching your head over the Puppet-Chef-CFEngine-Bcfg2-LCFG-etc. question while holding your old shell scripts together with duct tape, ponder no more. Let me tell you, Puppet isn't the only tool but it is a *fine* tool. This book will help you over the initial hurdles. You know the old "give a man a fish, teach a man to fish" chestnut? Some technical books err on the side of all theory, others give you pages of code and cli copy-paste but leave you without understanding what you're doing. The authors of this book struck a nice balance between the two extremes. There's just enough hand-holding to get you going and just enough theory to keep the book around for reference. They walk you through getting your initial management server (aka puppetmaster) up and running, and take you through some real-world scenarios managing various services on several different mainstream platforms. Then they move on to integration with source control, scalability, reporting, integration with third-party tools, and, finally, developing your own modules.

I did find a few gotcha mistakes in my review copy, particularly in the first couple of chapters, which are heavy on the cli copy-pastey bits. Nothing too hard to work around, though. Otherwise, if I had to make one criticism it would be that while there's a sizable base of third-party modules available (via the Puppet Forge Web site), the authors didn't spend much time on how to adapt these modules for your own use. Puppet comes with a good deal already built in, but most people are going to need external modules. The section on using a module from Puppet Forge was a bit weak at three pages; I think it could've been a stand-alone chapter.

To sum up, this is a fine introduction to Puppet. James Turnball's previous book on Puppet, 2008's *Pulling Strings with Puppet,* was badly in need of a rewrite. If you're already a hardcore Puppet user then this book probably won't be very interesting for you. But if you're interested in dropping the duct tape and shell scripts and graduating to a proper configuration management tool, buy this book and give Puppet a try.

—Reviewed by Trey Darley (trey@treyka.net)

Hadoop: The Definitive Guide, 2d Edition

Tom White

O'Reilly Media, Inc., 2010. 626 pp.

ISBN 978-1-4493-8973-4

I've been working with some large-data projects, and one of my co-workers suggested Hadoop. Being new to Hadoop, I jumped on O'Reilly's *Definitive Guide* and never really looked back.

To be honest, I've only read about half of the book so far, but it's become clear that (1) Hadoop is the right solution for my problem and (2) this is the right book for me to use. Chapter 1 walks you through the "hows" and "whys" of Hadoop, which introduces some of the problems of dealing with large data sets, as well as the inception and evolution of Hadoop to address those problems. Chapter 2 introduces you to MapReduce, a "programming model for data processing," which means that MapReduce is the workhorse of Hadoop—it is the mechanism you must write to take raw data and put it into Hadoop. Also in this chapter, a basic comparison between Hadoop and UNIX Tools (AWK) shows the scalability and power that Hadoop allows. In a nutshell, Hadoop handles scalability and redundancy, allowing the user/programmer to focus almost entirely on data issues (indexing, searching, etc.). A big part of the redundancy and scalability comes not from MapReduce, but from the Hadoop Distributed File System (HDFS), which is explained in Chapter 3. HDFS is designed to allow for large file storage (terabytes) and a transparent clustering system, which is the beauty of Hadoop. Increasing storage space simply means adding new systems to the cluster. HDFS separates this from the programmer, once again, allowing them to deal with the data and not worry too much about the underlying infrastructure. Chapter 4 extends this to explaining the mechanics behind Hadoop I/O.

Chapter 5 is a step-by-step walkthrough of building a MapReduce application, which is where everything starts to gel. You begin by building your Map and Reduce functions, and run them against a small subset of your data (as an aside, you do a very similar process in Chapter 2, but with sample data provided by the authors). Once you feel that everything is working as it should, you run your application on a cluster against the entire data set. Hopefully, there's more tuning and less troubleshooting at this point, since it can be difficult to identify bugs when dealing with tons of data across the cluster. Chapter 6 goes even deeper, by explaining how MapReduce works at a very low level. This provides for better tuning and more advanced MapReduce functions. Chapters 7 and 8 continue by explaining "MapReduce Types and Formats" and "MapReduce Features," respectively. I haven't spent much time on these chapters, but just skimming through them I can see that there is a lot to learn—and a pretty big difference between setting up a working Hadoop system vs. a finely tuned (and well-programmed) environment.

Chapters 9 and 10 show you how to actually set up a Hadoop Cluster and administer it. The next five chapters deal with various tools that have evolved to make using Hadoop easier:

Pig, Hive, HBase, ZooKeeper, and Sqoop. I've spent some time with Hive, but haven't yet dug into the others. Chapter 16 outlines seven different case studies, including Last.fm, Facebook, and Rackspace. It is truly amazing the amount of data we live with in today's Internet, and Hadoop is a very powerful, cost-effective (free) and useful tool for dealing with it. There are a couple of other Hadoop books out there, but of the ones I perused, this one seems like the right fit. It's well written, very technical, but not intimidating. If you don't work with Hadoop, you probably will, and this is the book to grab when it happens.

—Reviewed by Sam Stover

Hacking Exposed™ Wireless: Wireless Security Secrets & Solutions, Second Edition

Johnny Cache, Joshua Wright, and Vincent Liu

McGraw-Hill, 2010. 512 pp.

ISBN 978-0-07-166661-9

This book is a solid reference for wireless protocols, mechanisms, tools, and techniques. Some of the notable additions from the first edition include Zigbee (yay!) and new tools. I originally picked this book up to help with some Zigbee work I was doing, but ended up skipping around the whole book. There is a fair bit of basic info that you'll find in any wireless book, such as finding 802.11 networks and WEP cracking, so this can serve as a good introduction for beginners as well. For the OS X crowd, there is a decent amount of effort and time spent on explaining and introducing OS X tools and methods. Linux, of course, is also featured throughout.

Three sections divide the book: Hacking 802.11 Wireless Technology, Hacking 802.11 Clients, and Hacking Additional Wireless Technologies. As I mentioned earlier, the third section was the one that most interested me, and if you want to mess around with BlueTooth, Zigbee, and DECT, this is the go-to book for you. If you already have a solid grasp of other 802.11 technologies, you may feel that it's not worth the cost just for the additional wireless technologies, but I was glad to finally have a resource that gives a real introduction to Zigbee hacking, especially with the introduction of Killer-Bee, which is a "Python-based framework for manipulating Zigbee and IEEE 802.15.4 networks."

OK, enough about Zigbee, let me talk a bit about the rest of the book. As is typical with Hacking Exposed books, there are a ton of example scenarios which deal with realistic scenarios, which should be very helpful to the budding wireless pen-tester. Part 1 should be nothing new to the seasoned wireless expert, but lays a solid groundwork and gives updated information on the techniques and tools used. The step-by-step instructions and explanations should make it easy for just about anyone to follow along and learn by doing, which is my favorite way to do it.

Part 2 was a pretty pleasant surprise for me. There are a number of useful client-side tools out there that I wasn't aware of (IPPON, Ferret, and Hamster) and some old-and-still-good tools like Metasploit and Ettercap. As with Part 1, everything was well explained and easy to follow. I especially liked Chapter 6, which walks you through the entire process in OS X.

I've already waxed enthusiastic about Part 3, so I'll spare you more. Overall, this was a solid book with great examples, good overall 802.11 reference material, and enough new stuff to justify springing for the second edition. In fact, I'm anxiously awaiting the third edition to see what they add to the Zigbee/DECT sections.

—Reviewed by Sam Stover

DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X and FreeBSD (Oracle Solaris Series)

Brendan Gregg and Jim Mauro

Prentice Hall, 2011. 1152 pp.

ISBN 978-0132091510

This is the book you need if you are trying to understand performance, and debug performance problems, on a system that contains DTrace system analytics infrastructure. It also includes useful performance analysis methods, questions, and logical exploration that could help a junior or mid-level systems administrator or programmer learn about performance analysis, but of course the scripts would not be of much use.

This review won't spend space waxing eloquent about DTrace, as that has been done before, many times (including many publications by USENIX, as the Google search "site:www.usenix.org dtrace" reveals). I'll just summarize by saying that DTrace is the most important modern-era computing tool for understanding and debugging system behavior and performance.

DTrace itself is mind-bogglingly complex. It includes a new idea, implemented by kernel structures and the new D language. Before this book there were many sources of DTrace information, including the Solaris manual, tutorials, talks, toolkits, and forums. And before this book were other books, such as *Solaris Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris* by Richard McDougall, Jim Mauro and Brendan Gregg. That book concentrated on Solaris performance while exploring DTrace as one of the useful tools.

This book is what people who are interested in DTrace—and even people who are experienced with DTrace—have been waiting for. Those DTrace knowledge seekers now have, in one volume, information on what DTrace is, how to use DTrace, when to use DTrace, and lots of new, useful, informative scripts that can be typed in (or downloaded from the book's Web site) and executed to analyze a system. It also includes, as needed, information from those other sources, such as scripts from the DTraceToolkit. *DTrace* the book builds up knowledge of DTrace the facility from scratch, and quickly, to the point where a reader is able to write useful DTrace scripts and achieve a deep knowledge of system performance analysis.

Because DTrace was part of OpenSolaris and therefore had its source code released, and because it's so powerful, it has been ported to other operating systems, including FreeBSD and Mac OS X. Unfortunately, those ports are not quite as rich as the OpenSolaris implementation, so some information in the book does not apply to them and some scripts don't work on them. The book does a good job of pointing out these limits. For example, there is no tcp provider in Mac OS X, so the scripts in that section, including "tcpconnect," which shows TCP connections as they occur, will not run on Mac OS.

Rather than reading this book, you could start from the very good manual that comes with Solaris. However, that is daunting, complete and complex, and mostly, cleverly avoids showing "how to use DTrace to do useful stuff." Cleverly, because it's a powerful tool, and the authors don't want to limit the audience to using the tool only in the ways they have thought of. This is much like Sawzall showing exactly how to use all the features of its tool but not showing how to use it to cut a hole in a wall. Perhaps that would keep the users from realizing they could also cut holes in the roof, floor, and so on.

Part 1 provides a succinct summary of the language and the other DTrace components, and the remainder of the book shows how to use DTrace to examine various aspects of user and kernel mode operations, how to solve performance issues, and how to diagnose problems. It explores the included scripts line by line and character by character, teaching by example and stressing the learn-by-doing approach. In fact that's how the authors learned DTrace—in the field and within Sun/Oracle, solving customer's problems and writing scripts to make DTrace even more useful and efficient.

The authors are maintaining a Web site from which the scripts can be downloaded and where other information, such as the errata, is likely to be posted over time, at http://dtracebook.com/index.php/Main_Page. The book is also available electronically via Safari and on the Kindle. Either way makes the text available on a computer, which is great for searching as well as for copy-and-paste actions.

In short, these are the kernel innards and performance analysis details you are looking for. The book is a masterpiece of hands-on system performance analysis methodologies and tools. If you don't have Jim Mauro's cell phone number, this book is the next best thing. (Fair notice, Jim is a friend and it *is* nice to have his cell phone number.)

—Reviewed by Peter Baer Galvin

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 reports 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 USENIX Board of Directors

The biennial election of the USENIX Board of Directors will be held in early 2012. The USENIX Board has appointed Alva Couch 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 will be published on the USENIX Web site this fall.

USENIX Association Financial Statements for 2010

The following information is provided as the annual report of the USENIX Association's finances. The accompanying statements (p. 60) have been reviewed by Michelle Suski, CPA, in accordance with Statements on Standards for Accounting and Review Services issued by the American Institute of Certified Public Accountants. The 2010 financial statements were also audited by McSweeney & Associates, CPAs. Accompanying the statements are several charts that illustrate where your membership dues go. The Association's complete financial statements for the fiscal year ended December 31, 2010, are available on request.

—Ellie Young, Executive Director

Chart 1: USENIX 2010 Membership Dues Revenue

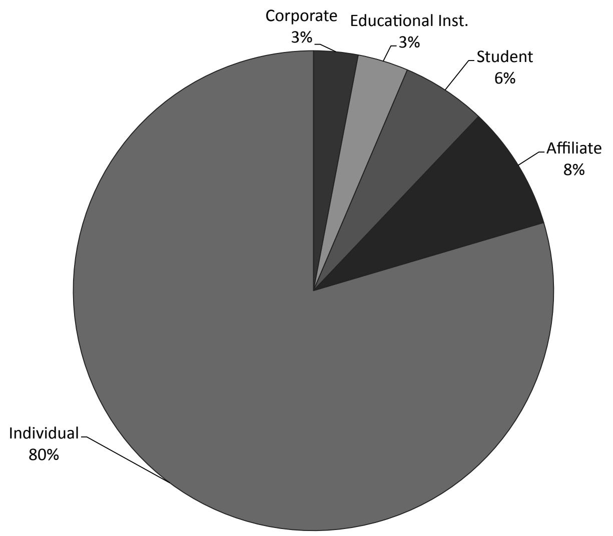


Chart 2: Where Your 2010 Membership Dues Went

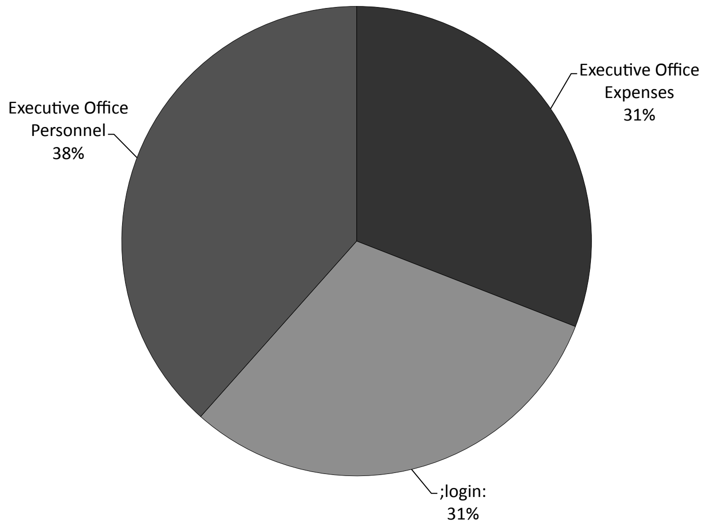
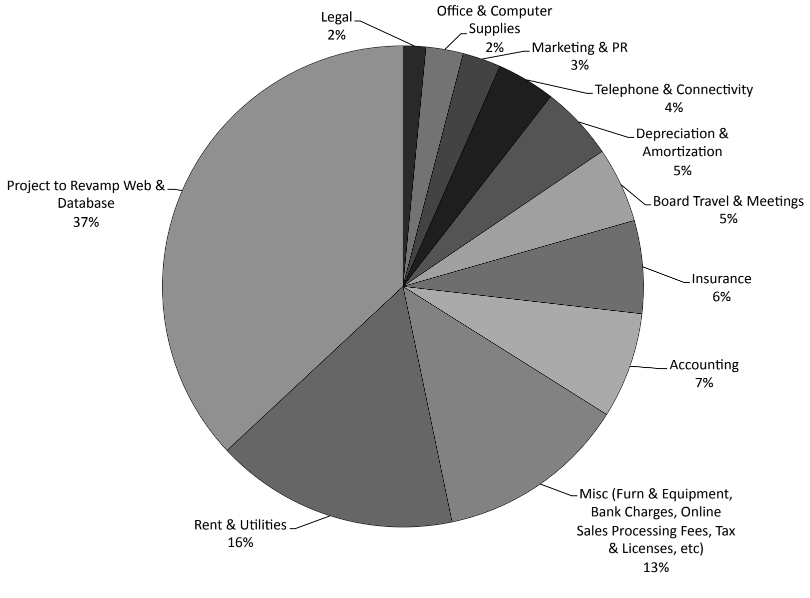
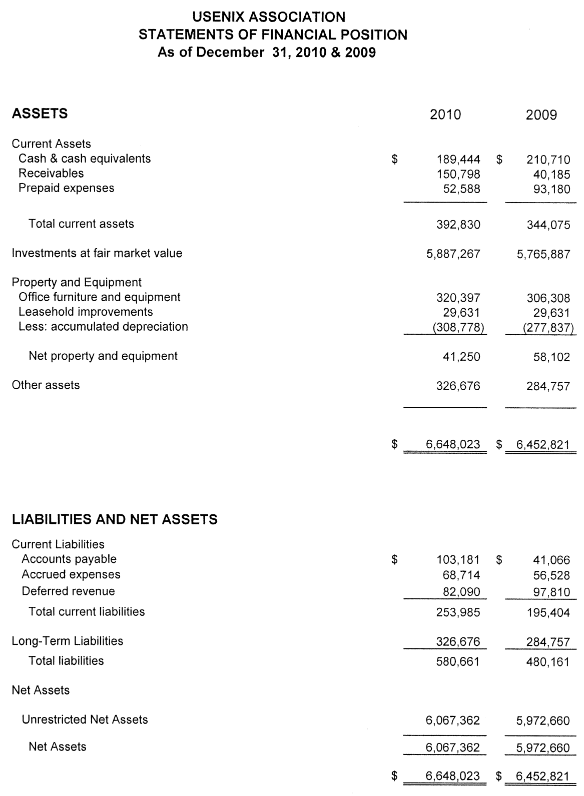


Chart 3: USENIX 2010 Administrative Expenses

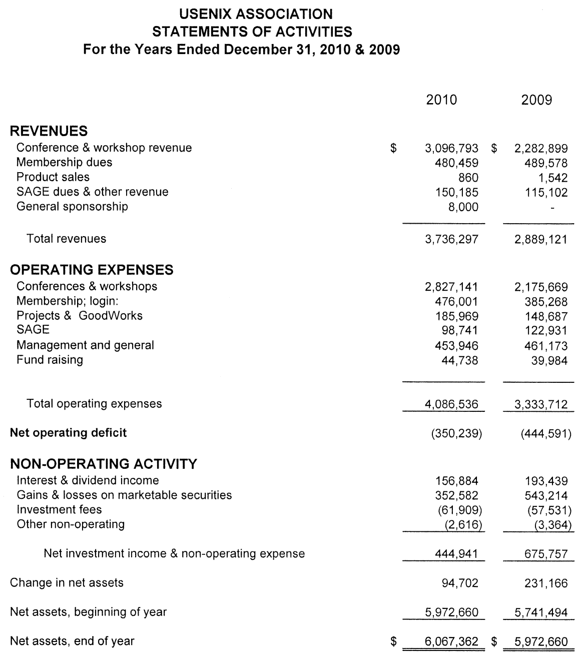




USENIX ASSOCIATION
STATEMENTS OF FINANCIAL POSITION
As of December 31, 2010 & 2009

ASSETS	2010	2009
Current Assets		
Cash & cash equivalents	$ 189,444	$ 210,710
Receivables	150,798	40,185
Prepaid expenses	52,588	93,180
Total current assets	392,830	344,075
Investments at fair market value	5,887,267	5,765,887
Property and Equipment		
Office furniture and equipment	320,397	306,308
Leasehold improvements	29,631	29,631
Less: accumulated depreciation	(308,778)	(277,837)
Net property and equipment	41,250	58,102
Other assets	326,676	284,757
	$ 6,648,023	$ 6,452,821

LIABILITIES AND NET ASSETS		
Current Liabilities		
Accounts payable	$ 103,181	$ 41,066
Accrued expenses	68,714	56,528
Deferred revenue	82,090	97,810
Total current liabilities	253,985	195,404
Long-Term Liabilities	326,676	284,757
Total liabilities	580,661	480,161
Net Assets		
Unrestricted Net Assets	6,067,362	5,972,660
Net Assets	6,067,362	5,972,660
	$ 6,648,023	$ 6,452,821



USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2010 & 2009

	2010	2009
REVENUES		
Conference & workshop revenue	$ 3,096,793	$ 2,282,899
Membership dues	480,459	489,578
Product sales	860	1,542
SAGE dues & other revenue	150,185	115,102
General sponsorship	8,000	-
Total revenues	3,736,297	2,889,121
OPERATING EXPENSES		
Conferences & workshops	2,827,141	2,175,669
Membership; login:	476,001	385,268
Projects & GoodWorks	185,969	148,687
SAGE	98,741	122,931
Management and general	453,946	461,173
Fund raising	44,738	39,984
Total operating expenses	4,086,536	3,333,712
Net operating deficit	(350,239)	(444,591)
NON-OPERATING ACTIVITY		
Interest & dividend income	156,884	193,439
Gains & losses on marketable securities	352,582	543,214
Investment fees	(61,909)	(57,531)
Other non-operating	(2,616)	(3,364)
Net investment income & non-operating expense	444,941	675,757
Change in net assets	94,702	231,166
Net assets, beginning of year	5,972,660	5,741,494
Net assets, end of year	$ 6,067,362	$ 5,972,660

Conference Reports

In this issue:

2011 USENIX Annual Technical Conference (USENIX ATC '11)

Portland, Oregon
June 15–17, 2011

Welcome, Awards, and Keynote Address

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

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

Summarized by Rik Farrow (rik@usenix.org)

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

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

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

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

Keynote Address: An Agenda for Empirical Cyber Crime Research

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

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

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

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

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

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

Scheduling

Summarized by Timothy Merrifield (tmerri4@uic.edu)

A Case for NUMA-aware Contention Management on Multicore Systems

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

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

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

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

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

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

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

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

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

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

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

Pegasus: Coordinated Scheduling for Virtualized Accelerator-based Systems

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

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

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

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

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

Virtualization

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

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO

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

Irfan started the session right after lunch on a light note by presenting the history of "Interrupt Coalescing," starting from the work of Djikstra to the patents filed so far and the implementations seen. He briefly described existing techniques and coined the abbreviations MIDL (maximum interrupt delay latency) and MCC (maximum coalesce count) to talk about past work on network interrupt coalescing done in the 1970s. The same techniques would also work for storage devices. However, virtualization has introduced a layer between the real hardware and the virtual machines. The existing techniques based on high-resolution timers become less practical for virtual devices, due to their large overheads. So they looked at the number of commands in flight (CIF) to set the delay timer instead of using a high-resolution timer.

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

Raju Rangaswami (Florida International) asked whether they ran into some corner case where they really needed a timer (e.g., 5 sec. SSD delay) when there were a lot of writes in flight. Irfan said there are opportunities in the hypervisor to inspect if interrupts have been delivered and there are optimizations. The current architecture gives them opportunities to inspect the completion queue whenever a VM exits into the VMM. All these things add up to a tight bound on maximum latency, but they can adjust. Raju asked whether they need to worry about the OS doing something it shouldn't because the time of interrupt delivery gets changed. Irfan replied that there is pathological behavior, especially in certain old operating systems, if an I/O device never returns. So people have looked at longer periods of keep-alive mode. However, all those behaviors have a range in seconds, whereas with vIC, interrupts never sit in the queue for so long and get delivered at the latest with the host timer. Alexandra Fedorova (SFU) asked about the effect of the technique on performance variability of certain workloads and end-to-end performance. Irfan acknowledged that there is some inherent variability in I/O completion times, but they haven't done precise studies to see if that variability is increased due to this technique. However, their experience with I/O path variability indicates that overall variability just becomes more structured. In fact it could be possible to lower variability by slowing down fast I/O in the same way that it happens for hardware variability at times. The last questioner noted that their results were conflicting, since interrupts on current machines take less than microsecs; however, their coalescing period is high, especially as seen for TPC-C. Irfan responded that in virtualization, interrupts cause VM exits, cache pollution, etc. TPC-C is sensitive to cache and just by improving that, they see benefits.

Power Budgeting for Virtualized Data Centers

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

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

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

VPS has multiple dynamic controllers: (1) a top-level shared infrastructure feedback PID controller that adapts to dynamic workload and power budget; (2) weighted fair sharing for individual applications, using priorities based on the total application budget for the next time period; (3) an application-level PID controller to distribute power for each tier as a cascading effect of controlling the power knob for the first tier; and (4) tier-level controllers. There are three tier-level controller options: open loop control (easy and instantaneous but no error compensation), PID control (slower, single control knob but compensates for error), and model predictive control (optimizes performance with multiple knobs but requires performance measurement and systems models that relate control knobs to system states). They evaluated tradeoffs for these techniques against accuracy, speed, amount of application information, and models required. Harold concluded his talk with some performance analysis and showing how VPS gives low budgeting errors compared to physical hierarchy controllers.

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

vIOMMU: Efficient IOMMU Emulation

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

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

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

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

Cloud Computing and Data Centers

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

HiTune: Dataflow-Based Performance Analysis for Big Data Cloud

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

In the first talk of this joint USENIX ATC/HotCloud session, Jinquan Dai presented HiTune, a performance analysis tool for "big data" clouds. Jinquan started by explaining the dataflow model for big data analytics, where applications are modeled as dataflow graphs, the user writes subroutines that operate on the vertices, and the runtime takes care of all of the messy low-level details. This model is implemented in the popular Hadoop system. Although this is a useful model for parallel programming, the complexity of the system means that it often appears to the user as a black box, and in particular, it can be very difficult for the user to understand, analyze, and tune runtime performance.

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

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

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

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

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

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

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

In-situ MapReduce for Log Processing

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

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

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

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

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

Joint ATC and WebApps Invited Talk

Helping Humanity with Phones and Clouds

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

Summarized by Timothy Merrifield (tmerri4@uic.edu)

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

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

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

Joint ATC and WebApps Poster Session

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

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

Evaluating the Effectiveness of Model-Based Power Characterization

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

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

OFRewind: Enabling Record and Replay Troubleshooting for Networks

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

Network troubleshooting is challenging for many reasons: high traffic volumes hamper pre-emptive recording for later analysis, pervasive black box devices prevent instrumentation, and distributed protocols may exhibit hard-to-reproduce faults. This work introduces the primitive of replay debugging to networks, implemented as OFRewind, a software tool for recording and replaying traffic with the help of OpenFlow. OFRewind allows network operators to centrally orchestrate the scalable recording of arbitrary flow-level granularity traffic in an OpenFlow network, maintaining a complete ordering of all flows. This traffic can be recorded in always-on fashion, and replayed in situ or in a lab to reproduce many different types of errors and localize problems.

BenchLab: An Open Testbed for Realistic Benchmarking of Web Applications

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

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

Making GPUs First Class Citizens

Vishakha Gupta and Karsten Schwan, Georgia Institute of Technology

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

Toward Online Testing of Federated and Heterogeneous Distributed Systems

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

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

Joint ATC and WebApps Plenary Session

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

Finn Brunton, Postdoctoral Researcher at NYU

Summarized by Muhammad Shahzad (shahzadm@msu.edu)

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

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

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

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

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

Measurement and Performance

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

Exception-Less System Calls for Event-Driven Servers

Livio Soares and Michael Stumm, University of Toronto

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

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

Abel Gordon (IBM Research) asked if they had evaluated performance when kernel threads were run on different sockets. Livio replied that they hadn't performed actual experiments but he would try to keep syscall threads on the same locality domain because of shared syscall entries that will be communicating back and forth. With more sockets the execution might have to be split differently. Does there need to be some kind of affinity with respect to interrupts? Livio said interrupts were being sent to the cores where syscall threads were actively being processed. Their system tries to increase or decrease the number of syscall cores based on application behavior. They change the APIC controller to go to those cores, so there is some mapping between cores that issue interrupts and those that handle them. Peter Desnoyers (Northeastern) asked if they saw any parallel with the interrupt coalescing work in the form of their deferred work queue. Livio said this work derived from the soft timers work, but his work is more from the application down. Michael Dexter asked about the security risks of using libflexsc. Livio replied that the fundamental property they relied on was MMU; if the OS set up the page tables correctly, only the corresponding process should get access. Besides, there were easier ways to compromise the OS. Someone asked about the complexity of using events in large programs. Livio said he did not have much experience with that, but that they would essentially look like large distributed systems.

Resizable, Scalable, Concurrent Hash Tables via Relativistic Programming

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

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

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

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

Evaluating the Effectiveness of Model-Based Power Characterization

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

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

He used a large set of common benchmarks like SPEC CPU 2006, PARSEC, Linux build, and so on to demonstrate how the power models did progressively worse on newer platforms, especially multicores, when compared to their accurate power measurement installation on the Calpella platform. He pointed out that the overall system power was still within tolerable limits but the error grew worse when characterizing individual components. He concluded that: (1) modern systems have lots of hidden state for multiple reasons (sometimes because manufacturers don't want to reveal them); (2) increasing system complexity with some interactions is hard to capture—for example, HDD power modeling error is half that of SSD; and (3) the problem becomes worse if you add hardware variability to the mix—even identical parts are not necessarily identical. He ended by emphasizing the need for low-cost instrumentation solutions for accurate power characterization, which will be quite essential in future systems.

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

Storage Performance and Migration

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

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

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

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

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

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

The Design and Evolution of Live Storage Migration in VMware ESX

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

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

The evolution of storage migration begins with a simple approach: making the VMDK base disk image read-only, sending writes to the snapshot, migrating the VMDK, reparenting to the destination volume, and migrating the snapshot. Since this approach has a number of disadvantages, including downtime, long migration times, and requiring disk space up to 3x the VM size, they tried something different: A dirty block tracking (DBT) filter marks dirty blocks. The migration involves copying changes and repeating until the set of changes is small enough, then stopping the VM, copying the final changes, and cutting over. This still had limitations such as questions about convergence (e.g., will it complete under workloads where blocks are dirtied very quickly?). The most recent insight was that storage is not memory: interposing on all writes is practical. I/O mirroring works by synchronously mirroring all writes and making a single pass to transfer the bulk data. This approach has effectively no downtime, low performance penalty, and atomicity. Throttling the source can be used when I/O mirroring to a slow destination. Future work involves leveraging workload analysis (heat maps) for I/O mirroring, sequential write pattern detection, lazy mirroring, and WAN migrations..

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

Online Migration for Geo-distributed Storage Systems

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

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

Tran proposed a new approach called distributed data overlays. The intuition behind this approach is read first at the new site, and redirect if the data is not there. This technique allows data to be accessible at all times, and the migration benefits (e.g., locality to user) are realized quickly. When an overlay is created, a layer is added to the top of the stack. Reads go from the top of the stack down, until valid data is reached. Writes are written only to the top layer. Migration between overlays involves copying data from the source to destination, while reads and writes are going on concurrently. Tran's group developed Nomad, an object storage system using overlays. The distributed data overlays performed better than traditional lock-and-log approaches. Write latencies are instantly lowered, read latencies get better gradually throughout the migration, and there is no point at which the data is inaccessible.

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

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

Joint WebApps and ATC Invited Talk

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

Software G Forces: The Effects of Acceleration

Kent Beck, Facebook, Inc.

Kent Beck's keynote was undoubtedly the liveliest in USENIX ATC '11. Kent started by reviewing the software development situation today and in the recent past. The software development cycle in the 1990s was mostly yearly, and only a few specific software products, such as bond traders' software, had a deployment cycle that was shorter than a quarter. Today the deployment cycle is considerably shorter, mainly quarterly, and crazy fringe things start to happen as some software products are deployed on a daily basis. Kent believes this trend will continue, and in the 2030s most of the software development cycles will be shorter than a month. Kent provoked the audience by asking, "What happens if 90% of the software will be deployed on an hourly basis in 20 years?"

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

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

Short Papers

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

Slow Down or Sleep, That Is the Question

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

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

Low Cost Working Set Size Tracking

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

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

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

Chunqiang Tang, IBM T.J. Watson Research Center

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

Okeanos: Wasteless Journaling for Fast and Reliable Multistream Storage

Andromachi Hatzieleftheriou and Stergios V. Anastasiadis, University of Ioannina

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

Toward Online Testing of Federated and Heterogeneous Distributed Systems

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

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

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

Philip J. Guo and Dawson Engler, Stanford University

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

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

Vsys: A Programmable sudo

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

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

Internet-scale Visualization and Detection of Performance Events

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

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

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

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

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

Storage Deduplication

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

Building a High-performance Deduplication System

Fanglu Guo and Petros Efstathopoulos, Symantec Research Labs

Awarded Best Paper!

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

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

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

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

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

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

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

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

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

Debugging and Diagnosis

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

G^2: A Graph Processing System for Diagnosing Distributed Systems

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

Haoxiang Lin began by noting that bugs are often easily noticed: a system malfunctions and a user complains. Although the final fixes may seem simple, finding the root causes of such bugs, especially in highly distributed systems, is a complex and painful process. Often, distributed systems are executing on many machines, detailed logs are recorded on each machine, and correlations between logs are not well preserved. This makes using those logs to diagnose problems difficult. G^2, a graph-based system for diagnosing bugs, uses an execution flow graph model to trace the root causes of malfunctions in a distributed system.

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

Slicing and hierarchical aggregation are implemented as traversing the execution graphs. G^2 builds a distributed graph engine and applies several special optimizations such as batched asynchronous graph traversal, partition-level interface, caching, and prefetching, which makes traversal on huge graphs very efficient. Haoxiang presented an evaluation using the SCOPE/Dryad distributed programming system.

Someone asked what types of bugs G^2 worked well with and what types it did not. Haoxiang responded that G^2 is useful for bugs having complex internal interactions, which are difficult for “simple” tools to detect. Using G^2, they can perform several runs of slicing and aggregation to find correlations and, hence, the root causes of such complicated bugs.

Context-based Online Configuration-Error Detection

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

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

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

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

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

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

OFRewind: Enabling Record and Replay Troubleshooting for Networks

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

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

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

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

ORDER: Object centRic DEterministic Replay for Java

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

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

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

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

Security and Privacy

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

Enabling Security in Cloud Storage SLAs with CloudProof

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

Raluca Ada Popa presented CloudProof's model. An owner stores his/her data on a cloud (that could be compromised in any way) and clients access this data based on their permissions. CloudProof considers four security properties (defined in the paper): confidentiality, integrity, write-serializability, and freshness. CloudProof not only enables clients to detect whether the cloud violated integrity, write-serializability, or freshness, but, importantly, enables clients to prove these violations to any external party. This proof-based system is critical to enabling security guarantees in SLAs, wherein clients pay for a desired level of security and are assured they will receive some compensation in the event of cloud misbehavior. CloudProof's design uses a mechanism called attestations, which are pieces of data exchanged by the cloud and the clients for every client request; clients verify these attestations to detect integrity violations, and the owner audits them probabilistically to detect violations of write-serializability and freshness. When a violation is detected, they can construct a proof using the attestations. CloudProof aims to scale to large enterprises, so it provides a scalable access control scheme based on broadcast encryption and block families. The evaluation on real traces shows that the security mechanisms have a reasonable cost and that the owner's workload is lightweight.

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

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

Carsten Weinhold and Hermann Härtig, Technische Universität Dresden

Carsten Weinhold gave the talk on jVPFS, a system for protecting confidentiality and integrity of application data. jVPFS improves on its predecessor VPFS (Virtual Private File System) by minimizing threats related to unclean shutdown of the system. To protect integrity and consistency, a Merkle hash tree spans the file system, and to log updates to the file system efficiently, jVPFS uses a journaling scheme. The journal itself is protected by continuously hashing all appended records, each hash being a hash of the entire previous history. jVPFS includes a novel cooperation scheme in which trusted and untrusted components cooperate and some metadata information is revealed to untrusted code to facilitate such cooperation.

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

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

Distributed Systems

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

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

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

Awarded Best Paper!

Ashish Chawla presented SPOCA, a new content-addressing algorithm for Yahoo!'s video platform. The platform serves 20 million video assets to users, who make 30 million requests per day from all continents. An early off-the-shelf solution had the problem that front-end servers in the various data centers handling user requests could not optimally cache video files in memory and on disk. For example, a user who accesses an unpopular video cached on a server on another continent may observe sub-standard playback performance. The platform was also prone to redundant caching (rarely accessed files consume cache space on many servers), and improving the cache promotion algorithm alone was not sufficient. The soft-state-only SPOCA and Zebra algorithms that are deployed at Yahoo! at large scale solve these issues.

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

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

TidyFS: A Simple and Small Distributed File System

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

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

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

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

Personal Devices

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

Eyo: Device-Transparent Personal Storage

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

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

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

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

Autonomous Storage Management for Personal Devices with PodBase

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

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

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

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

2nd USENIX Conference on Web Application Development (WebApps '11)

Portland, OR
June 15–16, 2011

Joint ATC, WebApps, and HotCloud Keynote Address

An Agenda for Empirical Cyber Crime Research

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

See the 2011 USENIX Annual Technical Conference report for this session.

Server-side Security

Summarized by Ioannis Papagiannis (ip108@doc.ic.ac.uk)

GuardRails: A Data-Centric Web Application Security Framework

Jonathan Burket, Patrick Mutchler, Michael Weaver, Muzzammil Zaveri, and David Evans, University of Virginia

Web frameworks significantly facilitate the creation of Web applications. However, they do little to facilitate the development of applications that are secure by design. In reality, lots of applications suffer from known, persistent types of vulnerabilities. Popular examples are cross-site scripting, SQL injection, and data disclosure vulnerabilities. For Rails Web applications, the root cause of such vulnerabilities is that developers have to consistently attach checks to controllers every time controllers manipulate user data. When developers forget these checks, they introduce vulnerabilities. The key idea of GuardRails is to associate data dissemination policies with the data that they protect and have the framework enforce the policies automatically.

GuardRails does source-to-source code transformations of Rails applications after they have been annotated with policies. There are two types of policies that GuardRails can enforce: access control policies and taint tracking policies. Policies can contain arbitrary code, and GuardRails supports a special syntax to define them. There are various default policies that can be used by application developers with minimal effort. For taint tracking policies, GuardRails can associate and track multiple policies per string, supporting up to character-level tainting. In order to support fine-grained sanitization according to the precise HTML context where data are used, GuardRails invokes the sanitization operations after the HTTP response has been fully generated. The authors tested GuardRails with existing vulnerable Rails applications of different sizes. Jonathan Burket reported that GuardRails prevented all vulnerabilities they tested it with. However, in terms of performance, GuardRails reduced the sustainable rate of transactions per second down to one-fourth of the rate of the original applications. Most of the overhead can be attributed to character-level taint tracking. They plan to improve performance by implementing most taint tracking routines inside the Ruby interpreter.

The Q&A mainly involved questions about how existing applications are affected by GuardRails. Jonathan responded that if the policies are written correctly, then GuardRails can avoid repeating access control operations and redundant sanitization. He also mentioned that the current prototype of GuardRails is not optimized for performance and, therefore, he sees a lot of room for improvement there.

PHP Aspis: Using Partial Taint Tracking to Protect Against Injection Attacks

Ioannis Papagiannis, Matteo Migliavacca, and Peter Pietzuch, Imperial College London

Modern Web applications rely on sanitization functions to avoid injection attacks. A sanitization function transforms user-provided strings so that the user cannot change the semantics of the operations that the Web application invokes. Developers often forget to call these functions and, by doing so, they introduce injection vulnerabilities. Past research has shown that taint tracking is effective in preventing such vulnerabilities: it can invoke sanitization functions automatically and use the taint information to sanitize the exact string characters that originate from the user. However, it is not supported by PHP.

Ioannis Papagiannis introduced PHP Aspis, a taint tracking system for existing PHP applications. PHP Aspis does taint tracking at the source-code level, by transforming the application scripts to propagate taint explicitly. This does not require support from the official interpreter or the maintenance of a custom interpreter. However, the transformations replace efficient low-level PHP operations with more computationally expensive counterparts (e.g., the concatenation operation is replaced by a function call that also propagates taint), and this reduces performance. To improve performance, the authors suggest partial taint tracking, i.e., to track taint only in the most vulnerable parts of Web applications. Ioannis supported their approach using the Wordpress example, where most past injection vulnerabilities involved plugins and not the Wordpress core. PHP Aspis separates application code in tracking and non-tracking code: the former is protected from injection attacks, but the latter is not.

In the Q&A, people wondered how PHP Aspis handles the dynamic features of PHP. Ioannis responded that PHP Aspis is also invoked at runtime to process dynamically generated code and inspect dynamic function calls. Another question concerned the separation between tracking and non-tracking code. Ioannis clarified that this separation is decided by the application's administrator and that the criteria for the separation may vary according to where injection vulnerabilities are expected to occur.

Secure Data Preservers for Web Services

Jayanthkumar Kannan, Google Inc.; Petros Maniatis, Intel Labs; Byung-Gon Chun, Yahoo! Research

Users trust services with large quantities of their sensitive data, a situation that can result in large-scale data leaks if the service gets attacked. Byung-Gon Chun suggested that the root cause of the problem is that users provide to centralized services complete access over their data. As a result, users cannot control how these services use the data. At a high level, this violates the least-privilege principle. In their paper the authors suggest the concept of Preservers, proxy objects that encapsulate user data and expose a secure API for data access. The approach targets applications that only access user data via a well-defined interface, rather than via direct raw access, but they claim that this is the norm for many real-world Web services. The client, instead of releasing his data, sets up a custom Preserver that connects to the service and implements arbitrary access control policies. Preservers may process the data they enclose and may also filter them to remove information that can identify the user. There are both stateless and stateful versions of Preservers.

Preservers may execute either in a third-party server that is trusted by both parties or be co-located with the user or the service. This flexibility avoids limitations placed by dominant service providers and enables the user to select an appropriate placement strategy according to his performance and security requirements: a trusted third party may offer better security, but co-location with the service offers reduced latency. To achieve secure co-location of Preservers with Web services, the authors suggest a Preserver implementation that relies on separate VMs for isolation. For the evaluation, the authors used three representative applications (day-trading, targeted advertising, and secure payments) and did micro-benchmarks to measure the latency overhead that the different Preserver placement options introduce. Their results show that a trusted third-party placement is the most secure but results in an order of magnitude higher latency compared to either client-side or server-side co-location.

In the Q&A, people worried about the latency of the third-party placement strategy. Byung-Gon suggested that latency can be improved by reducing the physical distance to the Preserver's host. He also said that their approach enables smaller Web sites to access user data that users would not trust otherwise. Another open question is how to find a proper interface for the Preserver, as this is not always straightforward.

Researchers' Workbench

Summarized by Veena Udayabhanu (veena@cs.umass.edu)

BenchLab: An Open Testbed for Realistic Benchmarking of Web Applications

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

Web applications have evolved from serving just static content to dynamically generating Web pages. Modern Web 2.0 applications include JavaScript and AJAX technologies that manage complex interactions between the client and the server. Currently, widely used benchmarks such as TPC-W and RUBiS rely on browser emulators that only mimic basic network functionality but cannot emulate other complex interactions that today's browsers possess, such as JavaScript processing. They use static load distribution. Also, the fact that most benchmarking experiments are only carried out in a LAN environment poses several questions about the accuracy of the results obtained, because the real latencies seen by geographically distributed users are not taken into account.

All these facts scream that Web applications have evolved but benchmarks have not! There are various factors that impact server performance, such as typing speed and the state size on the server. Emmanuel Cecchet presented BenchLab, an open testbed designed to address these issues.

BenchLab captures real-application workload, which is then replayed using real Web browsers, and detailed results are stored in the benchmark repository. The benchmark repository also can store virtual machines of applications under test, test traces, configurations, and results. This is useful in repeating experiments and comparing results. Capturing real traces can be done at the browser, proxy, or httpd level, depending on one's needs. Separating the generation and injection of workload is a key concept of BenchLab, and the BenchLab Webapp itself is a JEE Web application with an embedded database and repository.

In terms of the results obtained, we have seen the difference between using emulators and real browsers on server utilization. We have also seen the effects of JavaScript processing on the server workload. Finally, we have compared the effects of LAN versus WAN load injection.

One audience member asked how we deal with non-deterministic behavior of personalized Web pages such as someone's profile page on Facebook. We handle it by using HTML comparison. The repository can also contain the complete HTML responses generated by the server and we can use this to compare results between successive experiments. Another asked how they distinguish between real URLs and ones automatically generated by JavaScripts and style sheets. It is done by using the referrer field in the httpd logs and some intelligent processing.

Resource Provisioning of Web Applications in Heterogeneous Clouds

Jiang Dejun, VU University Amsterdam and Tsinghua University Beijing; Guillaume Pierre, VU University Amsterdam; Chi-Hung Chi, Tsinghua University Beijing

Emmanuel Cecchet presented this paper on behalf of Jiang Dejun, because Jiang had problems getting his US visa. In this work, the authors said that provisioning Web applications in a cloud that consists of heterogeneous machines is tough. The performance of these applications is the main concern. When the same program was run on 30 small instances of Amazon EC2 servers, they saw different performances on each of the systems.

The authors say there are two simple solutions to this problem: ignore the heterogeneous resource factors and apply current resource provisioning to make a decision, or profile each of the VM instances at each tier to make a decision (extremely time-consuming). Instead, they propose a novel resource provisioning technique consisting of five steps: (1) use a reference application for calibration; (2) correlate resource demands of reference applications and tier services on the calibration instance; (3) profile new instances with the referenced application; (4) check the performance on the new instance; and (5) apply "what-if" technique to predict the performance when a new instance is added. They showed the evaluation of their technique using TPC-W on EC2 and compared the results they got against standard techniques such as homogeneous provisioning. They concluded that profiling new instances with reference applications can be used to provision a Web application on a heterogeneous cloud platform.

In the ensuing discussion (rather than Q&A, since the authors weren't around to answer questions) the one thing most people agreed on was that checking which tier is the bottleneck is a better technique than just seeing the application performance or profiling VM instances.

C3: An Experimental, Extensible, Reconfigurable Platform for HTML-based Applications

Benjamin S. Lerner and Brian Burg, University of Washington; Herman Venter and Wolfram Schulte, Microsoft Research

Ben Lerner said that the focus of their research was on the client side of Web applications and then described what the client side of a Webapp behaves like. He mentioned the factors that make Web applications so "Webby" as follows: the code is always up-to-date; the code mainly comprises HTML, CSS, and a few JavaScripts; there is an option to view the source; and it has remixability, the ability to be combined, extended, and customized. He then explained that browser extensions are basically pieces of code that are written in order to customize browsers dynamically at runtime; these are required in order to experience new features as closed systems but are not sufficient for researchers who want to try new things.

The thinning barrier between Web applications and browsers and the slowly evaporating role of browsers prompted the development of the C3 framework. The goal of C3, which is a reconfigurable platform for HTML-based applications, is to make experiments with extensions that facilitate research. They followed a bottom-up approach to building C3, comprising design choice, layout tree structure, language bindings, extensible parser, and overlays similar in spirit to Firefox overlays. Ben demonstrated a PowerPoint application designed as a Webapp. He also explained that, as part of the future work, they plan to add conflict detection of extensions, security monitoring, pushing the limits of HTML5, and new user interface ideas to their framework.

Emmanuel Cecchet asked whether they expect to see C3 portability on iPhone or Android. Since more and more applications are becoming like a Web application platform, they will be treated like any Web application. Another questioner pointed out that most people are terrible at making UIs that are easy to use. Given this, will having such frameworks help? Ben answered that extensions don't become popular if they break the UI.

Lessons and Experience

Summarized by Ioannis Papagiannis (ip108@doc.ic.ac.uk)

The Effectiveness of Application Permissions

Adrienne Porter Felt, Kate Greenwood, and David Wagner, University of California, Berkeley

Traditional operating systems associate permissions with users, and this leads to overprivileged applications. Instead, modern mobile operating systems such as iOS and Android use fine-grained permission systems. Each application requests a set of privileges that the user has to approve. This can happen either at runtime or at installation time. In theory, a permission system can make users aware of what the applications they install can do. Moreover, fine-grained permissions can limit the impact of vulnerabilities of benign but buggy applications. However, this assumes that applications do not request more permissions than those they really need and that the permission system's design enables useful applications using only a few permissions. So, are modern permission systems effective? This paper, presented by David Wagner, attempted to answer this question by studying existing platforms that use fine-grained permissions.

For their evaluation, the authors tested a lot of Chrome extensions and Android Market applications. They categorized the extensions and the applications according to the dangerousness of the permissions that they requested. For Chrome, only 3% of the extensions requested the ability to run native code and another 50% asked for a significant number of dangerous permissions. For Android, most applications use fewer than four dangerous permissions. However, only 9% of Chrome extensions and 10% of Android applications do not ask for any permissions, and this can result in warning fatigue to the end users. Overall, David argued that the permissions system is better than the traditional overprivileged approach, as most extensions and applications are now significantly more constrained than before.

The Q&A triggered a lively discussion. What is the correct permission granularity? They do not know, but the granularity of permissions is important as it can reduce the frequency of warning prompts. Since a lot of users may consent to any warning, can we be optimistic for the future of permission systems? Yes, because a vocal minority of users who care about security will push the developers to only ask for the permissions they really need or to justify their requirements.

Experiences on a Design Approach for Interactive Web Applications

Janne Kuuskeri, Tampere University of Technology

Current Web applications are developed with technologies more suited for traditional Web pages. Lots of Web requests waste bandwidth, as they propagate client data such as views multiple times. On the server side, application state, client state, and views are all mixed to generate the correct reply. Different, fragmented technologies such as JSPs, CSS, JavaScript, and HTML are all used for a single page. This makes good software patterns very hard to apply. Model-View-Controller (MVC) is helpful, but multiple vendors use implementations that, although similar, are sufficiently different to complicate Web development.

To facilitate Web application design, Janne Kuuskeri suggested two ideas: (1) implement the whole MVC stack in the client's Web browser with JavaScript, and (2) have the Web server expose a RESTful API that all clients will use. Web applications will be single pages that load once the necessary scripts from the server arrive and then issue AJAX queries according to the client's actions. This decouples the Web client from the Web service, allows native applications to use the same API as the Web front end, and facilitates security and error handling. With a suitable JavaScript framework, developers do not even have to worry about HTML and CSS.

The limitations of the approach are the lack of support from existing Web frameworks and the reduced ability to crawl the resulting application for search purposes. This architecture is used in Valvomo, a production system in Finland's transportation sector that Janne demoed.

Exploring the Relationship Between Web Application Development Tools and Security

Matthew Finifter and David Wagner, University of California, Berkeley

Modern Web development is characterized by an immense number of choices: programming languages, frameworks, template libraries, etc. But how should one choose? Are they all equal? The goal of this paper, presented by Matthew Finifter, was to evaluate the security of different platforms. For this, the authors used a data set from a programming contest. Nine teams of similarly experienced developers were given the same specification to create a Web application in 30 hours. Developers were free to select their own tools and languages.

Given these nine implementations (three in Java, three in PHP, and three in Perl), the authors did black-box penetration testing and manual security audits to discover vulnerabilities. Matthew reported that there is no statistically significant association between the language and the number of vulnerabilities of each implementation. The authors also studied the association between the existence of framework-provided security features and the number of vulnerabilities found for each implementation. Again, there was no significant association for XSS or SQL Injection but there was for Cross-Site Request Forgery and Session Management vulnerabilities. Overall, they report that framework-provided security features do have a measurable effect on the number of vulnerabilities, but only when the feature is fully automatic and does not require the developers to understand it and use it correctly. The authors also report that for all three languages, there is framework support to automatically prevent all types of the vulnerabilities that they identified, but this support was not always used by the developers.

Matthew was asked to estimate the total sample implementations that would have been required to generate statistically significant results. He replied that he preferred to audit applications using more samples of smaller and simpler implementations over fewer samples of more complicated ones. He also mentioned that developers should not be blamed for the increased numbers of vulnerabilities; instead, the community should focus on providing fully automatic security features for most popular Web frameworks.

Joint ATC and WebApps Invited Talk

Helping Humanity with Phones and Clouds

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

See the 2011 USENIX Annual Technical Conference report for this session.

Panel: The Future of Client-Side Web Apps

Moderator: Michael Maximilien, IBM Research Panelists: Patrick Chanezon, Google, Inc.; Charles Ying, Flipboard, Inc.; Erik Meijer, Microsoft Corp.; Raffi Krikorian, Twitter, Inc.

No report is available for this session.

Extending and Protecting the Client

Integrating Long Polling with an MVC Web Framework

Eric Stratmann, John Ousterhout, and Sameer Madan, Stanford University

Detecting Malicious Web Links and Identifying Their Attack Types

Hyunsang Choi, Korea University; Bin B. Zhu, Microsoft Research Asia; Heejo Lee, Korea University

Maverick: Providing Web Applications with Safe and Flexible Access to Local Devices

David W. Richardson and Steven D. Gribble, University of Washington

No reports are available for this session.

Joint ATC and WebApps Invited Talk

Software G Forces: The Effects of Acceleration

Kent Beck, Facebook, Inc.

See the 2011 USENIX Annual Technical Conference report for this session

3rd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '11)

Portland, Oregon
June 14–15, 2011

Scheduling and Resource Management

Summarized by Byung Chul Tak (tak@cse.psu.edu)

Static Scheduling in Clouds

Thomas A. Henzinger, Anmol V. Singh, Vasu Singh, Thomas Wies, and Damien Zufferey, IST Austria

Damien Zufferey presented a job execution environment, called Flextic, for cloud applications. The goal of Flextic is to find a good interface between a cloud provider and customers. The user is expected to specify various job characteristics that will allow the provider to make efficient use of the datacenter resources. First, the user submits a program with information about resources, e.g., task durations and data size, using a custom language. This program is parsed to produce an execution plan. Then the scheduler generates several schedules for the given execution plan, with corresponding prices. The users select the preferred schedule, and this schedule is carried out by the execution platform. In selecting the most suitable schedule, the user can consider a price curve in which the price is high for shorter execution time and low for longer execution time. Choosing a longer time gives the provider more flexibility to optimize the global state of the datacenter; hence the price is lower. One challenge with this approach is that it requires solving large scheduling problems, so an abstraction technique is proposed to reduce the problem size (job model and infrastructure model) into a smaller one by exploiting regularities in the application.

The first question was whether it was fully implemented and what sort of tools were used. Damien said there was a proof-of-concept implementation. The scheduling part could scale well, but the execution part was still simple. Glenn Ammons (IBM) asked whether the technique was going to be used as an estimate or whether it would enable bidding. Damien replied that bidding would certainly be possible if people agreed on ways of describing the jobs. But the more fundamental question is whether the information needed in the technique is realistic. In the study, the main focus was scientific applications, but whether this technique is applicable to the more general class of application is important.

Migration, Assignment, and Scheduling of Jobs in Virtualized Environment

Seung-Hwan Lim, Pennsylvania State University; Jae-Seok Huh and Youngjae Kim, Oak Ridge National Laboratory; Chita R. Das, Pennsylvania State University

Why do we need to care about performance unpredictability in the cloud? Seung-Hwan Lim claimed that unpredictability creates a cascade effect in all the related jobs: a low-performing outlier dictates the overall performance of the entire application. In order to address this problem, virtual machine (VM) scheduling or reassigning to different physical machines has been considered. Amid VM scheduling, he mentioned that a set of VM migrations occur, and migration policy, in turn, determines the performance impact in reassigning VMs. He presented his measurement that showed that migration time could vary according to system configuration and how to group VMs for migration. He formulated an optimization problem that tries to minimize the total migration time when migrating a set of VMs while bounding the performance impact. This formulation allows him to estimate the completion time when multiple jobs contend for multiple resources. He also proposed performance slowdown as the metric of performance variance, which can be calculated from his formula.

How would this work handle cases in which jobs were dependent? This work assumed only independent cases, in order to ease the difficulty of calculating the probability of contention across multiple resources. The dependent case is more challenging and would be a direction for future work. Byung-Gon Chun asked how much accuracy degraded in estimating finish time when more than two jobs were considered. Lim said results showed about 15% accuracy degradation with up to four co-located workloads. How does this work compare with existing live migration work? Lim replied that many have considered the optimal state in terms of VM assignment, but this work looks at what happens during the state transition to optimal states. A live migration addresses migrating a single virtual machine, but they dealt with multiple VM migrations bringing a greater performance impact than a single VM migration.

Cloud Scale Resource Management: Challenges and Techniques

Ajay Gulati, Ganesha Shanmuganathan, Anne Holler, and Irfan Ahmad, VMware, Inc.

Ajay Gulati argued that resource management is critical for cloud deployments, both private and public. A desirable cloud management solution should provide high elasticity (i.e., scale) as well as high efficiency in terms of utilizing hardware resources. Current small-scale management solutions such as VMware DRS have some difficulty scaling up to the cloud level, but cloud providers want to maximize system efficiency in order to achieve higher revenue. On the other hand, cloud providers such as Amazon EC2 do not provide a rich set of resource management controls to allow for better multiplexing and over-commitment of hardware resources.

DRS provides an abstraction called "Resource pools trees" that allows the specification of resource allocation in a hierarchical manner. The VMs are the leaves of this tree, and one can specify controls such as reservation, limit, and shares for each inner node of this tree hierarchy as well as VMs. These controls dictate the minimum, maximum, and dynamic resource allocation in case of contention. The benefit of this technique is that it allows you to specify the aggregate amount of resources instead of per individual VM, and actual resource allocation per VM can be dynamically controlled. In the case where some resource is idle, it is reassigned to the other VMs within the immediate group first before being made available to any higher grouping units.

However, when trying to scale DRS to cloud scale, there are several challenges. First, resources are heterogeneous, so some VMs cannot be hosted on some set of machines due to various constraints such as storage and network connectivity. Second, operations need to be carried out at high frequency. With this, the centralized scheme can suffer from lock contention issues, and distributed schemes need to make decisions based on partial information. Lastly, failures are common at cloud scale. In order to deal with these issues, three techniques are proposed: hierarchical, flat, and statistical scaling. Hierarchical scaling builds a load balancer on top of clusters that are managed by DRS. Flat scaling builds an overlay of virtualized hosts (the main difficulty of this being the lack of consistent views). In statistical scaling, a subset of the cluster is selected to form an eCluster, and DRS is run on this eCluster.

Andrew Warfield mentioned that the DRS algorithm in the paper seemed to be non-terminating in the review version and the final version had newer features that made it look different. Ajay responded that the final version of the algorithm included additional details and that the algorithm does terminate in all cases. Some parts, such as cost-benefit analysis, were intentionally left out due to space limitations. Michael Schwartz asked about the disadvantages of statistical approaches. Ajay replied that looking at a subset does not give you the best solution, because picking some random set of machines implies some loss of efficiency. But, given the result from the power-of-two choices, the loss of efficiency should be small.

Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud

Gunho Lee, University of California, Berkeley; Byung-Gon Chun, Yahoo! Research; Randy H. Katz, University of California, Berkeley

The talk was about how to allocate resource and schedule jobs in a heterogeneous environment. Lee argued that the heterogeneity of a system environment made such tasks difficult, because some jobs require special hardware support such as a GPU. The proposed approach takes into account this need for specific hardware and, using the ProgressShare metric, provides fair scheduling of jobs. The traditional unit of scheduling was number of slots, but this does not work well in a heterogeneous environment. The proposed ProgressShare metric brings actual progress into the picture, and it can be used to schedule the jobs so that multiple jobs can make more progress than when using the SlotShare metric.

Orna from Technion asked how they would measure the ProgressShare of all the jobs that would have run using all the slots. Lee replied that they could run it on a small set of machines as a micro-benchmark to estimate the progress and could also utilize the historical data. Orna also asked what happens when jobs report false progress numbers to gain some benefit. Lee said that jobs cannot benefit from lying. If one job says this machine is bad when it's actually good, then it will receive more bad machines. On the other hand, if the job says this machine is good when it's bad, it can prevent other jobs from using that machine, thereby harming other jobs.

Ion Stoica opened the panel discussion by asking how each author took into account different types of resources and whether they had plans to incorporate them into their work. Ajay said that in VMware they had hosts for VM placement and that they considered CPU and memory during the placement. Virtual disk placement is a separate problem. Virtual disks can be placed in the best data store and the user VM on the best host in terms of CPU and memory. Demian said that, concerning I/O, they did consider the location of files for better scheduling, but did not deal with network congestion. Lim added that when you share disks such as SSD, workload pattern would differ in each case because access patterns change.

Chit-kwan Lin asked about time granularity in managing the cloud infrastructure. According to Ajay, in VMware products it is about five minutes; running the load-balancer takes about a minute to finish at the current level of scalability. Any finer time granularity will cause too much overhead. If the scale grows, the time has to increase to maybe 15 or 20 minutes.

The session chair, Byung-Gon Chun, asked how often the VM migration happens in practice. Ajay said that it depends on the specific environment; some clients said that there were many migrations once they turned on the DRS algorithm. But migration definitely happens more frequently than people expect. Steven Ko from SUNY Buffalo asked why scheduling research is still active after being studied for so long. Demian said that some of the assumptions have changed in the cloud environment, so new heuristics may be required. One attendee pointed out that scheduling is important because H/W cost is high and providers would like to utilize the infrastructure to the maximum. Orna followed by saying that there are two new things about cloud: there are consumers who pay for execution where grid is a kind of best-effort thing, and virtualization introduces interference. She argued that we should adopt work from grid computing.

Economics

Summarized by Tian Guo (tian@cs.umass.edu)

To Move or Not to Move: The Economics of Cloud Computing

Byung Chul Tak, Bhuvan Urgaonkar, and Anand Sivasubramaniam, The Pennsylvania State University

Byung Chul Tak began with the benefits of cloud computing: cost saving, automatic scalability, and transparent redundancy. The focus of the talk was the cost-saving aspect of migrating an application to the cloud. Pay as you go and elasticity make it cheaper and easier to match the cloud more closely with demand. However, there is no consensus that the cloud is really saving money. Therefore, in this work, they tried to systematically investigate the conditions and variables affecting the benefits of clouds, studying two representative applications from which they could draw conclusions.

The cost assessment process of their framework involves specifying application properties and potential service time in order to calculate hardware configuration, identifying direct quantifiable costs (converting future cost into present cost for fair comparison by using NPV), and comparing these costs among five hosting options for high and low workload intensities. The conclusion they drew is thatcloud-based hosting is preferable for lower workload intensity with a smaller growth rate. He also said that data transfer cost would be significant and component-level partitioning can be costly. In the analysis of the effect of storage and software license, he further explained the importance of these two factors in decision-making. Last, he briefly mentioned the need for more accurate performance estimation of cloud-based application and economic study for scientific applications.

In the Q&A, Suman Jana (UT Austin) pointed out the work only considers steady increasing workloads and doesn't have a model for spiky workload. Byung Chul Tak said that they did consider the workload in a variant increasing speed, but the workloads are flattened over a long period of time. Andrew Warfield (University of British Columbia) asked whether they were assuming cheaper leasing prices over time. The speaker expressed his hope for reduced fees. Andrew asked whether the cloud-based results are more efficient. Byung Chul Tak answered yes.

Cutting MapReduce Cost with Spot Market

Huan Liu, Accenture Technology Labs

Huan Liu told us that the work is about how to save money with the spot market, and he explained why it is reasonable for a spot market to exist in the public cloud. There are unpredictable spiky workloads even in a large cloud provider like Amazon. By providing economic incentives, users will have the motivation to move around their demand and help to smooth out utilization for the providers. The problem with the spot market is its unpredictability. The Cloud MapReduce architecture employs a distributed approach implementing several queues, using Amazon's Simple Queue Services. He said that the data processed in Mapper will be sent to a reduce queue instead of stored locally. The reducer will pull all the messages in the reduce queue after receiving the pointers from the master reduce queue. Later on the reducer will generate output messages and send them to the output queue.

Huan Liu highlighted two important things: the checkpoints are stored in the SimpleDB and all the intermediate results get sent to reduce queue as soon as possible; there is streaming going on between MapReduce stages. Specifically, in one map node, there is a temporary buffer storing only one key-value pair. The idea is to gain some time before the cloud provider actually shuts down the instances, in order to flush good results to reduce queues and to check the assignments left in input queues by preventing the soft shutdown script from executing. Cloud MapReduce is the first ever implementation that works for spot market and saves a lot of money.

What happens if there is not enough time for flushing? Huan answered that the correctness of the results is guaranteed, even though some partial results will get lost because the partial results will never be committed in the SimpleDB. Is it more valuable to have a hybrid scheme with control nodes that are not spot market instances? Huan said yes, in the sense of performance guarantees. Andrew Warfield (University of British Columbia) asked how much money could be saved by using the Cloud MapReduce. Huan said they would save more money if prices fluctuated frequently. Even if the client bid for the highest price in the spot market every time, there would still be a 50% saving.

Exertion-based Billing for Cloud Storage Access

Matthew Wachs and Lianghong Xu, Carnegie Mellon University; Arkady Kanevsky, VMware; Gregory R. Ganger, Carnegie Mellon University

Matthew Wachs said that the focus is on Infrastructure-as-a-Service in cloud accounting. In this setting, the providers want to recover their costs and the clients want to be charged fairly. Matthew focused on the storage consequences of accounting. He noted that providers only bill for capacity but fail to bill for access, which also incurs cost in buying more disks. He further pointed out that current metrics, such as IOPS or bytes transferred, are not directly proportional to time used and therefore are wrong. He later gave an example of billing for fixed I/Os, which is unsustainable in most cases. A few more alternatives were listed but none of them is ideal. Matthew said that charging for disk time is the fairest solution and that workload interference might affect the exertion required. The solution is to use performance insulation to avoid interference in the first place. He said that storage insulation could be achieved by preserving locality and providing predictable cache allocation. After using the insulation to limit the impact of other workloads, the exertion shown is close to ideal. Finally, he pointed out the importance of using disk time as a metric and performance insulation to guarantee fairness in billing for clients.

Zachary N.J. Peterson (Naval Postgraduate School) asked why providers are still using the metrics Matthew claimed to be wrong. Current metrics are easier for customers to understand and therefore they are more willing to pay. There are disadvantages with exertion-based billing, such as less transparency. Zachary asking about the mechanism that prevents providers from increasing the clients' disk time to create higher revenue. Matthew replied that billing for CPU time also has the same issue of provider trustworthiness; both need to be solved. Eyal de Lara (University of Toronto) questioned the advantages of this relatively complicated billing model compared to the current simple billing model. Matthew thought the sacrifice of simplicity is worthwhile, since clients are now paying much more money than they should. There was another question about why a simple billing model employed by a mobile company is not sufficient for cloud billing. Matthew pointed out there are a lot of available options in mobile markets and noted that the money paid for cell phone bills and for cloud service bills are not on the same scale. People would care more about precise accounting in the cloud environment, because of the amount of money involved.

John Wilkes (Google) asked how they would take latency of requests into account. Matthew said latency would not necessarily incur higher costs for providers. John again doubted the potential of the proposed billing model by giving an example of the potentially higher cost incurred by sequential accesses. Matthew concluded that disk time also belongs to opportunity costs with which we should motivate the billing models.

Panel

What do academics need/want to know about cloud systems?

Panelists: Chris Colohan, Google; Greg Ganger, Carnegie Mellon University; David Maltz, Microsoft; Andrew Warfield, University of British Columbia

Summarized by Sahil Suneja (sahilsuneja@gmail.com)

This was a post-lunch session and meant to instill enthusiasm and energy into the audience. The discussion truly lived up to expectations with the elements of fun and liveliness.

The panel started off with the industry people, David and Chris. David offered his list of things he wished everyone knew. A major point was that datacenter costs have unfortunately remained stable over time—the cost of a server today, irrespective of whether it is being used or not, is about $55 per month—so turning off servers isn't an attractive option. He also talked about the primary metric for a datacenter being profit, and that translates into minimizing costs by buying cheaper systems, increasing resource utilization, and reducing the cost of delivering power to the datacenters.

Chris emphasized the need for efficient and reliable datacenter design/layout. He raised questions regarding the means of estimating resource requirements for datacenters and the requirement of a theoretical basis for the correct size of clusters. Similar issues included the need to decide the number of machines per cluster vs. number of clusters, hierarchical cluster designs, decisions regarding uniformity/non-uniformity of machines, and hardware-software logic distribution, among other concerns. He talked about the lack of batch work to soak up available computational resources today, and the common error of using today's workloads to estimate future cluster sizes.

On the academic front, Andrew stressed the need to obtain realistic hints and technological constraints from industry people for driving research. The more disclosure behind the internal functioning of commercial datacenters, the better the improvement opportunities for researchers. He highlighted the practical necessity of the industry people and academic researchers spending time together to keep innovating and building systems in interesting ways.

Greg tried to play the old curmudgeon, targeting Dave in particular. Like Andrew, he emphasized the importance of getting feedback from industry so that researchers are prevented from making imprudent assumptions. At the same time, he stressed the importance of academics questioning industry-provided assumptions as perhaps being the bottleneck for just one particular industry instance vs. a global phenomenon.

Michael Kozuch from Intel kicked off the discussion by inquiring how academics should think about scale. Dave and Chris confessed that they are confronted with this issue in industry as well. Dave dodged the question by saying that a thousand is small while a million machines is big, to which Greg playfully responded that these bounds are beyond what academics have access to! John Wilkes from Google teasingly advised the panelists to collaborate like physicists; Greg responded that physicists have an entire generation of students who work on constructing mechanisms that one day would allow the experiments to run, and that is impractical in computer science.

Tal Garfinkel from VMware steered the discussion away from scale in cloud computing and sought the panelists' thoughts on an autonomous self-serving collection of machines, data service, and applications that eliminates the need to involve IT and operations people. Chris diplomatically answered that the reason behind automating system administration could be the system scale that might make a robot a more economical option than actual people to manage the system. David agreed that this is a hard problem, as it requires the knowledge of what metric is to be optimized and how to obtain the input/output data. This needs both applications and people to find out edge cases, as Tal had mentioned in his question. Greg challenged the notion and argued that people have been working on automation and a lot of work is focused on problem diagnosis, which seems to be the hardest issue to deal with. He was surprised to see a response coming from industry to eliminate IT people. Chris took the opposite route and stressed the need for more operational staff at Google, which Greg criticized by commenting that it's the PhD students that are the operational staff at Google! The crowd joined in by arguing for the need to employ people!

As the session ended, John gave a lighthearted final remark—a suggestion that every academic to pursue an industry person for an interesting problem until he grasps the problem completely.

Security

Summarized by Tian Guo (tian@cs.umass.edu)

The **HybrEx** *Model for Confidentiality and Privacy in Cloud Computing*

Steven Y. Ko and Kyungho Jeon, University at Buffalo, The State University of New York; Ramsés Morales, Xerox Research Center Webster

Steven Ko talked mainly about how the Hybrid Execution (HybrEx) model fits into the context, instead of the details of system implementation. He put forward a general question about the trustworthiness of the cloud environment. The main focus of the work is to figure out how to utilize clouds with partial trust. The realities of people's distrust, the potential threats to the cloud, and the benefits of using the cloud make the problem worthy to explore.

In the current cloud environment, there are only two extreme options for people: complete trust or distrust of the cloud. Steven remarked that HybrEx is a solution for dealing with the unexplored middle ground. He explained that the main ideas of HyberEx are partitioning and tainting. For partitioning, HyberEx categorizes data as either public or private. Since there are just partial trusts for the cloud, the client will only deal the private data in a private cloud environment. In order to prevent information leakage, tainting is used to keep track of the data. After explaining the framework of HyberEx, Steven discussed the specific contexts it applies to, namely MapReduce and Bigtable. The popularity and feasibility of the two applications make them ideal for a good start. However, there are challenges such as finding the appropriate applications which will benefit from data partitioning, sanitizing data to enable private to public shifts, potential performance decrease due to higher communication cost, and the correctness of the computation.

Aditya Akella (University of Wisconsin—Madison) asked whether it is easier to track public and private data on a large scale. Steven said building a tainting system inside a framework like MapReduce might solve the problem. Since MapReduce is already well partitioned, the tagging of private and public data is relatively easy. Suman Jana (University of Texas—Austin) questioned the scalability of the taint tracking of data in a virtual machine level because of the significant overhead. Steven said the tainting is incorporated into the MapReduce level instead of the virtual machine level in order to lower the overhead. Aditya questioned the difference between data partitioning and vertical partitioning. Steven replied that the existence of the hybrid approach is not the same as partitioning data into public and private. Following up, Aditya was curious about the cost-benefit analysis of hybrid execution. Steven thought that was a good direction for future work.

A Position Paper on Data Sovereignty: The Importance of Geolocating Data in the Cloud

Zachary N.J. Peterson, Mark Gondree, and Robert Beverly, Naval Postgraduate School

Zachary said that most people don't care about the location of data as long as it is accessible. However, it is a non-trivial problem, especially in the cloud environment, considering that some data should stay within political boundaries even when including data replication. Traditional data location doesn't represent the actual place where data is stored. The purpose is to efficiently locate some copies of data within certain boundaries. He stressed that tracking all copies is a hard and interesting problem.

Zachary explained that there are two techniques: geolocation of the host and possession of data. Simply combining the two techniques won't solve the problem, though. It only proves the existence of the host instead of the data. In addition, he pointed out that adversaries might purposely fake the data source by adding delay. For example, if some Web proxies cached subsets of the data and manipulate the delay measurements, people would gain incorrect information about the data location. He mentioned an important aspect of network measurement, which is that the server can only pretend to be outside the bounding area and never falsely pretend to be inside. An initial approach is leveraging MAC-PDP (a signed statement of Provable Data Position), which can be augmented with network delay measurement. In order to get the exact location of the data, multiple challengers should be used. The merits of the approach are it minimizes the latency without requiring the server-side computation, and it is easy to apply to existing infrastructure. However, higher communication costs are expected. Future directions include the evaluation of the initial idea and placement of landmarks.

Suman Jana (UT Austin) asked how to ensure the effectiveness of the mechanism if users want their data to be outside the boundaries instead of within. Zachary admitted that one-way verification will not help in this case but as a solution proposed doing computation that binds location. Chris Colohan (Google) questioned whether it is necessary to retain a copy of data locally in order to know the locations of other copies. Zachary responded that MAC-PDP protocol clients only need to store a MAC key k instead of the whole copy of data. It will recompute the copy to verify the authenticity. Someone asked about the legal framework regarding the goal of this work. Zachary said that it lies mostly on the IP side and made analogies to privacy violation and medical records. Aditya Akella (University of Wisconsin—Madison) mentioned that the extra copies outside the system are hard to track. Zachary agreed and pointed out the paper tries to show that the copies of data are within some boundaries.

Privacy-Sensitive VM Retrospection

Wolfgang Richter, Carnegie Mellon University; Glenn Ammons, IBM Research; Jan Harkes, Carnegie Mellon University; Adam Goode, Google; Nilton Bila and Eyal de Lara, University of Toronto; Vasanth Bala, IBM Research; Mahadev Satyanarayanan, Carnegie Mellon University

Wolfgang Richter started his talk by explaining the difference between introspection and retrospection for virtual machines. In introspection, we examine the live logs of a virtual machine during its execution. In retrospection, however, we can have access to all historical logs of all the virtual machines. He pointed out that we should treat VMs as big data instead of executable content. Retrospection is about deep search over historical VM data at a raw-data rather than metadata level while respecting privacy. For example, VM retrospection can be used as a unified interface to search all the historical data in a compromised VM for the root cause of the exploit. Searching a set of instances for privacy violation among different companies who use a similar cloud infrastructure would be another case. He said that the privacy goal of VM retrospection is achieved by data cryptography. So far they have explored the per-file, per-directory, and per-partition data encryption.

Wolfgang briefly described the design principle of their work. They provide on-demand search through the unified interface. Second, they offer VM owners the right to make the suitable retrospection policy. Last, they try to support generality of search. He concluded by talking about the implementation called Nanuk.

What level of VM data structure should be coupled with the implementation? Nanuk could query whatever data was available, regardless of the structure. What is the advantage of doing this work at the VM level? Wolfgang pointed to the potential security gain if the whole operating system is compromised. The snapshots of the VM guarantee the integrity of data even in the worst case. Aditya Akella (University of Wisconsin—Madison) asked about the trust model. Wolfgang replied that the trust model they explored provides as much as possible to the VM owners. The search of private data is possible only if the key is provided by the owners. Chris Colohan (Google) questioned whether it is necessary for VM users to take the snapshot after the data is encrypted. Wolfgang admitted that it is a question worth thinking about.

EVE: Verifying Correct Execution of Cloud-Hosted Web Applications

Suman Jana and Vitaly Shmatikov, The University of Texas at Austin

Suman Jana started his talk by providing a scenario of an interactive Web application running in the cloud. Once the application is submitted to the cloud, the correctness of the application is not visible to the owner. The incorrectness could be caused by things such as network failure, storage failure, or consistency failure. Knowing the detailed information about the application failure is crucial to owners, and running the applications in the cloud makes this a more challenging problem due to the low visibility of the cloud environment. He stressed that we should think more about consistency and partition tolerance compared to the availability of the service.

Suman gave an example of transient error in a tax application due to the low share of storage in the cloud. It is only possible to track the inconsistency if the owner monitors the application consistently, he commented. The focus of their work is to continuously verify the correctness of interactive Web applications. After analyzing the architecture of popular applications in the cloud, he concluded that the focus should be on verifying the consistency of data store operations. By checking consistency violations in the data store, faults would be easy to track. With EVE, witnesses keep logs of operations and send them to the verifier periodically for error detection performed by the streaming consistency verification algorithm. Finally, he talked about different scenarios where EVE could be useful, including checking the scalability of the application and comparing the quality of service among cloud providers.

Zachary Peterson (Naval Postgraduate School) asked about the efficiency of EVE's error detection for WordPress. Suman said that since WordPress doesn't employ an eventual consistency back-end database, no consistency violation can be detected. Zachary asked about the privacy issues of logs generated by the witnesses. Suman replied that the clients have the right to block some sensitive information and still have the potential to detect errors. Aditya Akella (University of Wisconsin—Madison) asked about the feasibility of mapping Web application operations to data-store operations. Suman admitted the importance of having a generic SQL-like interface, which will enable the portability of a variety of back-end infrastructures. Aditya wondered whether EVE could deal with things like quality degradation in the streaming service. Suman said that EVE is mainly designed to facilitate error detection in interactive Web applications. Also, it is not practical to keep logs of streaming service, due to the potential for growth in log file size.

Networking & Energy

Summarized by Byung Chul Tak (tak@cse.psu.edu)

Jellyfish: Networking Datacenters, Randomly

Ankit Singla and Chi-Yao Hong, University of Illinois at Urbana—Champaign; Lucian Popa, University of California, Berkeley; P. Brighten Godfrey, University of Illinois at Urbana—Champaign

Chi-Yao Hong presented Jellyfish, a technique for constructing datacenter networks that enables easy incremental expansion without sacrificing network bandwidth. He argued that one critical problem in datacenter networking is to enable incremental expansion and that current datacenter networks did not support this well. One commonly used fat-tree scheme allows bandwidth at a very coarse level limited by the available port count of switches. Upgrading switches in fat-tree schemes also requires full replacement. Other schemes suffer from similar drawbacks.

Jellyfish is based on the random graph, which makes it simple to expand the network to any desired size and provides high resilience from failures. Also, Jellyfish delivers more bandwidth than fat-tree structures in terms of bisection bandwidth. However, challenges remain. An unorthodox routing technique is required, since traditional techniques are mostly based on structural assumptions. He also discussed cabling issues. In order to connect N racks, he suggested using the square root of N as the number of rack clusters.

Ion Stoica asked about the impact of the square root of N on expandability; doesn't enforcing it require recabling other clusters of racks, which goes against the goal of easy expandability? And how does using the square root of N perform compared with a smaller random topology? Finally, in expanding servers, what is the impact of a large number of servers coming together? Hong responded that we do not have to stick with a square root of N option. It could be a starting point for cable configuration.

SilverLine: Data and Network Isolation for Cloud Services

Yogesh Mundada, Anirudh Ramachandran, and Nick Feamster, Georgia Tech

Yogesh Mundada reported that the recent series of data leakage incidents in major clouds made it difficult to adopt the cloud. Threats in the cloud can be classified into attacks on the shared resources and data loss/leakage. In order to address these problems, he proposed a technique called SilverLine to provide data and network isolation for VMs in the cloud environment.

SilverLine delivers data isolation by labeling data via information-flow tracking tools so that an enforcer module at the hypervisor level can check for any unauthorized access. If one malicious user tries to steal data through a SQL injection attack, the data will not be delivered to the attacker at the front end, because any data not owned by the attacker will be filtered by the cooperation of the declassifier and the enforcer. For network isolation, SilverLine makes use of IP address obfuscation and ping response normalization. This prevents attackers from identifying the location of victim VMs on the physical node.

Would introducing delays for network isolation have some performance impact? Yogesh said that there were no measurement numbers regarding performance impact, but he thought that it would be minimal.

Enabling Consolidation and Scaling Down to Provide Power Management for Cloud Computing

Frank Yong-Kyung Oh, Hyeong S. Kim, Hyeonsang Eom, and Heon Y. Yeom, Seoul National University

Frank Yong-Kyung Oh presented measurement studies of performance interference when VMs are consolidated. The goal of his study is to better understand the performance impact of VM consolidation so that it can be used for VM migration scheduling and consolidation in future studies. When several VMs with distinct characteristics are given, one of the goals is to consolidate them so as to minimize the effect on performance and the number of physical machines. This would allow us to turn off some servers, saving power consumption. They specifically looked at three effects: the effect of VM co-location, cache effect, and the effect of CPU thermal throttling. From studying the effect of VM co-location, they found that consolidating VMs that use different parts of resources in the system shows less performance interference; they also found that CPU and memory-intensive applications tend to consume more power than others. The cache effect revealed that disk-intensive VMs show better performance when pinned together with Dom-0 in Xen. And the insight gained from the thermal effect was that consolidating only CPU-intensive VMs may lead to unexpected performance degradation due to CPU thermal throttling. There were no clarification questions after the presentation.

The Data Furnace: Heating Up with Cloud Computing

Jie Liu, Michel Goraczko, Sean James, and Christian Belady, Microsoft Research; Jiakang Lu and Kamin Whitehouse, University of Virginia

Jie Liu presented Data Furnace, which proposes the use of server-generated heat as a household heating solution. He argued that energy can be more efficiently used by dispersing servers to homes or other buildings. This would provide the additional benefit of bringing computation closer to the user and some cost reduction from reusing already existing power infrastructure.

There were several interesting numbers from the presentation: home heating cost is about 10% of total home expenditure, and in the US, home heating is twice the IT energy cost. Another interesting figure was the estimate of number of servers required to properly heat a typical home in different regions of the US. San Francisco showed small variance of the number of servers, whereas Minneapolis showed large variance ranging from 140 servers to fewer than 10 servers.

Towards the end of the presentation, Jie outlined some of the questions regarding the usefulness of this idea, what the hidden costs are, thoughts on residential power capacity, some security issues, and performance concerns. In summary, this was an interesting idea with many challenges.

Ankit Singla asked whether this study considered the cost of moving data in such a distributed setting. Jie said that the cost of paying for the network, which was about $3000 for one server, was included in the calculation. Ankit asked again if this idea was part of realizing the green computing concept. Jie explained that the money is being spent for heating the houses anyway. The installation of a data furnace is a one-time cost at house construction time, just as when you would normally install a furnace. One attendee asked if there was any interest from national security folks. Jie said that this did not represent the official view of the organization and no comments from either the government or industry were received. Next, Orna asked how to compare this with the datacenter in Switzerland that heats all the office water near the datacenter. Jie said that, although similar, it would ultimately be cheaper to move computation around than energy or hot water. Whenever we can put computing near to where heat is needed, we can save costs in terms of transporting other things.

The first question in the session panel discussion was about information visibility between VMs and the cloud infrastructure. Presentations seem to assume that underlying infrastructure needs certain information about VMs in order to work. How much do VMs (or applications) need to tell the infrastructure, will they be able to tell the infrastructure, and how much can the infrastructure trust the information? Yogesh responded that they have built their technique at the VM's OS level, which made things simpler. If they had to go down to the VMM-level, they would have more control over the data items but would also need to understand the higher-level abstractions, which would be non-trivial.

Ion Stoica argued that installation of a Data Furnace at home would require a large maintenance effort. Jie said that one maintenance concern of the server is perhaps to provide operation without harming the environment; some have studied reliability vs. environmental conditions and have found that conditions were tolerable. Another maintenance concern would be replacing a failed part, swapping disks, and so on, which would require someone to actually go in and take action. Those tasks could be handled by over-provisioning.

One interesting discussion took place about the incremental scalability of datacenters. When current datacenters expand their hardware equipment, they buy servers in large numbers and configure them for a relatively long operation time until the next upgrade takes place. Studies such as Jellyfish and Data Furnace allow the incremental expansion of datacenters: the cloud provider could bring in new servers frequently and in much smaller numbers.

John Wilkes asked why cloud providers would pay to have servers installed in homes when there are not enough workloads to utilize even the current hardware resources in the datacenter. Jie said that the Data Furnace approach would make sense if workload was overcommitted. However, more opportunity arises from content caching near to where contents are created and needed. If data is located where it is needed, it can be served faster. Someone brought up the issue that current network speed is not fast enough. Jie said his analysis included the cost of installing fast fiber network, and its cost did not end up dominating other cost factors. Nevertheless, networking would be the greatest challenge.

Poster Session

No reports are available for this session.

Joint ATC, WebApps, and HotCloud Keynote Address

An Agenda for Empirical Cyber Crime Research

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

See the USENIX ATC '11 reports for a report on this session.

OSes and Frameworks ("There is an OS/App for that!")

Summarized by Henrique Rodrigues (hsr@dcc.ufmg.br)

Unshackle the Cloud!

Dan Williams, Cornell University; Eslam Elnikety and Mohamed Eldehiry, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia; Hani Jamjoom and Hai Huang, IBM T. J. Watson Research Center; Hakim Weatherspoon, Cornell University

Dan Williams said that IaaS providers are mainly focused on giving virtual machines to users. Nowadays, none of the features at the infrastructure level, such as VM migration and CPU boosting, are exposed to the user. In addition, the research community is continuously proposing innovations at the hypervisor level. The problem in the current scenario is that users don't have control over any of these features. To overcome this limitation the group proposes xClouds. The goal of xClouds is to provide extensibility to IaaS-based resource provisioning. Unlike current public clouds, xClouds gives users the ability to leverage their own set of hypervisor-level modules.

Dan presented some design alternatives to implement an extensible cloud. Among the three options, two of them, VMM extensions and exposing the hardware through the VMM, depend on the adoption of a new VMM by the provider. The third option is to use nested virtualization, which doesn't need provider cooperation. This latter option was adopted by xClouds, which was implemented using Xen and tested on an EC2 instance. Dan also presented an evaluation of xClouds, comparing I/O performance between single and nested virtualization setups using a combination of Xen/KVM/HVM.

Himanshu Raj from Microsoft asked if Xen needs any modification to be run on top of EC2 instances as a nested VMM. Dan said that there are some changes required to support the paravirtualized device operations. Muli Ben-Yehuda from Technion/IBM Research commented that both KVM and Xen, in the next release, will have hardware support for nested virtualization, so the performance of xClouds will be better in the near future.

The Datacenter Needs an Operating System

Matei Zaharia, Benjamin Hindman, Andy Konwinski, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica, University of California, Berkeley

For many, the datacenter is like a big computer, where users can run their applications and process their data, either interactively or in a batch-processing fashion. To deal with the growing range of applications and users, Matei Zaharia claims, the datacenter will need an operating system.

The datacenter operating system is not a replacement for the Linux host OS but is software that acts as an operating system at the level of the whole datacenter. Matei listed some features that an operating system should provide to its users: resource sharing, debugging, monitoring, programming abstractions, and, most importantly, enabling independently developed software to interoperate seamlessly. Some platforms that took steps towards providing some of these abstractions are Hadoop, Amazon Services, and Google Stack. The problem with current solutions is that they are all narrowly targeted and are not general/longer-term solutions. In the last part of his presentation, Matei discussed the problems that should be solved to have a practical implementation of a datacenter OS and how researchers can help in this process.

Someone from the University of Toronto asked about the difference between existing cluster OSes and a cloud OS. Matei replied that the main difference is the diversity of users and applications using datacenters. Why did Matei think the idea of building a cloud OS would succeed? Matei said that some approaches are already successful and pointed some of them out.

Large-scale Incremental Data Processing with Change Propagation

Pramod Bhatotia, Alexander Wieder, İstemi Ekin Akkuş, Rodrigo Rodrigues, and Umut A. Acar, Max Planck Institute for Software Systems (MPI-SWS)

Pramod Bhatotia began by discussing the advantages of incremental computation on large-scale datasets. The main idea of incremental computation is to leverage previous processed results to enable more efficient computation of recently updated data. Computing the page rank of a recently crawled URL is an example of a good use case for incremental computation. Two systems for incremental processing, Google Percolator and Yahoo! CBP, were presented. Pramod pointed out that the main disadvantage of current approaches is the need to rewrite existing applications in new programming models using dynamic algorithms, which are harder to design.

Current large-scale applications are developed using static algorithms and well-known programming models. The goal of Pramod's work was to make these applications as efficient as the ones that make use of incremental computation and dynamic algorithms. His presentation focused on how to achieve this goal in a MapReduce-based application. Their approach was to take an unmodified program and automatically make it incremental by (1) dividing computation into sub-computations, (2) keeping track of input dependencies between each sub-computation, and (3) recomputing only the computations affected by input changes.

Pramod then discussed some of the challenges to building such a solution for MapReduce-based applications. To evaluate the performance gains of their solution, the runtime speedup was compared against an increasing input dataset.

John Wilkes (Google) asked if there is any restriction on implementing the reduce task in order to make it divisible into sub-computations. Pramod said that developers should use MapReduce combiners to achieve a fine-grained division of the reduce task. Christopher Colohan (Google) pointed out that on large MapReduce computations usually the smaller you make your task, the higher the management overhead is, which is the opposite of the results shown. Pramod said that when you are making use of incremental computation, this overhead only affects the performance in the first computation.

TransMR: Data-Centric Programming Beyond Data Parallelism

Naresh Rapolu, Karthik Kambatla, Suresh Jagannathan, and Ananth Grama, Purdue University

Naresh Rapolu explained why current data-centric processing models such as MapReduce or Dryad are unable to deal with the side effects of parallelized algorithms that present data dependencies. To exemplify the problem, he used a simple MapReduce-based word count application. The limitation is mainly due to the deterministic replay-based fault tolerance model adopted by these programming models. This fault model assumes that the application semantics won't be affected by re-running a computation task in the case of task failure. However, not all algorithms have a data-parallel implementation compliant with this fault tolerance model.

To support parallel algorithms' side effects and overcome the limitation of current data processing frameworks, they proposed a transition-based MapReduce programming model. The key ideas of their approach are to develop every data-centric operation as a transaction and to use a distributed key-value store as the shared memory abstraction accessed by all operations. The concurrency model is based on two operations: put and get. With optimistic reads and buffered writes they could build a programming model that doesn't require any locks.

For evaluating their programming model, Naresh presented the speedup of two algorithms implemented on their current prototype, which uses Hadoop and HBase as the key-value store. The algorithms are the Boruvka's Algorithm for finding a graph's minimum spanning tree and the Push-Relabel algorithm to find the maximum flow of a graph. Both experiments resulted in a speedup close to four times for 16 nodes.

In the panel discussion, Christopher Colohan (Google) asked whether we are able now to design an API that will be widely accepted and that will last for a long period of time. Matei answered that we can at least try starting from lower-level primitives and that future extensions or changes to this API would not be a problem, because even standard OSes have had to make changes to support newer technologies. Dan added that if we want this API to last for a long time, it needs to be user-centric. Ion Stoica asked Naresh if he had any idea how to improve their system in order to achieve linear scaling? Naresh said that there are a lot of parameters the user can tune to have better performance, but most of them are application-dependent.

Steve Ko (SUNY Buffalo) asked how we can deal with the size of a datacenter while designing an operating system for it. Matei replied that most of the scalability problems still have to be solved for the systems being built today. The interesting thing about designing a datacenter operating system is that once one problem has been solved, it is possible to incorporate the solution into the datacenter OS and, therefore, we won't need to keep solving the same problem for each independent platform.

Rodrigo Fonseca (Brown University) commented about possible optimizations for xClouds and also pointed out that some of them will depend on provider cooperation. Dan replied that the deployability of xClouds among multiple vendors was more important than specific optimizations for individual providers.

Glenn Ammons (IBM Research) asked Pramod if his group had evaluated their framework using real-world applications. Pramod replied that he didn't show the results because of the time constraint but that they tested their framework using the Apache Mahout library.

Rodrigo asked why Matei didn't mention the word "cloud" in his presentation and whether he thinks that there are differences between a public datacenter and a private datacenter. Matei said it was because they think that the datacenter operating system should be generic enough to run on both private and public datacenters, regardless of the differences between them.

Performance

Summarized by Sahil Suneja (sahilsuneja@gmail.com)

Modeling the Parallel Execution of Black-Box Services

Gideon Mann and Mark Sandler, Google Inc.; Darja Krushevskaja, Rutgers University; Sudipto Guha, University of Pennsylvania; Eyal Even-Dar, Final Inc.

Mark Sandler presented work in which the goal is to estimate the impact of a change, deep down in the call stack following a user request, on the latency of a higher-level service in the hierarchy.

Call trees do not encode the parallelism structure among multiple calls, and this acts as a hindrance to accurate latency estimation up the stack. The proposed approach automatically reconstructs the flow of a service by looking at the nature of overlapping between multiple RPCs and combining multiple invocations of the service to generate a consistent call graph. Since a service can have multiple flows (e.g., different RPCs depending upon cache hit/miss), two different schemes are employed for deciding whether two invocations follow the same flow-clustering invocations having identical control flow graph (better) vs. invocations calling identical sets of lower-order services. During the training phase, using either of these two approaches, flows are constructed, and average latencies at each node are recorded using actual traces. Then, in the testing phase, for a given set of latencies at child services, best matching flow is found and simulated and its latency at the parent is reported.

This approach can allow modeling service dependencies and aid in detecting potential problems caused by changes down in the stack.

No questions were raised at the end of the talk.

CloudSense: Continuous Fine-Grain Cloud Monitoring with Compressive Sensing

H.T. Kung, Chit-Kwan Lin, and Dario Vlah, Harvard University

Chit-Kwan Lin emphasized the relationship between performance and monitoring—the more the available information about the state of a cloud, the better the decision-making with regard to its management. With increasingly interactive applications, finer-grained status information would prove beneficial in improving application performance.

However, the major challenge to fine-grained monitoring is the bottleneck at the collection point. The example used was that of MapReduce straggler detection, where the sooner a straggler is detected, the earlier the job can complete. This requires global relative comparisons and, in turn, global status collection. To overcome the collection bottleneck, the status stream can be compressed in the network. But since distributed compression is hard, a compressive sensing technique could be used which increases reporting granularity via in-network distributed compression, so the largest anomalies could be detected first and with few reports. The proposed solution is a switch design for compressive sensing called CloudSense. For a single rack, status from each node is collected into the signal vector. Random projections are computed onto low-dimensional space, generating measurement vectors which are sent to the master. After recovering the original signal vector, the master solves for L1 minimization by linear programming. In the two-rack case, an aggregation switch is added that summarizes the measurement vectors so that the data sent over links does not increase.

Rodrigo Fonseca from Brown University inquired about the signals to which the proposed technique was applicable. Chit Kwan's response was to use this primarily for performance counter reporting similar to MapReduce task progress reporting. He also hinted at obtaining inputs from industry colleagues. Another question sought clarification on means to identify when the values of two of the design parameters, signal sparsity and number of measurements, were sufficiently large. Chit-Kwan said that since signal sparsity is a static system property, it isn't supposed to be set dynamically. But for the number of measurements, there is a tradeoff with regard to loss in decoding accuracy. If it is too large, mistakes could be made, but the model gives very few false positives, while the false negatives could be dealt with easily—the more measurements are obtained, the better the decoding sensitivity is.

Virtual Machine Images as Structured Data: The Mirage Image Library

Glenn Ammons, Vasanth Bala, Todd Mummert, Darrell Reimer, and Xiaolan Zhang, IBM Research

Just as a VM image puts application configuration in one place, similarly an image library collects all enterprise configuration together. This aids in simplifying maintenance operations such as patching and security scans, allows version and access control, and permits offline analyses, search, mining, and comparisons. Glenn Ammons presented the Mirage virtual machine image library: while the hypervisor provides an unstructured VM image, Mirage presents the image as more structured data, allowing faster image deployment by indexing the file system structure.

The task of converting all images from Xen to KVM while using RC2 (Research Compute Cloud) was the first use case presented. This is inherently an iterative process—find a bug, fix it, and try again. The version control features of Mirage are especially useful in this scenario, allowing for rollback and comparisons for debugging. The second use case involves employing the IBM Workload Deployer, which deploys images to machines in an enterprise while providing an enterprise configuration in an all-in-one-place view. Its customers typically have complicated workflow environments: for example, the OS team creates an image while the middleware team installs the middleware to create the product consumed by the application team where apps are installed. When the OS team updates the OS, Mirage allows computing the difference between the different versions, among other things, and creates new middleware automatically.

During the Q&A, Glenn elucidated the fact that manual labor is still needed, even with the automated version controlling and patching, although it is reduced to just verifying the automated result. Someone asked about policies to deal with concurrent changes to the images: for example, when the middleware team changes the system configuration as a result of their work and the OS team upgrades the version, their change might conflict with the current configuration. Glenn clarified the need to verify the results, as the method employed is opportunistic in nature.

Accelerating the Cloud with Heterogeneous Computing

Sahil Suneja, Elliott Baron, Eyal de Lara, and Ryan Johnson, University of Toronto

Elliott Baron presented the idea of leveraging the AMD Fusion kind of heterogeneous processors, which combine the GPU and CPU on the same chip, for accelerating and offloading cloud management tasks at the hypervisor level. The on-chip architecture allows low-latency memory access to the GPUs, overcoming the traditional bottleneck of access over the PCI bus.

Various use cases were presented—memory scrubbing, memory deduplication, memory compression, virus scanning, batch page table updates, etc. A case study of hashing-based page sharing was presented, indicating significant speedups over the CPU versions, as expected. Even with the Fusion architecture, memory copying between CPU and GPU was exposed, even though the two cores share memory. Also, hardware management for sharing the on-board GPUs between guest VMs and the hypervisor was discussed. The idea of incorporating time and space multiplexing was proposed.

Chris Colohan from Google asked whether GPUs could perform I/O operations on the storage stack. That could enable accelerated scrubbing of hard drives. The need for data to be in memory before the GPUs could access them was flagged as the hindering factor by Elliott, who acknowledged the benefits of the proposed idea.

Ion Stoica (UC Berkeley) suggested keeping data compressed in memory with on-demand decompressing. Elliott had already hinted at this in his talk.

Matei Zaharia (UC Berkeley), who hadpresented his work on OS for datacenters in the pre-lunch session, asked about the use cases of Mark's work. In Mark's opinion, the main use case is to actually detect the root cause of your problem. This is because latencies can be very different even when datacenters are running identical code. In this case, his technique allows one to try different hypotheses and figure out what's the most likely reason for the variability.

Chris Colohan from Google asked a question combining the ideas from two of the talks—monitoring and GPGPU computing. He wondered if GPGPU could be used for monitoring applications—security monitoring and intrusion detection. The other speakers answered yes, but Elliott, while agreeing that the proposed idea seemed very logical, made a playful comment regarding Windows users not liking the antiviruses using their CPUs, and hence using GPUs to offload the CPU for this task made even more sense!

John Wilkes from Google wondered whether all this could be generalized. What could be present in a general framework for monitoring and what would the standard libraries offer? Chit-Kwan's opinion was that of a datacenter-wide bus with APIs at the bus level and no higher, where every status stream would be represented by a type that could be published along with the granularity. Elliott said that his work was not specifically in the monitoring game, but he mentioned one important feature for his general framework—to keep GPU interaction out of the hypervisor but in dom0 space.

Michael Kozuch from Intel put forward an open-ended question—the importance of performance and its ranking in the hierarchy of scale, reliability, security, etc. Elliott rated security above performance, while Chit-Kwan considered performance more important than monitoring. Glenn's view was that more important than performance is the visibility into performance. Mark considered performance as being of primary importance.

John Wilkes raised a debatable issue by stating that emphasis on performance should be to a lesser degree in academics. In his opinion, performance is relatively easier to add, and there are much more interesting things to be found outside the performance space. While Glenn agreed with this notion, Elliott clarified that his project is not so much about getting some performance points—it's about utilizing the new architecture that's hitting the markets. In Mark's view it depends on the problem being addressed—for example, number crunching performance is more important than serving user requests. Eyal de Lara (University of Toronto) said that people are still working in the performance space and it is important for the datacenters—not that a particular optimization adds some small percentage improvement, but definitely that an idea can reduce datacenter size by half. If the return is a small delta improvement, then the original comment made sense, but from there to assuming that we have all the performance we need and we don't really need to improve on it is not correct.

Byong Gon continued on the last discussion and inquired about predictable performance. Mark jokingly answered by contradiction—he was more comfortable in answering what could make performance unpredictable. He believed there was no single answer to the original question—perhaps having sufficient resources. In his opinion, consistent performance was more important. Going back to the last discussion, he agreed that 10% performance improvement was not very important, but 10x was definitely important.

Michael Kozuch put forward the final question, which dealt with hardware innovation possibilities. Elliott believed heterogeneity was an important step forward, with network processors and FPGAs finding their way onto chip in the future. Glenn lightheartedly routed the question to the Google guys to put forward their demands and requirements to the academics, with reference to the panel discussion on day one.

Cloud Computing and Data Centers (Joint Session with ATC)

See the USENIX ATC '11 reports for a report on this session.

Invited Talk (Joint Session with ATC)

Helping Humanity with Phones and Clouds

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

See the USENIX ATC '11 reports for a report on this session.

3rd USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '11)

Portland, OR
June 14, 2011

Panel

Big Data, No SQL, Big Problems, No Worries

Moderator: Margo Seltzer, Harvard School of Engineering and Applied Sciences
Panelists: Mark Callaghan, Facebook; Andy Twigg, Acunu and Oxford University; Andy Gross, Basho and Riak; Alex Lloyd, Google

Summarized by Dutch Meyer (dmeyer@cs.ubc.ca)

The four panelists each brought lessons and observations drawn from their industrial experience in tackling large-scale data storage and processing.

Mark Callaghan, who leads the MySQL team at Facebook, spoke first about NoSQL, describing how the need for multi-master replication and write-optimized storage was pushing SQL in new directions. Rather than literally providing no SQL, Callaghan would actually prefer what he termed "SomeSQL." He described a collection of rich features per node that would help him in practice, including secondary indexes, multiple operations per transaction, non-indexed predicates and data types, 10,000 queries per second (at one IOP each query), and high concurrency access to high contention data.

In describing other problems he'd like to see addressed, Callaghan stressed the challenge of reconfiguring storage in a production environment. Often a storage system cannot afford to restart in order to apply changes. How does one coordinate schema changes when applications must specify an access path to the data and the people who wrote the apps are no longer available? In addressing write-optimized storage, Callaghan first explained its many benefits, such as lowering the demands of random writes, simplifying hot backup and compression, and possibly making redundancy cheaper. At the same time, this write optimization introduces problems with increasing random reads and requiring file scans for compaction. The latter problem might be masked by merging backup and compaction into a single scan. In closing Callaghan stated that the world has a surplus of clever ideas but that the challenge, and our focus, should be getting things into production. He advised running a server before writing a new one, and investing heavily in support for monitoring, debugging, and tuning.

Andy Twigg, a research fellow at Oxford and founder of Acunu, has been working to optimize the kernel stack for big storage. He began by questioning the definition of big data. Today, the term might be used incorrectly to mean scale-out, Web-scale, or NoSQL, but some of the biggest data problems actually use SQL and some of the best-known NoSQL databases (such as CouchDB) don't scale out properly. To Twigg, managing big data is the process of finely balancing several huge and unstoppable forces. He drew an analogy to surfing, where harnessing the force of a wave requires knowing how it will behave; trying to work against it is futile, if not dangerous.

There are three fundamental forces involved in big data, and the first is the storage technologies being employed. One hundred dollars today can purchase 60 GB of flash storage which can deliver 40,000 operations per second, or 2 TB of magnetic disk that delivers 100 operations per second. To approach big data problems, one must devise algorithms to exploit these traits. However, this is not always straightforward, as Twigg demonstrated with a graph showing a precipitous drop in SSD performance as the device is filled near capacity. A second fundamental force is the workload, which designers must understand and characterize. Big data workloads often show high rate update and large range queries on data items of varying sizes and value. Naive algorithms and abstractions are probably suboptimal for any particular workload. Twigg's last fundamental force is scale. A social media startup might choose to pay a storage provider for specialized hardware, or to purchase many more smaller machines and scale them out. Twigg pointed to the existing literature on distributed systems as a useful resource.

Andy Gross from Basho next reflected on the state of distributed system research, and open questions for the future. Gross declared that big data is boring; the interesting problems are really distributed systems problems. Web developers today have moved from arguing over frameworks and APIs to discussing Paxos and the CAP (Consistency, Availability and Partition-tolerance) theorem.

Several factors have led to this renaissance of distributed systems. First, the cloud has simplified cost and scalability. Venture capitalists often demand that new companies use EC2 rather than scaling up their own services. Second, customers expect more availability for products, which is changing business requirements. Third, workloads are changing. Log data that would previously have been discarded is now being harnessed as a revenue generator. These advancements are disruptive to the traditional view of storage.

However, Gross pointed to several advances that show progress in the field. The Bloom language from Berkeley can perform consistency analysis and verification of order of independence. Gross's own work on Riak Core provides a generic distsys toolkit that enables experimentation and rapid prototyping. Stasis and Leveldb offer modern storage engines that are permissively licensed. Still, there are important problems left to solve. Global replication is still largely impractical, and the operation of increasingly complex systems is difficult. Formal verification methods could be developed to the point of providing some assurance of system correctness. Finally, the nuances of virtualization and the cloud likely change the assumptions underlying our systems, but we have yet to fully understand them.

Alex Lloyd from Google described how many aspects of SQL have been discarded with the move to NoSQL simply because they are hard to implement. He argued that it is time to determine what useful features an application team needs and to figure out how to provide them. For example, transactions are important to minimize the time application writers spend reasoning about concurrency concerns. Without this feature, application developers must each reason about concurrency and consistency above the storage layer. Another traditional database feature, joins, is extremely useful, despite being very difficult to scale. Application developers also need to be able to express queries concisely, rather than repeatedly querying the storage server. Other issues he brought up were compaction, conflict resolution, and performance isolation. Lloyd hopes that ultimately "we can have our features and scale them too." However, we need a scalable programming model that gives predictable performance, and unified data repositories. Today each group works with their own island of data, but they need to be able to come together in a tightly coupled system.

Panel Chair Margo Seltzer asked about the relative merits of scaling up (on a single host) versus scaling out (to many hosts). Initially, several panelists saw no difference, but as the discussion progressed some did emerge. For many, scaling up to a single very high-powered machine is an option. Furthermore, there are some differences in how scaling occurs. Paxos, for example, is not appropriate for a single host, but there are reasons to run multiple SQL servers on a single node. Peter Desnoyers (Northeastern University) asked each panelist for the most important reason to scale out. Twigg and Gross agreed on fault tolerance, while Lloyd replied that scaling up could only take a system so far. Twigg reminded the audience that for most people, there are limits to the size of a database, after which more scaling is not necessary.

Erik Riedel (EMC) recalled a comment from Alex Lloyd: "The complexity has to go somewhere [in the storage stack]." Riedel asked if we could use layering to remove complexity at the source and, if not, wondered where the complexity should go. Lloyd believes in finding common operations and integrating them into storage—bringing legal discovery tools into storage, for example, where they may also be useful to other applications. Callaghan and Gross also saw potential to hide asynchronous replication and elements of relational database in storage. Albert Chen (Western Digital) asked the panel to what degree they are concerned about power usage. Gross replied that he didn't even think about it. Callaghan explained that people who care about power are not usually close to the database servers, and Lloyd said that he was dubious about getting predictable performance from complex power-saving systems.

A Solid State of Affairs

Summarized by Luis Useche (luis@cs.fiu.edu)

Don't Thrash: How to Cache Your Hash on Flash

Michael A. Bender, Stony Brook University and Tokutek; Martin Farach-Colton, Rutgers University and Tokutek; Rob Johnson, Stony Brook University; Bradley C. Kuszmaul, MIT and Tokutek; Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P. Spillane, and Erez Zadok, Stony Brook University

Rick Spillane introduced a new probabilistic data structure, similar to Bloom filters, especially designed for solid state storage. Their work was prompted by the infeasibility of fitting Bloom filters in memory for large storage systems. With Quotient Filters, a replacement for Bloom filters, the idea is to hash the elements and use part of the key to index in an array and store the rest in the array location. This new structure has the same properties as Bloom filters, with the addition that they can be merged into bigger Quotient Filters. This last property is key to implement Cascade Filters, a Bloom filter replacement especially designed for flash devices. Cascade Filters maintain a set of Quotient Filters (one in RAM and the rest in disk) organized in a way to allow lookups bounded by log(N). Cascade Filters allow writing sequentially to flash devices while still maintaining fast lookups. With this technique they achieve 40x faster insertions and 3x slower lookup throughput compared to Bloom filters.

How will collisions in the insertion operation affect lookup times? They ensure with their technique that sequential scans to QF clusters will not be more than log(N) where N is the number of elements in the cluster; moreover, this worst case is an unlikely event. James Lentini asked whether the data structure they designed was resilient to power loss or crash. Rick answered that they can have atomicity by using a combination of COW and journaling that is not yet implemented.

Onyx: A Prototype Phase Change Memory Storage Array

Ameen Akel, Adrian M. Caulfield, Todor I. Mollov, Rajesh K. Gupta, and Steven Swanson, University of California, San Diego

Ameen Akel introduced a new prototype of SSD implemented using Phase-Change Memory (PCM), an emerging byte-addressable persistent memory. This technology takes advantage of the difference in resistance when molecule phases are changed. Current PCM outperforms flash technologies, especially in reads, making it suitable for SSD implementation. Its projected performance is three orders of magnitude faster than current SSDs. Their real data shows different results than simulators, making the case for a prototype PCM-based SSD for better estimation of performance results.

Onyx was compared with FusionI/O, a high-end SSD, and showed consistently better read throughput as request sizes increased. Ameen pointed out that PCM does not require complex FTL logics that significantly slow down flash technologies. For writes, Onyx showed better performance only in small request sizes. Ameen attributes this to the more mature flash technology heavily optimized for writes since its conception. When they ran Berkeley DB benchmarks, Onyx did not show exceptional gains. In conclusion, Ameen emphasized the potential of PCMs compared to flash due to its simplicity and because of the absence of FTL.

Andy Twigg asked whether this technology, given that it is byte addressable, will eliminate the problem of creating large requests to obtain better performance. Ameen said that these devices eliminate this problem, making the interaction between the application and the backing stores easier. Peter Desnoyers was curious what lessons they learned from constructing an experimental device like this. Ameen said that developing Onyx was difficult but worth it, as it renders better results than the simulations used for previous studies. Irfan Ahmad was concerned about what problems PCM technology might have before it can be commercially available. Ameen said that the main concern with PCM technology is whether it will be able to scale in size as flash has been doing in recent years. Irfan asked what interfaces other than ROM are currently available. Ameen suggested that DIMM interfaces will be a big step forward because they will make PCM easier to use. Any ideas for future work? It was important to investigate better interfaces and the impact this technology would have in application performance. Peter wondered whether they felt confident that PCM will replace flash as the choice of SSD. Ameen expressed high confidence in PCM's future.

SSD Characterization: From Energy Consumption's Perspective

Balgeun Yoo, Youjip Won, Seokhei Cho, and Sooyong Kang, Hanyang University, Korea; Jongmoo Choi, Dankook University, Korea; Sungroh Yoon, Korea University, Korea

Youjip Won stressed the importance of understanding the internals of SSDs. He mentioned that disk characterization has been done for decades and has allowed the design and implementation of many of the important improvements available today. Now the question is, what measurements can be used to characterize the SSD? Given the electronic nature of SSDs, they decided to characterize based on energy consumption. SSDs have multiple channels to communicate with the NAND chips. SSD logic usually maximizes parallelism by using as many channels as possible to increase performance. In this paper they focused on how the channels in the SSD are used to service every request.

They started the characterization by measuring the power consumption of the SSD while increasing the request size. They found peaks that indicate how many channels are used. Moreover, the duration of the peaks give an estimate of how long the channels are activated to service the request. Just for comparison, 16 KB and 32 KB request sizes showed same duration but different peak sizes. This indicates an increase in the number of channels involved when the request size is doubled from 16 KB to 32 KB. On a different example, 256 KB and 512 KB request size showed the same power consumption but with 2x difference in the duration of the peak. Youjip also showed the tradeoffs between parallelism and the peak power consumption of the device: with higher parallelism comes a higher peak power consumption. High peaks cause problems such as supply voltage drop, signal noise, and blackout. For this reason, they propose a technique called Power Budget that will maximize the parallelism as long as the peak power is held below the specified maximum. Youjip ended by highlighting the usefulness of power consumption in characterizing SSDs.

How willing are manufacturers to disclose internals of SSDs? Youjip said that more important than the number of channels, which is usually available, companies should standardize the way the power is reported to upper layers. Theodore Ts'o asked why the peak power consumption was more important in the paper when it is not as representative as the area under the curve. Youjip replied that peaks were relevant because they can accidentally turn off the machine if not controlled. Irfan Ahmad asked whether additional features could be extracted with this technique. Youjip said that unfortunately there are some SSDs that do not show clear behavior, limiting the scope of the power consumption characterization. Irfan wondered whether information from the FTL could also be extracted. Youjip replied that FTL is complex and they do not know how it works. However, they can use comparisons to find which type of FTL is more energy efficient. Finally, Peter Desnoyers wondered whether this technique could leak information that was not intended to be public. Youjip thought there was no relation between the power consumption and the information in the SSD.

Exploiting Heat-Accelerated Flash Memory Wear-Out Recovery to Enable Self-Healing SSDs

Qi Wu, Guiqiang Dong, and Tong Zhang, Rensselaer Polytechnic Institute (RPI)

Qi Wu predicted a bright future for SSDs, which are lowering their price while continuing to grow in size. Moreover, they are the perfect candidates for high-performance applications. Unfortunately, NAND flash chips, the most popular technology behind SSDs, suffer from a limited number of program/erase (P/E) operations, and that limits their lifespan. Interface traps are one of the causes for NAND flash failures. In previous work, researchers found that NAND flash can recover from interface traps, because they heal faster when heat is applied.

In this presentation, they proposed a new self-healing SSD architecture that recovers flash chips from interface trap failure. The basic idea is to include a small heater on every chip that will be used once the number of P/E cycles are close to the limit. Qi mentioned that while the heating process is in progress the data in the chip under recovery cannot be accessed. For this reason, they add one additional chip on every SSD channel, to be able to migrate the data before applying heat. While the backup operation occurs, the chips attached to the channel in use cannot be accessed. Qi said that they found a tradeoff between latency, backup time, and copying granularity: Faster and higher copying granularity leads to higher latencies.

They set up a multi-component simulator to evaluate their new architecture. In their simulation they found that their architecture can result in a fivefold increase in SSD lifespan. On the downside, this architecture could increase the latency of I/Os up to 15% compared to commodity SSDs.

Peter Desnoyers asked whether they have implemented a real prototype of this architecture. Qi said that they are relying on real implementations in previous works. Peter then asked what type of market will embrace this technology. Qi replied that any type of write-intensive application can benefit from this technology, since it will increase the life of their backing stores. Somebody asked why they did not try the experiment of heating a commodity SSD and checking its lifespan. Qi replied once again that they are relying on previous work. Moreover, Peter added that such an experiment will also heat the controller, which can ultimately damage the SSD.

A Coterie Collection

Summarized by Sejin Park (baksejin@postech.ac.kr)

Italian for Beginners: The Next Steps for SLO-Based Management

Lakshmi N. Bairavasundaram, Gokul Soundararajan, Vipul Mathur, Kaladhar Voruganti, and Steven Kleiman, NetApp, Inc.

Datacenters' increased system complexity arising from service automation needs causes low operational and management efficiency. Gokul Soundararajan laid out current datacenter trends: the move from a siloed to a shared world to improve resource utilization; increased configuration complexity (e.g., RAID level, dedup) in which the impact of combining these technologies is very hard even for the system administrator to understand; huge scale, which requires a large number of administrators to manage the datacenter; and dynamic applications, requiring administrators to understand dynamic resource requirements. To handle all this, the datacenter industry provisions for peak demand and hires a lot of administrators.

Gokul said the solution is automated management with service level objectives (SLOs). SLOs are specifications of applications' requirements in technology-independent terms, and their attributes are performance, capacity, reliability and availability, and security and compliance. The MAPE (Manage, Analyze, Plan, Execute) loop is used to achieve automated management. However, SLOs are slow to be adapted because, among other reasons, it is difficult to specify SLO requirements. He suggested focusing on process, not product, through the use of pre-defined SLOs (Qualified SLOs) as a way to manage various systems simply and reliably.

Someone asked how Qualified SLOs provide support if someone's system isn't working properly. Gokul emphasized that customer support is one of the benefits of Qualified SLOs.

In Search of I/O-Optimal Recovery from Disk Failures

Osama Khan and Randal Burns, Johns Hopkins University; James Plank, University of Tennessee; Cheng Huang, Microsoft Research

Traditionally, systems are made reliable through replication (easy but inefficient) and erasure coding (complex but efficient). Because storage space was a relatively expensive resource, MDS codes were used to achieve optimal storage efficiency with fault tolerance. However, time and workload have changed and the traditional *k-of-n* MDS code would require k I/Os to recover from a single failure. Osama Khan addressed this problem and suggested a new way to recover lost data, with minimal I/O cost, that is applicable to any matrix-based erasure code.

Osama claimed that enumerating all decoding equations is not an easy job and finding a decoding equation set with minimal I/O is the challenge. He transformed this into a graph problem and used Dijkstra's shortest path algorithm. He also explained that GRID code is an I/O-efficient recovery code. GRID code allows two (or more) erasure codes to be applied to the same data, each in its own dimension. With the GRID code, the author could combine the STAR (for high redundancy) and Weaver (for low I/O) codes and make an I/O-efficient code.

Someone asked about their approach to the variety of other erasure codes. Osama recognized the presence of alternative erasure codes besides the traditional Reed Solomon code and said the technique they used is applicable to all types of erasure codes that can be represented in matrix form. Someone asked whether CPU utilization had been considered, since erasure coding is CPU-intensive. Osama replied that CPU utilization was not part of their study; they focused, instead, on measuring I/O for recovery.

ViDeDup: An Application-Aware Framework for Video De-duplication

Atul Katiyar, Windows Live, Microsoft Corporation, Redmond; Jon Weissman, University of Minnesota Twin Cities

Atul Katiyar talked about the kinds of redundancy in large-scale storage systems and said that the redundancy is managed if the storage system is aware of it and replication is performed for specific goals such as fault tolerance or improved QoS. The redundancy is unmanaged when the storage system is unaware of it and it merely acts as an overhead on the storage system. The system views redundancy as an identical sequence of bits. However, the application-level view of redundancy is a little different, defined as a metric that gauges redundancy at the content level with the flexibility to define and hence tolerate noise in replica detection as dictated by the application. Atul gave examples of videos encoded in different formats, frame resolution, etc.

Atul said that large-scale centralized Web storage is an emerging trend and there is a significant degree of unmanaged redundancy in such storage systems. Application-level redundancy can significantly reduce the storage space by its deduplication. The ViDeDup system is an application-aware framework for video deduplication, and it detects similarity among contents. The framework provides application-level knobs for defining acceptable noise during replica detection. He enumerated various aspects in which near-duplicate videos differ. They leveraged the research of the multimedia community in adapting, modifying, and integrating existing approaches for video similarity detection into the framework. In contrast to system-level deduplication, in ViDeDup the choice of which of the two duplicate replicas to store is not trivial. They propose a centroid-based video deduplication approach, where the centroid video is the representative video of good quality in the cluster, against which remaining videos of the cluster are deduped. Atul presented an algorithm for centroid selection which balances the tradeoff between compression and quality within the cluster.

Someone asked whether this is lossy compression and how this technique compares with other video compression techniques. Atul said it uses lossy compression; standard mpeg compression looks intra-file, while this compression seems more inter-file. In some interesting key contexts, such as in-cloud service wherein uploading video is for dissemination, it might make sense to tolerate loss. Peter Desnoyers expressed doubt about whether this compression can really work. Atul proved it with his demo video of two videos having the same basic nature, compressed to result in a final video. Someone asked how long it tookto process the data set. Atul said compression of 1017 videos took two hours, but the system-level deduplication processed in 15–20 minutes.

A River of Data, More or Less

Truly Non-Blocking Writes

Luis Useche, Ricardo Koller, and Raju Rangaswami, Florida International University; Akshat Verma, IBM Research—India

Exposing File System Mappings with MapFS

Jake Wires, Mark Spear, and Andrew Warfield, University of British Columbia

Stratified B-trees and Versioned Dictionaries

Andy Twigg, Andrew Byde, Grzegorz Miłoś, and Tim Moreton, Acunu; John Wilkes, Google and Acunu; Tom Wilkie, Acunu

No reports are available for this session.

Invited Short Talks and Wild Ideas

Summarized by Dutch Meyer (dmeyer@cs.ubc.ca)

Using Storage Class Memory for Archives with DAWN, a Durable Array of Wimpy Nodes

Ian F. Adams and Ethan L. Miller, University of California, Santa Cruz; David S.H. Rosenthal, Stanford University

Adams made the argument that the research community should consider Storage Class Memory (SCM) as an archival storage medium. SCM refers to a class of technologies, including Flash, PCM, Memristor, and others, that have a high cost-to-capacity ratio, but offer much higher storage performance than magnetic disk, especially for random reads. These characteristics are not usually associated with archival storage, but Adams pointed out several ways in which the total cost of ownership for SCM may actually be lower than magnetic disk.

He began by reviewing current technologies for archival storage. Hard drives have a lower initial purchase cost, but magnetic disk is mechanically complicated. Large racks of disks are heavy to the point that they may require reinforced flooring. They also are vibration and shock sensitive and require a great deal of power to operate. Tape is even denser, requires more maintenance and cleaning, and has poor random access performance. It also doesn't scale well, in that a cost-benefit analysis may show tape-based storage to be a poor value in surprising ranges of storage capacity. Alternatively, Adams imagines an array of simple low-power SCM-based system boards he calls DAWN. Such an architecture could be scalable, power-efficient, and largely self-managing, as each unit would be responsible for its own integrity checks, and be replaceable. Adams suggested that this approach may provide a lower total cost of ownership, but he saidthat more investigation is needed to characterize current and future SCM and to do the necessary cost analysis.

Erik Riedel (EMC) asked whether backups could be left on the shelf without any maintenance. Adams replied that they see a wide variety of customer workloads, ranging from write-once, read-maybe to high-frequency scrubbing and error checking. Riedel suggesting looking closely at disk drive technology, as the failure rate for a disconnected drive likely approaches that of disconnected SCM. Peter Desnoyers from Northeastern joined the presenter in calling for more research into the effects of temperature on archival storage and into whether even the best device would survive for very long lifetimes. Several attendees asked what range of devices should be considered, and the answer covered a broad range of archival options, including S3 and paper printouts.

Principles of Operation for Shingled Disk Devices

Garth Gibson and Greg Ganger, Carnegie Mellon University

Less than half of the audience for Ganger's talk had heard of shingled disks. This new technology will lead to higher capacities in magnetic disks, but not without introducing some new performance effects. Instead of writing each track with gaps in between, in a few generations disks will write tracks that overlap. One consequence is 1.5 to 2.5 times more density, but it also means that one cannot overwrite old data without erasing newer data. Firmware could theoretically hide this behavior, as is done in flash devices, but the resulting read-modify-write cycle is 1,000 to 10,000 times longer than that of Flash. Ganger believes that this would be impractical, and instead proposes to explore an interface to let the software above the device manage its peculiarities. Such software could minimize in-place modification through a log structure, very large block sizes, or a hierarchy of performant hardware, such as flash or PCM. Ganger closed by stressing the new questions that the introduction of this technology will raise. Researchers must determine the right interface for this storage, how best to exploit features and dodge costs, and what role firmware will play in managing the device.

Session chair Anna Povzner (IBM Research) asked if shingling is an appropriate capacity/performance tradeoff. Ganger replied that this is an inevitability. While we may have the choice of not using shingled regions of the disk, that would be abandoning the order-of-magnitude capacity increase. Geoff Kuenning (Harvey Mudd) asked if there was any hope of getting to a place where we don't have to keep rewriting systems for each new technology. Ganger argued that there is enough difference between devices that it's not clear that we'd want to generalize our storage software.

Albert Chen (Western Digital) said that they had considered many of the options discussed in the session, and that in the short term, drives will not need much support from the software above them. He and the presenter agreed that hints may be a method of striking the right balance between compatibility and specialization.

Panel

Storage QoS: Gap Between Theory and Practice

Moderator: Carl Waldspurger

Participants: Greg Ganger, Carnegie Mellon University; Kaladhar Voruganti, NetApp; Ajay Gulati, VMware; Ed Lee, Tintri

Summarized by Dutch Meyer (dmeyer@cs.ubc.ca)

Moderator Carl Waldspurger posed questions for both the panelists and the audience. First, do users and administrators want to specify QoS in terms of predictable performance, or service level objectives (SLOs)? Even if we had an answer, it's not clear we'd know how to quantity QoS; we might measure latency, IOPS, transactions per second, or other metrics. Enforcing these metrics is also challenging, because performance isolation is challenging and the storage stack is complex. There's also a tension between delivering QoS and performance.

Ajay Gulati (VMware) took the position that storage QoS will be pervasive in the next five years. His vision is driven by virtualized environments giving rise to the need for per-VM controls and will be made possible through the deployment of SSD. Current systems use deadline-based scheduling or CFQ, or one of the virtualization-specific schedulers such as SFQ or mClock. Mostly, these schedulers are based on proportional allocation, which doesn't give applications much of a guarantee. The lack of guarantees is because of the efficiency versus fairness tradeoff, because the metric isn't clear, and because customers are scared off by worst-case performance numbers. To solve these issues, he proposes that QoS be defined in terms of latency, perhaps up to a specified number of IOPS. To meet this latency bound, arrays of SSDs large enough to fit the working set of applications could offer reliable performance.

Kaladhar Voruganti (NetApp) reframed the QoS problem in terms of SLO. His model consists of three parties: a storage vendor who delivers features, a storage provider who creates a service catalog with quality tiers, and a storage subscriber who orders a particular level of service. Managing an SLO is preferable for a subscriber because they understand application requirements, not the effects of storage system latency. Still, there are problems with current approaches. SLO specification remains difficult: the complexity of storage makes management models hard to create and must be made to cross management layers. Kaladhar sees promise in combining proactive approaches to SLO management, such as application-specific templates for performance, with reactive approaches like hybrid flash and disk systems that can minimize the impact of an incorrectly specified SLO.

Ed Lee (Tintri) offered another definition of QoS: it avoids user complaints without spending a lot of money or time. He pointed out that users don't actually notice fairness, but they do notice performance inconsistency, and will complain about slow-downs. Fairness, consequently, is less useful than performance consistency. Lee's presentation proceeded to point to a number of current problems in QoS. First, the technologies are all built by different vendors, and each will develop their own notions of QoS. It's appropriate to build QoS at each level, but mechanisms need to be able to work together. Storage systems are large and complex, with many components that can affect performance. Furthermore, constraints must be specified in aggregate. Finally, Lee made the case for building rational systems that have no performance cliffs, are consistent over time, and provide a simple, transparent model of their behavior.

Greg Ganger's presentation explored the gap, which he referred to as a "chasm," between QoS theory and practice. First, he explained how the theoretical assumption of starting with a clear SLO specified by a customer is flawed. Humans, even experts, are bad at expressing their needs in terms of performance. In practice, one usually guesses by choosing from broad tiers of quality, and reacts to performance problems as they arise. Theory might also dictate that workloads should be admitted based on demand, but this assumes that demand on the system is predictable. In practice, this kind of stability can only be seen in a very large window of time. Since workload varies both in intensity and in characteristics like locality, it is very difficult to predict what the actual system demands will be for any workload.

To make matters worse, one might assume that device load could be determined by the workload. However, in practice, the observed load given a device and a workload is often not repeatable. There are many sources of variability, including interference between workloads, internal maintenance, and device retries, where the disk initially fails to complete a request. Even differences between two devices of the same apparent make and model can skew workload results. The good news for QoS advocates is that no one is expecting perfect results. If the current approach is to start with one of a small number of service tiers and then respond to complaints, perhaps we can make that process faster and easier.

The first question addressed the different views on what metric QoS should use. Lee said that the chief difficulty is that different applications need different metrics. Youjip Won (Hanyang University) noted that the more general QoS problem has been around for a decade, but that storage has a much larger range of variability. He asked whether turning to SSD to solve this problem was just another form of over-provisioning. Several members of the panel acknowledged that flash brings new problems, but it does make some issues (such as random read latencies) easier to solve. The overall customer experience can be improved in embracing SSD.

This conversation sparked a discussion about SSD latencies. According to Lee, SSD can have latencies much higher than disk under some conditions, although Gulati suspects that only lower-cost SSDs would exhibit this behavior. Peter Desnoyers (Northeastern University) confirmed that he had seen an iSCSI interface time out while connected to an SSD under a stress test. He wondered what level of performance isolation is ultimately possible. Lee and Ganger explained that nearly perfect performance isolation is possible, even under multi-tenancy, by giving different applications different time quantums, but in practice we want more than isolation. Goals like cost-efficiency, high performance, or performance reliability interfere with isolation.

Desnoyers also asked if SLO specifications could be made for each application or at each level of the storage stack. Lee responded no. Even a very well designed system would specify aggregate SLOs, ideally automatically so that more time could be spent managing user expectations. Alex Lloyd (Google) thought that it seemed it would be years until we could hope to describe SLO for a single user/single tenant environment, so perhaps it would be better to push for wider interfaces with more control. Lee agreed that vendors should expose more of their system state and suggested they could provide hooks with reasonable defaults that naive customers could ignore. He hoped that this would lead to consensus around a standard model. Ganger was more pessimistic about companies agreeing on standards.

Irfan Ahmad (VMware) noted that someone shipping through FedEX gets several service options, and you get an SLO that you can characterize, manage, and buy insurance around. He wondered if we should stop focusing on the tails of the performance curve and instead focus on the 80th or 90th percentile. Ganger, Lee, and Ahmad discussed some of the potential benefits of keeping requests in-buffer to smooth the variability in performance. It would make performance predictable and also give administrators the ability to easily increase performance in response to customer complaint.

3rd Workshop on I/O Virtualization (WIOV '11)

Portland, OR
June 14, 2011

I/O Virtualization Architectures

Summarized by Sejin Park (baksejin@postech.ac.kr)

SplitX: Split Guest/Hypervisor Execution on Multi-Core

Alex Landau, IBM Research Haifa; Muli Ben-Yehuda, Technion Israel Institute of Technology and IBM Research Haifa; Abel Gordon, IBM Research Haifa

Machine virtualization lies at the foundation of many datacenters and cloud computing, but its use is often limited due to unacceptable performance overhead. Muli Ben-Yehuda and his co-authors argue that this overhead is inherent in Popek and Goldberg's "trap and emulate" model for machine virtualization. In that model, when a guest operating system executes a privileged instruction, the instruction traps, causing the core to exit from the guest context and switch to the hypervisor context. The hypervisor then emulates the trapping instruction and switches back to the guest OS. The overhead of machine virtualization comes from these exits.

To achieve the holy grail of zero-overhead machine virtualization, the authors propose the SplitX architecture, where the guest and the hypervisor each runs on a dedicated set of cores. Exits are replaced by inter-core messages. When the guest executes a privileged instruction, the guest's core sends a message to the hypervisor's core, which handles the exit and sends a message back. Such an architecture replaces the costs of an exit completely, replacing them with the cost of fast inter-core communication, which is an order of magnitude smaller. It also enables removing some or all of an exit's synchronous cost, since the hypervisor can handle certain types of exits while the guest continues running. Muli presented an analysis of a networking workload which incurs a 35% slowdown with current methods. With SplitX, the same workload incurs a slowdown of less than 1%. SplitX requires some hardware support for running unmodified operating systems, but they are implementing SplitX functionality on current hardware.

Flash Memory Performance on a Highly Scalable IOV System

Peter Kirkpatrick, Adel Alsaadi, Purnachandar Mididuddi, Prakash Chauhan, Afshin Daghi, Daniel Kim, Sang Kim, K.R. Kishore, Paritosh Kulkarni, Michael Lyons, Kiran Malwankar, Hemanth Ravi, Swaminathan Saikumar, Mani Subramanian, Marimuthu Thangaraj, Arvind Vasudev, Vinay Venkataraghavan, Carl Yang, and Wilson Yung, Aprius, Inc.

No presentation was made for this paper.

VAMOS: Virtualization Aware Middleware

Abel Gordon, IBM Research Haifa; Muli Ben-Yehuda, Technion Israel Institute of Technology and IBM Research Haifa; Dennis Filimonov and Maor Dahan, Technion Israel Institute of Technology

Abel Gordon said that virtualization overhead is still a problem, due to the switching between the guest and the hypervisor. Because previous approaches to deal with this performance problem focused on the interaction between the hypervisor and the guest OS, there are still potential optimization issues in the application layer. These can be assigned to virtualization-aware middleware such as databases, Web servers, or application servers, thereby reducing virtualization overhead. The architecture he showed was simple to understand. In contrast to the traditional architecture, some I/O module—say, module C—in the middleware moved down to the hypervisor. Thus module C directly interacts with the middleware in the guest OS. By using the hypervisor-level middleware module, the virtualization performance is improved without any modification to the operating system, and because the author moved the module, the code can be reused. In the evaluation, they achieved about 5%–30% improvement. They have plans to apply this technique to other middleware and are also considering rethinking the middleware from scratch.

During Q&A, Abel Gordon explained that Oracle has a similar architecture. In Oracle, there is some kind of JVM and Java application that runs directly on the hypervisor without the OS. Actually what they did, interfacing between the OS and the middleware, created some kind of small OS. Someone asked whether the authors tried multiple guest scenarios. Abel said just the single desktop was considered.

Invited Talk

Data Center Challenges: Building Networks for Agility

David A. Maltz, Senior Researcher, Microsoft

Summarized by Henrique Rodrigues (hsr@dcc.ufmg.br)

David started his presentation giving a general view of the network usage in a datacenter. Using Bing as an example, he described how network-intensive applications such as data mining and search indexing algorithms, which are focused on improving user experience, can saturate the core of a datacenter network. In this scenario, if any device at the highest level of the network topology goes down, the result would be massive network congestion.

David argued that increasing network capacity does not solve the problem, because demand is constantly growing. To support his point of view, he used a network utilization graph that showed the demand growth on the same period that the network received some capacity upgrades. Having noticed this problem, his research group started to think about a different solution to solve the capacity issues.

David commented that, most of the time, datacenter resource utilization is between 10% and 30% of total capacity. To increase the return on investment (ROI), the datacenter needs to be agile, able to quickly expand and contract the pool of active resources dynamically, following user demands. However, today's datacenter networks are built using a tree-based topology and VLANs to isolate different layer-2 network domains, which restricts their ability to assign any service to any server. Each service is constrained to a single L2 domain. This network architecture is also the cause of poor performance isolation and high oversubscription ratios. Finally, traffic measurement results show not only that traffic patterns on datacenters are very different from Internet traffic patterns but that the datacenters' traffic matrices are highly volatile.

David presented VL2, whose main principles are randomized routing to deal with datacenter traffic matrix volatility; decoupling server names from locations to allow the assignment of any service to any server; use of existing technologies to make the solution deployable on today's devices; and use of end-hosts, which are programmable and have lots of useful resources. The two key actions performed by a VL2 network are the encapsulation of complete routing information on each packet, performed by each end host using the information stored on a centralized directory system, and the random traffic spreading over multiple paths using valiant load balancing and ECMP. David explained how the solution works on a given Clos topology and how VL2 is able to provide full-bisection bandwidth and high resilience in case of link failures. Evaluation results showed that VL2 achieves high throughput and good performance isolation between different services.

Is it possible to have more than one tenant on each physical server when the datacenter is using VL2? It is not possible, because the network performance isolation provided by the hypervisor is not as good as the performance isolation provided for CPU and memory. Is VL2 able to handle the incast problem? VL2 is not a solution and will try to keep the queuing at the edges of the network and as low queuing as possible inside the network. What is the overhead imposed on end hosts to encapsulate routing information on each packet? They didn't measure such overhead but it was very low, because only a few more instructions were added to the original code. Would you clarify how traffic flows between the datacenter network and the Internet? The logical view provided by a VL2 network is that all the devices of a datacenter (servers and external links) are connected to a single bus which provides full bisection bandwidth between any pair of devices.

Performance Management in IOV Systems

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

Revisiting the Storage Stack in Virtualized NAS Environments

Dean Hildebrand, Anna Povzner, and Renu Tewari, IBM Almaden; Vasily Tarasov, Stony Brook University

Dean Hildebrand started this session with an illuminating presentation on the difficulties virtualized systems create for makers of NAS (Network Attached Storage) systems. Makers of NAS systems go to great lengths to optimize their storage controllers and the protocols used to access them (e.g., NFS) for the I/O profiles of common workloads. More and more of these workloads, however, are now running in virtual machines. The introduction of an additional layer between the workload running in the virtual machine and the storage controller dramatically changes the I/O profiles seen by the controller. Whereas a workload running on bare metal would generate a mixture of metadata (e.g., create) and data (e.g., read and write) I/O requests to the controller, for example, the same workload running in a virtual machine would generate only data (e.g., read and write) I/O requests, because all of its block operations appear to the controller as file operations: reads and writes to the virtual machine's image file.

The bulk of Dean's presentation included detailed information on the I/O profiles of the same workloads running on bare metal and in virtual machines and the differences between them in a mixture of operations, in I/O sizes, and in sequential vs. random characteristics. Both bare-metal and virtual machine experiments were conducted using NFSv3. His conclusions were that virtual machine image files being read and written over NFS make NFS do "unnatural things" and that NFS and server file systems, as well as NAS makers, will need to adapt to these new workloads.

Wenji Wu of Fermilab asked whether the slides will be available after the workshop. Slides are available at http://www.usenix.org/events/wiov11/tech/.

Nested QoS: Providing Flexible Performance in Shared IO Environment

Hui Wang and Peter Varman, Rice University

Hui Wang said this paper is unusual for the workshop, in that it is fairly theoretical. It presents a quality-of-service model for virtualized environments ("nested" environments—not to be confused with nested or recursive virtualization). The nested QoS model offers a spectrum of response time guarantees based on the burstiness of the workload. Since a disproportionate fraction of server capacity is used to handle a small tail of highly bursty requests, the hope is that by providing a range of different response times which depend on the burstiness of the workload, overall server capacity could be reduced.

The model works by dividing incoming requests into different traffic classes, also called traffic envelopes, with each request's response time guaranteed as long as traffic remains inside the corresponding envelope. The model was evaluated on traces of block level I/Os from different workloads and appears to work well, leading to a large potential reduction in server capacity without significant performance loss. The results were all based on simulation, which led Himanshu Raj of Microsoft to ask Hui whether she had any idea what the runtime cost of implementing nested QoS would be. Hui answered that the cost is mostly in classifying requests into the different envelopes and is expected to be "very small."

Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks

Henrique Rodrigues, Universidade Federal de Minas Gerais (UFMG); Jose Renato Santos and Yoshio Turner, Hewlett-Packard Laboratories (HP Labs); Paolo Soares, Universidade Federal de Minas Gerais (UFMG); Dorgival Guedes, Universidade Federal de Minas Gerais (UFMG) and International Computer Science Institute (ICSI)

Suppose you have a server that runs virtual machines belonging to multiple tenants, who do not necessarily trust or cooperate with each other. All of the tenants share the server's network bandwidth. How can you provide network performance isolation to the different tenants, so that one tenant will not be able to overload the network at everyone else's expense? Henrique Rodrigues explained why neither TCP or UDP solves this problem, and that using rate-limiting is not enough, since it limits the senders but not the receivers. He then presented Gatekeeper, which satisfies the four practical requirements for a traffic isolation mechanism: scalability, an intuitive service model so that tenants can specify their requirements and understand what they are receiving, robustness against untrusted tenants, and the ability to trade off flexibility vs. predictability and make use of idle bandwidth. Gatekeeper works by limiting the transmit and receive bandwidth of each virtual machine (VM) through distributed agents running at the hypervisor layer on each of the servers. The Gatekeeper prototype is implemented in Open vSwitch on Xen/Linux using the Linux traffic shaping mechanism (HTB) for rate limiting.

Himanshu Raj of Microsoft asked about the difference between Gatekeeper and Seawall. The primary difference is that Gatekeeper divides available bandwidth between different tenants (where each tenant has multiple VMs), whereas Seawall only allocates available bandwidth between different flows. Wenji Wu of Fermilab asked how one knows to set the limits to the minimum bandwidth required by each service. Henrique replied that it is the tenant's responsibility to specify the bandwidth requirements of the applications.

Panel/Wild Ideas Session

Panel: Challenges for Virtualized I/O in the Cloud

Participants: Muli Ben-Yehuda, Technion—Israel Institute of Technology and IBM Research—Haifa; Alan Cox, Rice University; Ada Gavrilovska, Georgia Institute of Technology; Satyam Vaghani, VMware; Parveen Patel, Microsoft

No report is available for this session.

3rd USENIX Workshop on Hot Topics in Parallelism (HotPar '11)

Berkeley, CA
May 26–27, 2011

Day 1, Session 1

Summarized by Sabrina M. Neuman (sneuman@mit.edu)

Considerations When Evaluating Microprocessor Platforms

Michael Anderson, Bryan Catanzaro, Jike Chong, Ekaterina Gonina, Kurt Keutzer, Chao-Yue Lai, Mark Murphy, David Sheffield, Bor-Yiing Su, and Narayanan Sundaram, University of California, Berkeley

Bryan Catanzaro opened the HotPar '11 workshop with an examination of the problems plaguing GPU and CPU microprocessor platform comparisons. The two key conclusions of the investigation were that comparison results should be contextualized within a certain point of view, and that comparison results should be reproducible.

To illustrate the first conclusion, Catanzaro invoked the parable of the blind men and the elephant, where the men draw inconsistent conclusions because each collects data from a different point of view. He likened the men in the story to modern application researchers and architecture researchers, and suggested that comparison results need to be consistent with the point of view of their intended audience. The goals and values of application and architecture researchers were surveyed. Different points of view will require different sorts of comparisons, asserted Catanzaro. For example, large applications are useful for application researchers and micro-benchmarks are useful for architecture researchers.

To realize reproducible comparison results, as suggested by the second conclusion of the paper, Catanzaro argued that more detail must be presented with comparisons. Researchers should avoid making absolute claims about the superiority of one platform over another unless a full architectural study is performed. The structures of algorithms and data sets must be explained. The descriptions of the platforms being compared must be explicit. Catanzaro made a plea for researchers to practice good science by providing full details in their microprocessor platform comparisons, insisting that bad comparisons are holding back progress in the field.

Most questions centered on benchmarks as a means of comparison. Mike McCool (Intel) asked if there were any existing benchmarks that made for fair comparisons according to this work. Catanzaro replied that there are some good low-level benchmarks, but that micro-benchmarks are less useful than full applications. Dave Patterson (UC Berkeley) wondered if asking for reproducibility from cloud computing would be too restrictive, since it might require having to run on some particular cloud every time. Catanzaro replied that it would still be a good idea, but allowed that it would be great for the application researchers and difficult for the architecture researchers. An audience member asked what would be considered "cheating" for benchmarks. Catanzaro replied that the important thing is that the results are reproducible. Several audience members asked for Catanzaro's opinion about several particular benchmark suites. Catanzaro maintained that the conclusions of his presentation set the standard for good comparisons: point of view must be considered and reproducibility is essential, which means there must be sufficient explicit detail provided.

RADBench: A Concurrency Bug Benchmark Suite

Nicholas Jalbert, University of California, Berkeley; Cristiano Pereira and Gilles Pokam, Intel; Koushik Sen, University of California, Berkeley

Nicholas Jalbert presented RADBench, a suite of benchmarks containing 10 concurrency bugs found in large open-source software applications such as Mozilla SpiderMonkey, Apache Web Server, and Google Chromium Browser. Concurrency bugs are plentiful and painful to fix, asserted Jalbert. They are growing ever more commonplace, and they take a long time to diagnose and repair. To facilitate concurrency bug research, RADBench presents concurrency bugs "in the wild" by providing full snapshots of large buggy code and scripts to run the code.

Concurrency bugs have been difficult to track down and mine from old code sources, and tricky to recreate on demand. Inputs and thread scheduling behavior were often insufficient to conjure the bugs. Environmental factors such as interrupts, random numbers generated, and communication socket delays were critical to buggy conditions. Overall, the concurrency bugs did not fit into neat categorizations or classical patterns, making them hard to understand and reproduce. Concurrency bugs break the traditional programming paradigm; they are reactive and wacky.

Deterministic record and replay bug reproduction requires cumbersome overhead. Lightweight bug reproduction would be desirable, but remains an open problem. One approach is to record fewer details and perform analysis offline to reduce online overhead. However, with this technique there is no guarantee of bug reproduction.

Sasha Fedorova (Simon Fraser) asked for clarification of the purpose of RADBench. Jalbert explained that RADBench's purpose is to present a collection of full snapshots of concurrency bugs "in the wild." It is not a bug-finder or debugging tool. Several audience members asked about the scalability of the work done to create RADBench. Jalbert responded that the bugs did not follow classical patterns, so they were difficult to find and hard to categorize. No faster way to identify the bugs was found. Many audience members suggested places to find more concurrency bugs: work done at IBM, code samples from undergraduate class projects, highly commented buggy entries in project code archives. Jalbert agreed that these were all good potential concurrency bug sources. Jalbert stressed that the RADBench work is just the tip of the iceberg, and acknowledged that many problems in addressing and solving concurrency bugs remain.

Day 1, Session 2

Summarized by Jaswanth Sreeram (jaswanth@gatech.edu)

How to Miscompile Programs with "Benign" Data Races

Hans-J. Boehm, HP Laboratories

Several researchers have investigated the distinction between "harmful" data races (so-called destructive races) and "benign" data races, which do not affect the semantics of a concurrent program and can be safely ignored. In this talk, Hans-J. Boehm argues that even such benign data races, while appearing harmless at the program level, can potentially be compiled into code that produces incorrect results.

Data races are considered errors in several language models, including Ada 83, POSIX and C++/C. However, in Java and C#, data races are not considered errors (although the semantics are not clear). One of the problems with benign data races in languages that consider them errors is that the program may work well when compiled with a specific compiler version but, since the language standard prohibits races, a future version of the compiler may produce incorrect code for the same program.

An important work in PLDI '07 on benign data races identified five types of races at the source-code level. Hans described how each of these five examples could be miscompiled by a reasonable compiler to buggy code. In one case, code hoisting by the compiler resulted in a write operation failing to become visible to other threads. Another type of common benign race is when the reader does not care if it sees the old value before the write or the new value after the write. The problem with this benign race is that it is possible for the reader to see a value that is neither the old value nor the new value. If, for example, the writer updates the high bits of the variable and then the low bits in two distinct operations that are separately atomic, then the reader may see the intervening state of the variable.

Hans gave a seemingly innocuous example of a benign race between two threads, each writing the same value to the same variable. Surprisingly, even this race between two redundant writes can be compiled incorrectly. Briefly, this problem is caused by the compiler promoting the shared variable to a register and then both threads nullifying each other's updates, with the outcome that neither write is seen. Hans remarked that spurious self-assignment instructions, which are another factor contributing to this miscompilation, are disallowed in both the POSIX and the upcoming language standards. Burton Smith from Microsoft noted that self-assignments are a common occurrence in some SIMD codes, and this phenomenon may be prevalent in those programs.

Phil Howard (Portland State) asked how many of the benign races described in the PLDI '07 paper can be miscompiled. Hans replied, all of them. Todd Mytkowicz (Microsoft) said there is a paper in the upcoming PLDI that proposes code motion only on data that is thread local with a slowdown of about 20%. He asked if benign races are important and if race-freedom could be enforced strictly. Hans replied that even if that were possible, a programming model that only permits sequential consistency would not be very useful.

Bryan Ford (Yale) noted that programmers will continue to use benign data races even if there are no guarantees of portability or correctness on a different compiler or platform, and he wondered whether there was a way to write racy programs that would not be miscompiled. Hans answered that in C/C++ there are ways to write racy stores in a manner that aligned with the language standards and with low overheads. Bartosz Milewski (Corensic) remarked that weak-atomics in these languages are like benign races that have been sanctified by the standard and hence are okay to use.

Deterministic OpenMP for Race-Free Parallelism

Amittai Aviram and Bryan Ford, Yale University

Determinism in parallel programs has received a considerable amount of attention in recent years. Deterministic concurrency essentially guarantees that a concurrent program will always produce the same output for a given input. This determinism makes concurrency bugs and transient bugs reproducible, thereby easing debugging. It also enables several mechanisms for fault-tolerance that rely on reproducible replaying of computation on a fault in a particular module performing that computation. Finally, determinism helps in the end-to-end verifiability of concurrent programs.

Synchronization primitives are divided into two classes. Naturally deterministic primitives are those in which program logic alone determines which threads are involved and where the synchronization occurs in each thread's execution. The rest can be classified as naturally non-deterministic.

Amittai Aviram presented a form of Deterministic Open MP (DOMP) that has a pure naturally deterministic programming model and race-free semantics. DOMP features the subset of the OpenMP parallel constructs that are deterministic, such as "parallel," "loop," and "sections." Amittai reported that in an analysis of the SPLASH, PARSEC, and NPB suites, around 92% of the occurrences of naturally non-deterministic constructs were to express programming idioms that were purely naturally deterministic at a high level. One of the reasons for this is that OpenMP's deterministic constructs are not expressive enough. For example, the OpenMP "reduction" clause constrains the input type to a scalar and the operation to simple arithmetic or logical operators. OpenMP also lacks a high-level "pipeline" construct, necessitating having to build it from spin-loops and the non-deterministic "flush" construct. The abstractions in DOMP are designed to be expressive enough that programmers do not have to resort to using lower-level naturally non-deterministic constructs to implement them. DOMP offers a generalized reduction clause and a pipeline construct (both of which are naturally deterministic). However, DOMP excludes the naturally non-deterministic constructs from OpenMP, such as "atomic," "critical," and "flush," citing the above observation that these constructs are usually used as low-level components of higher-level idioms which the programming model does not itself provide.

Finally, the DOMP runtime is designed to be race-free, since determinism at the program level requires race-freedom in both the program and the runtime. DOMP uses a "working-copies" programming model which eliminates races. In this model each thread makes a private copy of the shared state and operates on it in isolation. When the thread is finished the runtime merges this updated shared state with the parent thread's pristine copy of the state. If at this point the runtime detects that two or more threads have concurrently modified the same state, then it signals a runtime error.

Steve Johnson (Wave Semiconductor) asked if the merging of copies was data-dependent. Amittai answered that entire regions of data that are in scope have to be merged. Gilles Pokam (Intel) asked how the merging was done and if the order in which states were merged was important. Amittai replied that the runtime simply checks to see if the value in a shared location in some thread's private copy differs from the parent thread's pristine copy; if two or more threads have modified this location, the runtime signals an error. Bryan Ford asked if the runtime's model was similar to snapshot isolation. Amittai replied that it was close to it. Steve Johnson asked, if the runtime signals two writes to the same location as an error, then can the program have runtime errors that are data-dependent and, if so, would that affect the determinism guarantee? Amittai replied that such errors are possible and that the usual testing and quality assurance processes were still required. Hans Boehm (HP Labs) remarked that determinism was in the eye of the beholder and asked whether the authors considered malloc to be deterministic. Amittai replied that malloc uses locks so he wouldn't consider it deterministic.

Day 1, Session 3

Summarized by Bryan Catanzaro (bcatanzaro@acm.org)

Parkour: Parallel Speedup Estimates for Serial Programs

Donghwan Jeon, Saturnino Garcia, Chris Louie, and Michael Bedford Taylor; University of California, San Diego

Before embarking on a project to parallelize an application, it's natural to ask what payoff you should expect from parallelism. Some applications are naturally more parallel than others, and you would like to be confident that your application is parallelizable before actually rewriting it to take advantage of parallelism. Donghwan Jeon presented a method for doing that. Parkour instruments a binary during compilation, and then examines execution traces of the instrumented binary for parallelism, given a model of the target parallel hardware platform.

Parkour uses a variation of critical path analysis (CPA) to estimate parallelism. CPA constructs a dataflow graph of the instructions in a program to discover the dependencies in the program execution. The longest dependency chain in the execution trace is used to find the span, and the overall graph shows the amount of work in the execution trace. Simplistically, CPA can give a parallelism estimate of work/span, but this is only a theoretical limit and is very poorly correlated with realizable parallelism. Parkour uses a hierarchical critical path analysis (HCPA), where a hierarchy is imposed by programmer-visible structures in the original code. Given a model of the type of parallelism a programmer would use to target a particular platform, HCPA then performs local critical path analysis at each node in the hierarchy. Heuristics are used to create parallelism estimates for each level, limiting the parallelism described by the CPA at each level by the realizable parallelism presented by the target platform. Results were presented for two platforms—a multicore x86 platform and the MIT Raw processor—on a selection of benchmarks, compared to hand-parallelized implementations. In general, it was clear that the HCPA approach advocated here gave a fairly accurate estimate of achievable parallelism. One next step would be to use this tool to aid in actually parallelizing an application, since it seems to identify portions of the program which can be parallelized.

Sasha Fedorova asked if Donghwan and his co-authors were applying CPA dynamically as opposed to statically. Donghwan said they were. Mike McCool asked if loop unrolling stole loop-level parallelism, and Donghwan again answered yes. Someone asked how Parkour avoids underestimating speedup. Donghwan replied that they need to use a machine model. Sasha asked if Parkour could be extended to implement parallelism, and Bryan Catanzaro followed up by asking if they inserted OpenMP pragmas. Donghwan said that Parkour does find important regions to parallelize, and they covered that in another paper, but it does not insert OpenMP constructs, as that was too complicated. Craig Mustard (Simon Fraser) asked if Donghwan could elaborate on the planner. Donghwan replied that they roughly model the characteristics of two planners.

Enabling Multiple Accelerator Acceleration for Java/OpenMP

Ronald Veldema, Thorsten Blass, and Michael Philippsen, University of Erlangen-Nuremberg

Ronald Veldema explained that this project is aimed at heterogeneous clusters, including both traditional CPUs and accelerators, in this case GPUs. Since clusters are dynamically loaded, it's difficult to know statically what mixture of traditional CPUs and accelerators an application will have at its disposal during execution on a shared cluster. To make this easier, this work proposed writing parallel platform-independent code, using OpenMP directives embedded as comments in Java source code. When a program is instantiated across a cluster, a modified Java class loader examines the compute resources of each node and then dynamically compiles the computation to fit the resources discovered. Work is dispatched to traditional CPUs using parallel Java execution and to GPUs using CUDA. Work is partitioned automatically by running micro-benchmarks which give an estimate of the capabilities of each of the processors; more capable processors are assigned proportionally more work.

An important part of this project was the memory model used to simplify cluster programming: if the compiler can prove that all references to a particular array are local with respect to the loop being parallelized across cluster nodes, the compiler is able to statically partition the data structure, allowing the program to work with large data structures that cannot fit in a single node's memory. Otherwise, the data structure is duplicated across the cluster. Duplicated data structures are made coherent at OpenMP boundaries in the original program, by keeping a shadow copy of the duplicated data structure and diffing it against the potentially mutated copy created during program execution. Diffs are interchanged to synchronize data across the cluster, all without programmer intervention.

Scaling results from this approach seemed good on the examples they presented, with a clear benefit from using the GPUs to accelerate large computations on the cluster. However, this approach does not allow programmers to take advantage of the widely divergent memory subsystems of the CPU and GPU and, instead, requires programmers to write simple OpenMP parallel loops. The goal of this project, therefore, is not to obtain performance competitive with hand-tuned parallel programs but to enable programmers to very productively exploit heterogeneous clusters, including both CPUs and other accelerators, such as GPUs.

Bryan Catanzaro said that with CUDA 4, you can share GPU memory. Ronald responded that you cannot do MPI message passing between GPUs. Bryan next asked if this can be made deterministic. Ronald answered that they have no control over which hardware will be used—for example, if a GPU uses a different type of float. Bryan countered that Java uses a strict float model, but Ronald responded that this is true only if something is marked as strict. John Kubiatowicz (UC Berkeley) said that this reminds him of work in the '90s, where there were interface issues and communication that didn't quite work. He suggested looking at the older literature to see where it would fit into this work. Ronald said that if he could get MPI to work on a GPU, he would be happy. John pointed out that diffs work well in hardware already, and Ronald replied that we don't need hardware support for finding difference between arrays, as the cost is now negligible.

Day 1, Session 4

Summarized by Sean Halle (seanhalle@yahoo.com)

CUDA-level Performance with Python-level Productivity for Gaussian Mixture Model Applications:

H. Cook, E. Gonina, and S. Kamil, University of California, Berkeley; G. Friedland, International Computer Science Institute; D. Patterson and A. Fox, University of California, Berkeley

Selective Embedded Just-in-Time Specialization (SEJITS) is a framework for applying specializing high-productivity languages to specific hardware at runtime. The programmers use a convenient productivity language and make calls to the SEJITS library for functions they need. For example, using the Python library's Gaussian Mixture Model, the SEJITS library picks the version of the code that performs best on the particular hardware during the run. These versions were created during library development, coded by hand in an efficiency language like C or C++.

This talk focused on Speaker Diarization for speech recognition. This was coded in Python and calls the SEJITS library to leverage the Gaussian Mixture Model (GMM). This GMM library uses multiple kernels, each optimized to particular data characteristics. The SEJITS library picked the best one. Performance was tested on the GTX285 and 480 GPUs, for a number of different input data sizes. It showed an average 32% improvement by using multiple kernel variants and picking the best one dynamically over the course of the run. For larger input sizes, improvement rose to 75%.

When comparing the hand-coded C++/CUDA version during the first run, using SEJITS added 71% overhead. But the library remembered the results, and subsequent runs were 17% faster than the hand-coded C++/CUDA version.

Someone from MIT asked about debugging with all the extra layers. What happens when there's a bug down inside the kernels in the specializer? They're working on the ability to trace where a bug happens; such buried bugs are the same in any multi-level library approach. Is the SEJITS approach the same as just dynamically linking a library? They can do the same thing by linking an appropriately tuned library implementation into Python, so what does SEJITS buy? Armando Fox answered that SEJITS is designed to be used by productivity programmers, not specialists. Also, the specializer can take into account the amount of data at runtime, something you cannot do at compile time. Where is most of the debugging time spent? Most of the development and debugging is in writing the C++ and CUDA low-level kernel code, then embedding that kernel into the SEJITS specializer and linking that to the Python API. Linking to Python is straightforward, with lots of tools available. This linking exposes the CUDA programming model to the framework.

Sasha Fedorova wanted to see a code example, and E. Gonina showed a 40-line example that would be thousands of LOC in C. Sasha then asked how the programmer interacted with SEJITS, and Armando said that the productivity programmer never see SEJITS. Only the kernel expert needs to see the C code. Steve Johnson asked how productivity programmers can improve performance. Gonina answered that they can run the specializer over and over again, while Armando said that they really can't do anything.

Pervasive Parallelism for Managed Runtimes

Albert Noll and Thomas R. Gross, ETH Zurich

Albert Noll summarized the known phenomenon that task scheduling overhead can grow to dominate work as the number of tasks grows large. He showed a graph that marks the critical point where the number of individual tasks is too large relative to the work-time of a single one. He termed this the critical point. Albert emphasized that a JIT is isolated from parallel-task knowledge and so has no means of alleviating this parallel scheduler overhead problem.

Their contribution is to modify the JVM, by modifying the intermediate representation, calling it ParIR. This modification relies on Cilk-style spawn and sync semantics. No mention was made of more interesting semantics to cover a larger class of applications. Noll showed two optimizations that can be done using this IR during the run. The first is merging parallel tasks into a single composite task. He showed that this improves performance when the number of tasks is above a critical point for that task-type. He suggested that profiling information can be collected, and the code recompiled when it discovers that the task size is too small or too large. The JIT modifies the intermediate representation of the code to inline a chosen number of tasks, then it recompiles.

The second optimization done with ParIR is moving invariant code out of a parallel section. . He said that recompilation based on profile information is only possible with a JIT.

Burton Smith commented that the semantics look Cilk-like, which lets the compiler merge iteratively generated tasks easily. However, merging for recursively generated tasks is harder for a compiler inlining approach. Noll agreed. Sean Halle asked if profiling has been implemented—which watches and then does the recompilation? How does it know the critical task size? It's a work in progress. They expect to use hardware counters to collect profiling. Hans Boehm asked how the profiler knows the critical point. It is different for each task-type, and many tasks have variable running time. They will have to perform some kind of statistical analysis. It will have to try many task sizes before it knows whether the current one is critical size.

Poster Session

First set of posters summarized by Shane Mottishaw (smottish@sfu.ca)

Support of Collective Effort Towards Performance Portability

Sean Halle and Albert Cohen, INRIA, France

Sean Halle presented this work prompted by the growing desire to express parallel programs in a single source language and achieve good performance across a range of hardware platforms. The authors recognize that the challenges of productivity, portability, and adoptability cannot be feasibly achieved by any one group, nor can they be solved solely at the language, runtime, or hardware abstraction level. Because there is a wide array of research projects involved in performance portability that involve different runtimes, hardware abstractions, and languages, the authors propose a comprehensive support system which provides a framework in which independent groups can plug their solution (e.g., runtime, language, hardware abstraction) into a layer of the framework, making it available for everyone else to utilize in their own work. There are three main layers: toolchains (languages and compilers), parallel runtimes, and hardware abstractions. This support system is based on Virtualized Master Slave (VMS), the authors' virtualization mechanism, which replaces threads and provides pieces for each level of the support system (e.g., VMS cores for hardware abstraction and plugins for runtimes). To achieve performance, layers of the support system share information with each other. For example, the toolchain can derive information about data and computation needs of a task, which can then be used by the runtime to make scheduling decisions.

Challenges in Real-Time Synchronization

Philippe Stellwag and Wolfgang Schröder-Preikschat, Friedrich-Alexander University Erlangen-Nuremberg

Philippe Stellwag described a parallel NCAS library (rtNCAS) that enables the creation of arbitrary, lock-free, or wait-free data structures. Additionally, the library guarantees that all data structure operations are linearizable. Users of rtNCAS provide a function that implements a (sequential) algorithm to perform an update to a data structure (e.g., an enqueue operation for a FIFO queue). This function returns a structure with the expected old and new values for some state of the data structure and is used by the library to perform an NCAS operation to conditionally swap the old and new values. The rtNCAS library performs the user-defined operation as follows: it first tries to speculatively call the user-defined function and attempts to update the data structure with an NCAS operation. On failure, this operation is delayed by pushing it onto a wait-free FIFO queue (called the operation queue). All threads (regardless of the success of speculative execution) cooperatively execute stalled NCAS operations on the queue. This cooperative behavior combined with speculative execution provides wait-free, disjoint-access parallel access to data structures. The wait-free property also provides upper-bound execution times, which is crucial for real-time applications.

Coding Stencil Computations Using the Pochoir Stencil-Specification Language

Yuan Tang, Rezaul Chowdhury, Chi-Keung Luk, and Charles E. Leiserson, MIT Computer Science and Artificial Intelligence Laboratory

Yuan Tang described a new domain-specific language/compiler framework that allows for efficient stencil computations to be embedded in C++. The user describes their stencil computation (including boundary conditions, shape, dimensionality, data types, and computation kernel) using the Pochoir language. The Pochoir compiler and template library then perform automatic parallelization and cache optimization. Pochoir improves upon a "trapezoidal decomposition" algorithm produced by Matteo Frigo and Volker Strumpen by performing hyperspace cuts to partition n-dimensional grids, yielding more parallelism without sacrificing the cache efficiency of the original trapezoidal decomposition algorithm. The Pochoir runtime system also employs a number of other stencil-specific optimizations. The Pochoir runtime system utilizes Intel Cilk Plus to parallelize code written in the Pochoir language.

Dynamic Prioritization for Parallel Traversal of Irregularly Structured Spatio-Temporal Graphs

Bo Zhang, Duke University; Jingfang Huang, University of North Carolina at Chapel Hill; Nikos P. Pitsianis, Aristotle University; Xiaobai Sun, Duke University

Bo Zhang presented work concerned with the execution of fast/sparse algorithms for all-to-all transformations (e.g., fast Fourier transform, FFT, and fast multipole method, FFM) on multicore architectures. The authors represent FFT or FFM computations as a spatio-temporal directed acyclic graph (ST-DAG) where nodes define computations on spatial entities (e.g., cells in a grid) and edges define spatial and temporal (iteration) dependencies. A parallel traversal of this graph is executed to perform the transformation. At any point in the graph, however, there are often far more tasks available for execution than there are resources to satisfy the requests, and therefore efficient scheduling of these tasks is required. The goal is to first schedule tasks that have the largest impact on the overall execution time. The authors introduce dynamic prioritization to solve this problem. Dynamic prioritization takes into account resource limitations and varying task sizes to perform adaptive ranking of available tasks and executes tasks of higher ranks first in order to reduce the time of the parallel traversal.

Efficient and Correct Transactional Memory Programs Combining Snapshot Isolation and Static Analysis

Ricardo J. Dias, João M. Lourenço, and Nuno M. Preguiça, Universidade Nova de Lisboa, Portugal

Ricardo J. Dias proposed a novel method for reducing memory tracking overhead of transactional memory systems, without sacrificing serializability. Using snapshot isolation (where each transaction executes using a private copy of system state), only write-write conflicts need be detected, thus reducing runtime overhead. However, snapshot isolation may lead to non-serializable executions. To correct this, static analysis (specifically, shape analysis) is used to determine the abstract read and write sets of transactions. These read-write sets can then be compared to determine dependencies between transactions, thus detecting potential conflicts. These conflicts are then corrected automatically prior to executing the transactions. For example, a dummy write can be inserted to force a write-write conflict at runtime.

Second set of posters summarized by Craig Mustard (craiig@gmail.com)

Feasibility of Dynamic Binary Parallelization

Jing Yang, Kevin Skadron, Mary Lou Soffa, and Kamin Whitehouse, University of Virginia

Jing Yang presented a method of parallelizing binary-only executables by analyzing the execution of the binary and identifying frequently repeated sections which become candidates for parallelization. When the sequential program reaches a point that has previously been traced and is a candidate for parallelization, the sequential execution halts and a parallel version is speculatively executed with a copy of the program state. If the parallel execution fails due to misprediction, then the results are discarded and the sequential version is executed. The authors present a prototype implementation using a simulator that they evaluate with the SPEC2000 and MediaBench benchmark suites. The authors also applied dynamic binary optimization techniques (DBO) to their experiments. The authors find that DBP and DBO enable a 2x speedup for 7 out of 10 floating point benchmarks, and a speedup of 1.27x for integer benchmarks.

Automated Fingerprinting of Performance Pathologies Using Performance Monitoring Units (PMUs)

Wucherl Yoo and Kevin Larson, University of Illinois at Urbana-Champaign; Lee Baugh, Intel Corp.; Sangkyum Kim, Wonsun Ahn, and Roy H. Campbell, University of Illinois at Urbana-Champaign

Wucherl Yoo described a way to automatically fingerprint the performance behavior of applications. The authors first wrote a variety of micro-benchmarks which exhibited different pathological performance characteristics. Then they trained a decision-tree learning algorithm to identify these micro-benchmarks by their exhibited performance characteristics. They then analyzed timesliced profiles of benchmark applications from SPEC and PARSEC and classified particular program phases as exhibiting pathological performance behavior. They achieved an accuracy rate of over 97% for all benchmarks. This work is designed to be added to a profiling suite so that performance characteristics of applications can be classified and instructions can be provided to the user about ways to fix such problems.

PACORA: Performance Aware Convex Optimization for Resource Allocation

Sarah L. Bird, University of California—Berkeley; Burton J. Smith, Microsoft

Sarah Bird and Burton Smith presented PACORA, which endeavors to optimally allocate resources by mathematically modeling the resources to the quality-of-service tradeoff for each program. The authors combine the resource-performance curve of each program and apply convex optimization techniques to find an optimal configuration of resources such that the resources-performance tradeoff for the entire system performance is maximized. In their model, the authors include a special idle process that represents free resources, which contributes to energy savings. Since this optimization can be done iteratively using a gradient descent approach, the authors believe PACORA will be a useful and high performance technique for resource management.

Are Database-style Transactions Right for Modern Parallel Programs?

Jaswanth Sreeram and Santosh Pande, Georgia Institute of Technology

The authors argue that database-style transactions are too rigid to effectively express certain parallel programming patterns. They describe the applications that can benefit from relaxed models as "soft computing applications." Kmeans, for example, benefits from relaxing the guarantees of transactional memory in order to speed up the clustering algorithm by allowing threads to use old and slightly inaccurate values. When the accuracy is allowed to vary by 0.1, the performance increase is significant, while there is no significant increase in error. List search is another example: if a conflict is detected while a thread is searching the list, instead of aborting, the search can be repaired by reading in a new value.

Day 2, Session 1

Summarized by Bryan Catanzaro (bcatanzaro@acm.org)

Balance Principles for Algorithm-Architecture Co-Design

Kent Czechowski, Casey Battaglino, Chris McClanahan, Aparna Chandramowlishwaran, and Richard Vuduc, Georgia Institute of Technology

This paper advocates the use of theoretical modeling to guide architecture development. How much of the architecture should be devoted to cores, for example, and how much to cache? Given a particular parallel architecture, what classes of computation would perform efficiently? This work responds to the observation that simulators are hard to build and require a lot of investment. Once the simulator is built, there are many fundamental assumptions which have been baked into the simulator and are expensive to change. Instead, this paper advocates that processor performance should be theoretically modeled based on high-level characteristics, such as communication and scalability.

The presentation defined a balanced architecture as one where the memory wait time T_mem <= the computation time T_comp. In order to evaluate T_mem and T_comp for a particular algorithm and architecture, one needs to derive a model for both expressions. For T_mem, they used an external memory model (I/O model). For T_comp, they used a Parallel DAG model to discover the work and span of a computation, and then found parallelism using Brent's theorem. The I/O model depends on the parallel cache complexity, which needs to be derived separately from the sequential cache complexity, and depends on scheduling choices. As a punchline, the paper presented results showing that even dense matrix multiplication, the canonically compute-bound kernel, will be bandwidth-bound on GPUs by 2021 if the current scaling trends continue in computation and bandwidth resources. This approach does not consider power dissipation, which might suggeest use of a more general-cost metric.

Crunching Large Graphs with Commodity Processors

Jacob Nelson, Brandon Myers, A.H. Hunter, Preston Briggs, Luis Ceze, Carl Ebeling, and Dan Grossman, University of Washington; Simon Kahan, University of Washington and Pacific Northwest National Laboratory; Mark Oskin, University of Washington

Important graphs found in many real-world applications are both very large and have a low diameter, meaning that they are very hard to partition. This poses complications for graph algorithms which operate on these graphs. The Cray XMT architecture has been very successful at operating on these problems, but it is expensive, and its performance is not competitive on dense problems. This work attempts to utilize insights from the Cray XMT design to enable commodity CPU clusters to perform well on graph algorithms as well as on the denser problems for which they have already been shown to perform well. The main advantages of the Cray XMT over a commodity x86 cluster are the number of contexts the Cray XMT can keep on chip and the high number of outstanding memory transactions it can support. This work describes SoftXMT, a library for x86 processors which aims to provide these advantages to x86 software through the use of lightweight multithreading in software.

SoftXMT uses co-routines to break memory transactions into separate stages. For example, a load becomes a prefetch, a yield, and then a blocking load. The full implementation of SoftXMT will use a compiler which transforms memory transactions into multiple stages. A lightweight library round-robin switches between suspended threads which are waiting on memory requests. The authors presented data which shows that on a single node, the co-routine library used in SoftXMT is efficient, allowing the node to saturate its available memory bandwidth almost as well as the hardware can, as is demonstrated with a simple pointer-chasing benchmark. Future work involves making a complete cluster-based implementation and showing good performance on complete graph algorithm problems.

Day 2, Session 2

Summarized by Amittai Aviram (amittai.aviram@yale.edu)

Multicore Performance Optimization Using Partner Cores

Eric Lau and Jason E Miller, MIT Computer Science and Artificial Intelligence Laboratory; Inseok Choi and Donald Yeung, University of Maryland; Saman Amarasinghe and Anant Agarwal, MIT Computer Science and Artificial Intelligence Laboratory

Increasingly, parallel architecture is exposing more hardware resources to the programmer, who must cope with the contradictory requirements of high performance and energy efficiency in a programming environment whose growing complexity is getting unmanageable. Self-aware programs can manage their resources dynamically, but they burden the CPU with a new meta-program management layer of work, introducing more interrupts and context switches. Eric Lau presented this work which, assuming a tiled general architecture for the future, introduces the idea of partner cores as one possible solution: a smaller core, one-tenth the main core's size and optimized for efficiency, alongside each main core, with its own, lower-powered network router and with dedicated "probes" feeding it performance counters and other status information from the main core. The partner core could then handle the meta-program management, for instance, by running a helper thread to prefetch data from main memory into the cache. Experiments on a simulator showed that partner core memory prefetching could raise performance close to 3x, while more than doubling energy efficiency. Other scenarios that could benefit from partner cores are (1) keeping track of "tainted" pointers in information flow control, (2) running a redundant trailing thread to check output against the main thread periodically for error detection, and (3) prioritizing messages on an event queue.

Phil Howard (Portland State) asked who would program for partner core architectures. Eric answered that it would be system programmers, hopefully, at a level that application programmers would not have to see. Stephen Johnson (Wave Semiconductor) asked whether partner cores could be used to execute assertions so as to provide runtime correctness checks in deployment while staying out of the way of the main program execution. Eric found this very plausible.

Parallel Pattern Detection for Architectural Improvements

Jason A. Poovey, Brian P. Railing, and Thomas M. Conte, Georgia Institute of Technology

Although parallel programming promises performance improvements, optimizing to get the most out of a parallel program poses design challenges. Jason Poovey presented this work on how designing according to parallel patterns can help meet these challenges. Researchers have identified patterns at the level of concept, algorithm, and low-level implementation. The algorithmic level offers both quantifiability and breadth sufficient to help automate optimization and to guide improvements in hardware architecture. For example, thread schedules for pipeline and divide-and-conquer algorithms are quite distinct. While algorithms involving intensive data sharing require the usual MESI (Modified, Exclusive, Shared, or Invalid) cache coherency, algorithms in which data migrate from thread to thread, as in pipelines, could get by with the weaker, and more efficient, MI protocol. Algorithms with little inter-thread communication need far less network bandwidth than those that communicate frequently. Pipeline and divide-and-conquer are two of the six classes that typify parallel algorithms. Those organized by task are task parallel and divide-and-conquer algorithms; those by data, geometric decomposition and recursive algorithms; and those by the flow of data, pipelines and event-based coordination.

In prior work, Jason and colleagues had shown that significant performance improvements are possible when the pattern was known and was used to determine the thread-balancing mechanism. To identify the pattern to which a program belongs, we have two major sources: static detection, including the programmer's own annotations, perhaps through a new API for this purpose; and dynamic detection, which measures data-sharing behavior, thread balance over time, and the uniqueness of instructions to each thread in order to identify signature combinations suggesting particular parallel algorithm patterns. In the case of data sharing, a modest modification to the cache could provide accurate information at a reasonable cost. The team created five benchmarks, one for each pattern except event-based coordination, to serve as reference points ("golden copies"). Once they had compiled measurements for dynamic detection from the reference benchmarks, they ran several standard benchmarks on a simulator and used the same measurements to predict their respective patterns. In each case, they knew from the outset the appropriate pattern, and they found that their data-based predictions had mixed success. They plan further improvements in data gathering methods and modeling in order to improve prediction accuracy.

Michael McCool (Intel) suggested a connection with the previous presentation: could we use partner cores to help detect parallel algorithm patterns? Jason agreed. Another question was how much hardware we need for pattern detection. "Right now, a lot," Jason answered. He then detailed some of the larger sources of data. Hopefully, people may find methods for detecting patterns that will eventually require fewer resources. Could we use software instead of hardware for these measurements? In principle, yes, but it would be even less efficient. Does Jason foresee the need for custom hardware? For thread scheduling, no; but for switching between cache coherency protocols, yes. Is the benefit worth the cost of custom hardware? Switching cache coherency protocols when possible could offer significant performance benefits. Why did the team have to use a simulator? This was the only way that they could collect large data sets for pattern detection. However, their eventual usage model would be dynamic pattern detection and optimizing adjustments during runtime. They also hope to identify more distinct patterns.

Day 2, Session 3

Summarized by Amittai Aviram (amittai.aviram@yale.edu)

Parallel Programming of General-Purpose Programs Using Task-Based Programming Models

Hans Vandierendonck, Ghent University; Polyvios Pratikakis and Dimitrios S. Nikolopoulos, Foundation for Research and Technology—Hellas (FORTH)

Hans Vandierendonck presented this work. Pipelines are a common and essential pattern of parallel programming, in which later tasks depend on earlier ones, sometimes in complex ways, following a directed acyclic graph (DAG) rather than a mere sequence; some tasks are necessarily sequential (such as I/O), while others may be run in parallel but depend on the completion of predecessor tasks. Yet current thread-based programming languages and frameworks, even higher-level ones such as Cilk++ and TBB, are not designed to make pipeline construction easy or intuitive. The awkwardness is evident when the programmer wants to have variables renamed to optimize pipelines. This happens when one task must wait to write to a variable until another task has read the old value (write-after-read, WAR, or anti-dependency), and when one task must overwrite a variable only after another has written a previous value (write-after-write, WAW, or output dependency). Variable renaming enables the second task to proceed without locking or blocking, but Cilk++ and TBB, both thread-based, make the programmer have to program versioning manually with complicated, unintuitive syntax.

A task-based dataflow language could manage the versioning automatically when it infers the need for it, using new extensions to Cilk++ to annotate variable arguments to tasks with their dependency types (indep, outdep, or inoutdep), as well as the standard Cilk++ versioned hyperobject keyword. The result is simpler, more readable code. Such a language could be further extended to accommodate speculative execution, where a thread could execute in parallel while presuming a condition, and then check the state of a variable on which it depends before either committing results or aborting. One could also use dependency-annotated types to prove the deterministic execution of a parallel program.

In implementing task-based dataflow parallelism, the key challenge is to design an efficient scheduler, which could automatically manage dependencies and blocking with a minimum of locks. Hans's team's solution is a ticket queue system, requiring only one lock per task and one on the global queue. Experiments on the SPEC2 benchmarks bzip2 and hmmer show that the task-based dataflow extensions to Cilk++ impose essentially no additional runtime cost relative to standard Cilk++ implementations.

Michael McCool (Intel) picked up on Hans's comment in passing that his team had to accept compromises in designing the scheduler. Hans clarified that the resulting task graph could have extra leaves corresponding to tasks that were waiting to execute. Leo Meyerovich (UC Berkeley) asked whether they had tried comparing their system to task-parallel systems on established benchmarks. Hans said they had compared it with SMPSS, and found that their system performed about the same as SMPSS on Choleski and better than SMPSS on Jacobi, because of the scheduler's reduced overhead.

Parallel Programming with Inductive Synthesis

Shaon Barman, Rastislav Bodik, Sagar Jain, Yewen Pu, Saurabh Srivastava, and Nicholas Tung, University of California, Berkeley

Ras Bodik presented this work. In scientific computing, high-level code synthesizers and libraries ease high-performance code implementation, but only apply to the restricted domains the knowledge of which they reflect. Absent such domain theory, the programmer must use more general, lower-level compilers, with lesser performance, and may need to optimize by hand, which takes time and invites errors, especially when optimization involves parallelizing.

This team's project aims to resolve this dilemma with a tool that enables programmers to specify enough information so that a code synthesizer can do the rest, without the programmer having to know all about the domain. Their solution is based on the SKETCH inductive program synthesis framework, which has the programmer provide (1) a specification of what the equivalent result should be (using a naive algorithm) and (2) a high-level "sketch" or template of what a more efficient algorithm should "look like," in which the programmer uses placeholders ("??") for key constants or variables. The SKETCH synthesizer fills in the placeholders to complete the source code, which can then be compiled down to a high-performance executable. For example, a programmer could provide a specification and template of matrix transposition, and SKETCH would fill in the right index variables in the algorithm to produce a correct and efficient program.

One limitation of SKETCH is that it does not prove correctness, only functional equivalence of its code to the specification for each element in a finite subset of the domain. SKETCH also has scalability limitations, which the current project aims to overcome by experimenting with an interactive refinement and auto-tuning cycle. SKETCH would first produce a naive algorithm based on a simple template (but more efficient than the algorithm in the specification). Next, the programmer would adjust the template based on the resulting algorithm (source code) and resubmit it to SKETCH for automatic tuning. The programmer could continue repeating this cycle. In particular, each phase of refinement could reflect the knowledge of an expert in a distinct domain, each one working at a high level, so that the resulting code would reflect several levels of domain knowledge without requiring any hand optimization. The team applied this technique to the particularly difficult and error-prone task of parallel programming for GPUs, in particular, to solve the maximal independent set (MIS) problem. They began with a sketch that would suggest an exponential algorithm, but refined and auto-tuned to have an efficient dynamic programming algorithm, based on the parallel scan operation.

Another refinement, reflecting "parallel algorithm expert" knowledge, led to an efficient SIMD algorithm to implement the parallel scan network. Finally, refinement according to "GPU tuning expert" knowledge optimized for execution on GPUs by avoiding bank conflicts, having the synthesizer produce an array index translation function so as to map logical arrays to physical arrays. The result was an efficient algorithm, whose further refinement is still in progress.

The ensuing discussion showed lively interest and the need for clarification. SKETCH does not provide a proof of correctness; in the matrix transposition case, the specification might use a matrix of size *n* where *n* is small, and then use a small range of larger *n* to check functional equivalence. At times, the programmer might have to prove correctness by hand. SKETCH looks like constraint programming, but the constraints are in the metalanguage, restricting the search space to a manageable size. In the specification and template, you can also place constraints to force optimizations or to filter out inefficient programs. In principle, one could also use SKETCH to build up a whole library of alternative solutions to the same general problem, but SKETCH cannot yet generate variations automatically in a way that makes sense, which would require that it distinguish meaningful from trivial variations.

Day 2, Session 4

No report is available for this session.

A Relativistic Enhancement to Software Transactional Memory

Philip W. Howard and Jonathan Walpole, Portland State University

Quarantine: Fault Tolerance for Concurrent Servers with Data-Driven Selective Isolation

Thanumalayan Sankaranarayana Pillai, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

Linux Magazine
ACADEMY

20% off for USENIX members

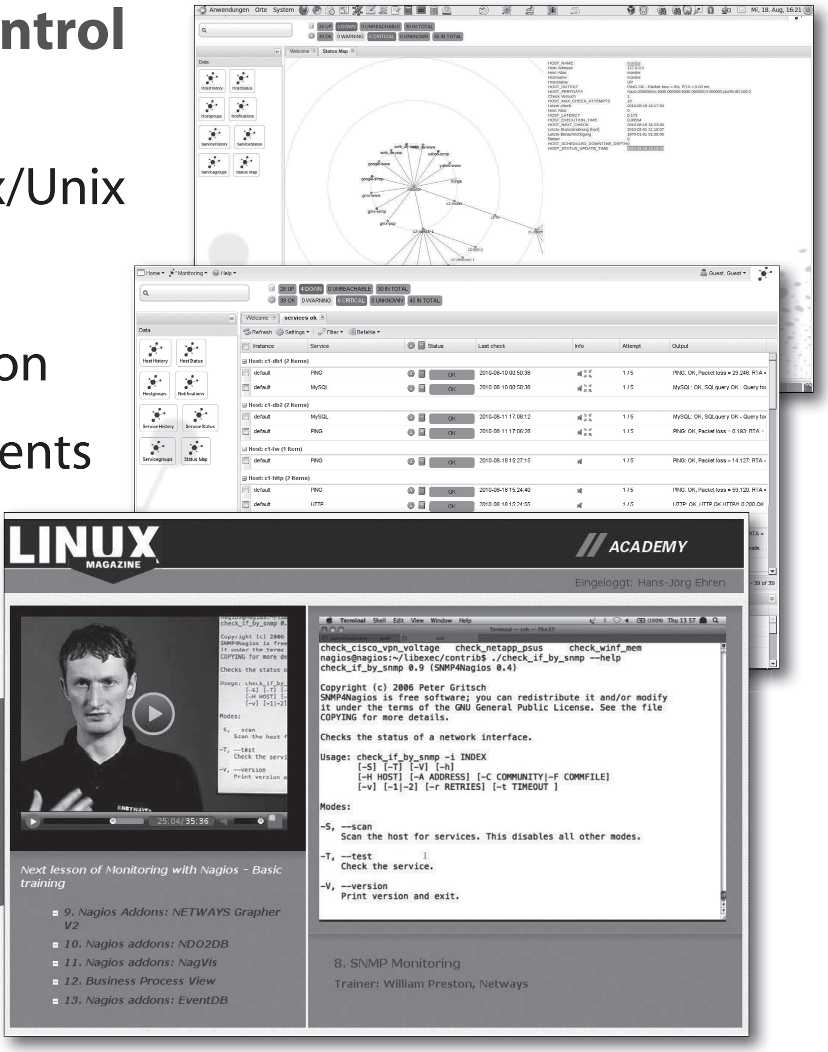
Online Training at Linux Magazine Academy

Monitoring with Nagios

Getting your IT under control the easy way:

- Monitoring Windows/Linux/Unix
- Web front-end
- Structuring the configuration
- Monitoring SNMP-components
- Nagvis, Grapher V2, and ND02DB add-ons

Many practical examples!



Eingeloggt: Hans-Jörg Ehren

```
check_cisco_vpn_voltage   check_netapp_psus   check_winf_mem
nagios@nagios:~/libexec/contrib$ ./check_if_by_snmp --help
check_if_by_snmp 0.9 (SNMP4Nagios 0.4)

Copyright (c) 2006 Peter Gritsch
SNMP4Nagios is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License. See the file
COPYING for more details.

Checks the status of a network interface.

Usage: check_if_by_snmp -i INDEX
       [-S] [-T] [-V] [-h]
       [-H HOST] [-A ADDRESS] [-C COMMUNITY|-F COMMFILE]
       [-v] [-1|-2] [-r RETRIES] [-t TIMEOUT ]

Modes:

-S, --scan
   Scan the host for services. This disables all other modes.

-T, --test
   Check the service.

-V, --version
   Print version and exit.
```

Next lesson of Monitoring with Nagios - Basic training

- *9. Nagios Addons: NETWAYS Grapher V2*
- *10. Nagios addons: NDO2DB*
- *11. Nagios addons: NagVis*
- *12. Business Process View*
- *13. Nagios addons: EventDB*

8. SNMP Monitoring
Trainer: William Preston, Netways

More information:
www.academy.linux-magazine.com/usenix

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 '11 for training and face time with experts in the sysadmin community.

The theme for LISA '11 is "DevOps: New Challenges, Proven Values."

LISA '11 will feature a Keynote Address: "The DevOps Transformation," by Ben Rockwood, and a Closing Session: "Tales from IBM's Watson—Jeopardy! Champion," as well as:

6 days of training on topics including:

- Virtualization
- Security
- Configuration Management
- And more!

Plus a 3-day Technical Program:

- Invited Talks
- Paper Presentations
- Guru Is In Sessions
- Practice and Experience Reports
- Vendor Exhibition
- Workshops
- Posters and WiPs

Stay Connected... http://www.usenix.org/facebook http://twitter.com/LISAConference

Register today at www.usenix.org/lisa11/lg