usenix ;login:

JUNE 2011 VOL. 36, NO. 3

Apache Hadoop: The Scalability Update
KONSTANTIN V. SHVACHKO

Building a Cloud File System
JEFF DARCY

Testing the Transport Side of Highly Available Hosts
STUART KENDRICK

Conference Reports from FAST '11, NSDI '11, and more

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

usenix UPCOMING EVENTS

2011 USENIX Federated Conferences Week

June 14–17, 2011, Portland, OR, USA
http://www.usenix.org/fcw11
Events include:

3rd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '11)
June 14–15, 2011
http://www.usenix.org/hotcloud11

3rd USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '11)
June 14, 2011
http://www.usenix.org/hotstorage11

3rd Workshop on I/O Virtualization (WIOV '11)
June 14, 2011
http://www.usenix.org/wiov11

2011 USENIX Annual Technical Conference (USENIX ATC '11)
June 15–17, 2011
http://www.usenix.org/atc11

2nd USENIX Conference on Web Application Development (WebApps '11)
June 15–16, 2011
http://www.usenix.org/webapps11

3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP '11)

June 20–21, 2011, Heraklion, Crete, Greece
http://www.usenix.org/tapp11

The 9th Annual International Conference on Mobile Systems, Applications, and Services

JOINTLY SPONSORED BY USENIX AND ACM SIGMOBILE IN COOPERATION WITH ACM SIGOPS

June 28–July 1, 2011, Washington, DC, USA
http://www.sigmobile.org/mobisys/2011/

The 2nd ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2011)

SPONSORED BY ACM SIGOPS IN COOPERATION WITH USENIX

July 11–12, 2011, Shanghai, China
http://apsys11.ucsd.edu/

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

20th USENIX Security Symposium (USENIX Security '11)

August 10–12, 2011, San Francisco, CA, USA
http://www.usenix.org/sec11

Workshops co-located with USENIX Security '11 include:

2011 Electronic Voting Technology Workshop/ Workshop on Trustworthy Elections (EVT/ WOTE '11)
SPONSORED BY USENIX, ACCURATE, AND IAVOSS
August 8–9, 2011
http://www.usenix.org/evtwote11

4th Workshop on Cyber Security Experimentation and Test (CSET '11)
August 8, 2011
http://www.usenix.org/cset11

USENIX Workshop on Free and Open Communications on the Internet (FOCI '11)
August 8, 2011
http://www.usenix.org/foci11

5th USENIX Workshop on Offensive Technologies (WOOT '11)
August 8, 2011
http://www.usenix.org/woot11

2nd USENIX Workshop on Health Security and Privacy (HealthSec '11)
August 9, 2011
http://www.usenix.org/healthsec11

6th USENIX Workshop on Hot Topics in Security (HotSec '11)
August 9, 2011
http://www.usenix.org/hotsec11

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

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

SPONSORED BY USENIX IN COOPERATION WITH LOPSA AND SNIA

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

USENIX ;login:

JUNE 2011, VOL. 36, NO. 3

OPINION

Musings RIK FARROW 2

FILE SYSTEMS

Apache Hadoop: The Scalability Update KONSTANTIN V. SHVACHKO 7

Building a Cloud File System JEFF DARCY 14

HARDWARE

System Impacts of Storage Trends: Hard Errors and Testability STEVEN R. HETZLER . . . 22

SYSADMIN

Testing the Transport Side of Highly Available Hosts STUART KENDRICK 33

COLUMNS

Practical Perl Tools: Give as Good as You Get, My Tiny Dancer DAVID N. BLANK-EDELMAN 44

iVoyeur: A Handful of Gems DAVE JOSEPHSEN 50

/dev/random ROBERT G. FERRELL 54

BOOKS

Book Reviews ELIZABETH ZWICKY, WITH EVAN TERAN AND RYAN MACARTHUR 56

CONFERENCES

FAST '11: 9th USENIX Conference on File and Storage Technologies. 59

NSDI '11: 8th USENIX Symposium on Networked Systems Design and Implementation 80

LEET '11: 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats 100

Hot-ICE '11: Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services 106

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.

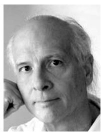
$90 of each member's annual dues is for a subscription to *;login:*. Subscriptions for nonmembers are $125 per year. Periodicals postage paid at Berkeley, CA, and additional offices.

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

I just got back from NSDI in Boston. Wintry weather (where's spring?) and interesting talks, as usual. I really liked the LEET talks [1] about spam and botnets. The scale of these operations is just amazing. It is also amazing to learn just what can be done with a handful of servers, even if they are working to send out over a trillion spam emails. The NSDI talks also covered issues of scalability, with one of the best papers describing how GPUs can be used to support parallel SSL operations.

Scalability of today's systems remains an intriguing question, one that Konstantin Shvachko takes on in this issue. Konstantin has worked on the largest known Hadoop cluster, at Yahoo!, and is now working to build Hadoop clusters at eBay. In his article, he examines the limits to scalability.

The Factory Model

I am always fascinated—like many computer enthusiasts, I imagine—by the latest offerings of Intel and AMD. CPU clock speeds have reached approximate upper limits, and there are real reasons for this. But it is hard to think clearly about these limits, when processors and I/O busses are just so darn fast. Clock speeds in gigahertz, where one gigahertz means a cycle time of one nanosecond, are far afield from the human sensory realm. So I decided to think about a factory model instead.

My imaginary factory has an assembly line that can crank out 3.4 thousand widgets per hour—as long as you keep it supplied with parts. Just to get it to start producing finished widgets, you have to feed it the parts it needs and keep the production pipeline as full as you can. And there's a problem with my factory: the methods for keeping the assembly line fed with parts is nowhere near as fast as the assembly line itself.

The assembly line is supplied by a hierarchy of specialized feeds. Those next to the assembly line work three times slower than the assembly line itself—and this feed device is often missing the desired parts. The next level in the hierarchy is 10 times slower than the assembly line, and while it is much bigger than the first-level feed, it too may often not contain the needed parts.

The level beyond resembles a just-in-time warehouse, with huge arrays of shelves and automated devices that bring the desired parts to the assembly line feeds. But this huge warehouse can be over 100 times slower. Sometimes the parts required can be delivered in batches, and getting the first batch of parts is only 60 times slower than the assembly line. Additional parts from the same part of the warehouse can be delivered twice as fast, but that is, at best, still half the speed of the second-level feed. And if the parts required come from more random parts of the warehouse, getting each part is always 60 times slower.

If our modern, just-in-time warehousing fails, and the factory needs a part that is not there, it is going to take thousands of times longer to arrive in the warehouse (before it can be delivered to the feeders for the assembly line). Thus, even though our assembly line can crank out 3.4 thousand widgets per hour, it will never actually do so. It can only produce widgets as fast as the slowest part of the part feeding system. Hopefully, the parts will always be in the warehouse, and the assembly line can churn out perhaps 600 widgets per hour.

Oh, I forgot to mention that we actually have four identical assembly lines all sharing the upper-level feed and warehouse supplies, so their total production has the same limitations as the single assembly line.

Allegorically Speaking

I modeled my factory on an Intel Sandy Bridge CPU, using the LGA 1155 socket [2]. This CPU sports clock speeds from 1.4 to 3.4 GHz. While the Intel spec sheet says it can perform up to 2.5 billion memory transfers per second, that is best case because of how DRAM works. Actually, this is sort of like saying you have a car that can go 200 miles per hour, but only for a second at a time. The rest of the time it cruises at 65 mph.

The elaborate "feed" system in my factory represents the memory hierarchy: registers, L1 and L3 caches, DRAM, and, finally, the dreadfully slow (by comparison) hard disk and network. Only registers can provide data at CPU clock speeds. Fetching data from L1 cache will take three clock cycles, and from L3 (which is up to 100 times larger than L1) takes many more clock cycles. It is not that the memory in cache is slower than register memory: both are built of static RAM (SRAM). It just takes longer to discover if the data you need is in a particular cache, and the larger the cache, the longer it takes to search for a hit in the cache.

DRAM is different from SRAM. SRAM requires five transistors to hold one bit of memory, but DRAM only requires one transistor and one capacitor. DRAM both has a smaller footprint and requires less energy than SRAM, making it suitable for bulk storage (the warehouse in my analogy). But it is much slower than SRAM for several reasons, including how DRAM is addressed: the address is supplied in two parts, the row, then the column, each with multiple nanoseconds of latency. Then the actual data must be "sensed" before it can be read. So reading data from a random part of our memory warehouse can take up to tens of nanoseconds.
In "burst mode" (our car that can go really fast for a short time), as long as the row address remains unchanged, DRAM can provide 8 or 16 bytes every 10 nanoseconds. Once the row address is changed, there is an additional waiting period before burst mode can start again. And if random access is really random, forget about burst mode (and the 200 mph car).

Our warehouse of DRAM gets supplied by disk or network. Practically speaking (that is, no 200 mph car), each disk can supply 60 MB/s, and a one gigabit network can supply 119 MB/s. When you compare these data rates to DRAM data rates, they are many times slower. On top of that, disk and network latency is measured in milliseconds, that is, in millions of nanoseconds. For a CPU clocked at 3.4 GHz, about .29 nanoseconds per clock tick, a latency of 10 milliseconds translates into 340,000 clock cycles. That's a very long time to wait when you are a CPU.

As a friend from Intel recently told me, it's all about memory. If you can provide enough memory bandwidth to keep the CPU busy, you are doing great. The best rate of speed for the memory controller used in our example is 21.3 GB/s. But if our memory yields 16 bytes every 10 nanoseconds, even for a short amount of time (one page), that's 1.6 GB/s. Ooops. Even if I am off by a factor of two, that is still far below the maximum rate of the memory controller, and certainly far below the rate of registers and cache, especially when you consider that there may be four cores of CPU, each waiting for four or eight bytes of data.

Before you decide not to buy that really fast CPU, keep in mind that I am making a point through exaggeration here. CPUs can process more data than memory can provide, especially fast, manycore processors. But many applications don't require reading and writing vast amounts of random data, or even better, they feature locality of both code and data. These applications do perform better on faster CPUs. Also, the threading provided in modern processors also helps hide latency by allowing the processors to work on something else instead of idly waiting for data. So things are not as bad as I made them seem.

The memory hierarchy is real and is the biggest challenge facing hardware, compiler, and system designers and anyone who wants to tune an application so that it performs as fast as possible. Otherwise, you have idle assembly lines, that is, CPUs.

Clusters and Clouds

Hadoop provides a mechanism for dealing with the huge disparity between data and processor speed when you want to process massive amounts of data. First, you focus on having the data you want to process local to the processor by dividing that huge amount of data into many slices stored on many disks. Then you process those slices in parallel. Divide and conquer, the way forward in the days of Caesar, and still useful today with big data.

Konstantin brings us up to date on scalability issues with Hadoop. Hadoop has become increasingly popular as a method for processing enormous amounts of data. It is still not the fastest way—for that, you need more advanced methods, such as Google's Percolator [4]. Konstantin brings a great amount of experience to the table, as he has worked with Hadoop and its data storage systems for many years.

Jeff Darcy describes a project he has been working on, CloudFS. Jeff provides a very thoughtful description of the challenges facing any cloud file system. Then he explains how he deals with these problems by leveraging GlusterFS to create CloudFS, both open source projects. I first heard about CloudFS during a FUDCon, where Jeff spoke just as eloquently about the issues facing scalable, secure, manageable, and reliable cloud storage.

I met Steve Hetzler after the tutorial he presented at FAST. Steve is an IBM Fellow, and not one to mince words. And Steve doesn't like how disk error rates are presented by storage companies. After all, how many of you read bits from your hard drives? Last time I checked, your only option was to read sectors, and you could only do that at a certain maximum rate, based on IOPS.

Steve also casts his steely gaze upon solid-state devices (SSDs). While flash appears to solve many issues for I/O bound applications, its endurance is an open question and, according to Steve, not one we can answer any time soon.

Stuart Kendrick volunteered to write about the hazards of high availability (HA). Stuart has worked for many years on servers designed to be highly available but which turned out to be highly unavailable (HU) instead. He shares his experience in testing HA in the hopes that you too might have HA instead of HU.

David Blank-Edelman dances around an elegant module for serving up HTTP content. Two years ago, David wrote about the CGI::Application framework [5]. This time, David introduces Dancer, another framework that strives to keep things simple. Dancer is a port of Sinatra (who is not a dancer), a Ruby framework that is simpler than Rails. Just the thing if you know Perl and need something like Rails, but simpler.

Dave Josephson dazzles us with a handful of monitoring gems, but not until he has had a chance to rant. And what a wonderful rant (unless you or your significant other is an auditor). Dave shares his hard-won experience in monitoring distributed, Internet-facing servers with four great monitoring tools.

Robert Ferrell brings a (literally) fevered imagination to the subject of viruses.

Peter Galvin decided not to write for this issue. I expect he will be back by August, but who can tell what the future may bring?

We close out this issue with summaries from FAST '11 and NSDI '11. Having taken a peek at the USENIX Web logs, I can tell you that the FAST paper on proximal writes by several NetApps employees is extraordinarily popular. I personally learned a lot about deduplication, working with flash, dealing with very large directories, and more by attending FAST and reading the summaries. At NSDI, there were no invited talks. I liked the CIEL paper (no surprise there) and the SSLShader paper, where researchers describe creating an SSL accelerator with an older GPU. The performance limitation for SSLShader is not the GPU; the memory bandwidth available is the bottleneck. What a surprise (not). As always, you can download any paper from FAST or NSDI, as well as watch videos of presentations or view the slides [6].

I don't want to denigrate the performance improvements we have seen in processor and system design. I mention Intel Core i5 because I am using one in my new desktop. Not only does it provide me with a vast increase in CPU performance, I finally have decent graphics performance as well. Not that I really have any great need for either, to be honest. My 8 MHz 68k-based System V server kept up with my typing just fine. But sometimes it is nice to play with Google Earth. I've found that looking at maps is one thing, but flying along over familiar routes grants me an entire new perspective on how the world around me appears in reality. Or at least, something closer to reality than my own mental maps.

Designing balanced systems is still the way forward. We now have incredibly powerful systems, even if their processors far outpace the I/O that feeds them. The path forward includes new systems and software designs that play to the strengths of the hardware we have, especially through designs that emphasize the most efficient memory footprint possible or, at least, as much sequential I/O as possible and clusters of systems for dealing with Big Data.

References

[1] 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats: http://www.usenix.org/events/leet11/tech/.

[2] Intel Core i5 CPU: http://ark.intel.com/Product.aspx?id=47341.

[3] Intel Core i5-750 and i7-870 processors, The Tech Report: http://techreport.com/articles.x/17545/5.

[4] Daniel Peng and Frank Dabek, "Large-Scale Incremental Processing Using Distributed Transactions and Notifications," OSDI '10: http://www.usenix.org/events/osdi10/tech/full_papers/Peng.pdf.

[5] David N. Blank-Edelman, "Practical Perl Tools: Scratch the Webapp Itch with CGI::Application, Part 1," *;login:,* vol. 34, no. 4, August 2009: http://www.usenix.org/publications/login/2009-08/pdfs/blank-edelman.pdf.

[6] 9th USENIX Conference on File and Storage Technologies (FAST '11), Technical Sessions: http://www.usenix.org/events/fast11/tech/.

Apache Hadoop

The Scalability Update

KONSTANTIN V. SHVACHKO

Konstantin V. Shvachko is a veteran Hadoop developer. He is a principal Hadoop architect at eBay. Konstantin specializes in efficient data structures and algorithms for large-scale distributed storage systems. He discovered a new type of balanced trees, S-trees, for optimal indexing of unstructured data, and he was a primary developer of an S-tree-based Linux file system, treeFS, a prototype of reiserFS. Konstantin holds a PhD in computer science from Moscow State University, Russia. He is also a member of the Project Management Committee for Apache Hadoop.
kshvachko@ebay.com

Scalability is one of the primary forces driving popularity and adoption of the Apache Hadoop project. A typical use case for Hadoop is an emerging Web site starting to run a five-node Hadoop cluster and then gradually increasing it to hundreds of nodes as business grows.

Last year *;login:* published my article [12] summarizing one aspect of Hadoop scalability, namely, the limits of scalability of the Hadoop Distributed File System [13]. There are many other dimensions to Hadoop scalability. Here I would like to address some of them.

Source of Innovation

This has been an eventful year for Apache Hadoop [1]. The project has emerged as a data mining platform, becoming an industry standard for Big Data. Apache Hadoop is successfully used in science and a variety of industries. Scientific applications include mathematics, high energy physics, astronomy, genetics, and oceanography. The platform adoption has scaled far beyond information technology—its original target area—into most industries, excluding only hunting and fishing, but probably not for long.

Started as a computational platform for search engines, Apache Hadoop is now used for data warehousing, behavioral analysis, recommendation engines, cryptanalysis, meteorology, fraud and spam detection, natural language processing, genomic analysis, image processing, semantic text analysis, etc.

Apache Hadoop was used to compute the record two quadrillionth (10^{15}) digit of π [15], which turned out to be 0, and helped IBM's Watson to win on *Jeopardy* in the "Man versus Machine race," as media presented it. Recognized for its influence on technological innovation, the Apache Hadoop project has won the 2011 MediaGuardian Innovation Award over nominees such as iPad and WikiLeaks.

While Hadoop was the driving force of technological innovation, its internal innovation has been on a rather slow rise. Not to imply that it's inert. On the contrary, a lot of development is going on in the field. Hard to underestimate the value of implementing security for Hadoop, or building analytical tools with Hadoop, or stabilizing internal company releases, which are essential for businesses running Hadoop. However, due to the lack of dominating gravitational force, these highly dedicated activities did not materialize in common production-ready releases, uniformly supported by different developer groups, which would have consolidated the project.

Size Matters

Incremental improvements in HDFS performance led to gradual growth of Hadoop clusters over the year. The largest Hadoop clusters are run by Yahoo and Facebook, with eBay catching up in a hurry.

- Yahoo reportedly ran numerous clusters having 4000+ nodes with four 1 TB drives per node, 15 PB of total storage capacity, 70 million files, and 80 million blocks using 50 GB NameNode heap.
- Facebook's 2000-node warehouse cluster [2] is provisioned for 21 PB of total storage capacity. Extrapolating the announced growth rate, its namespace should have close to 200 million objects (files + blocks) by now, but an immense 108 GB heap should allow room for close to 400 million objects.
- eBay runs a 700-node cluster. Each node has 24 TB of local disk storage, 72 GB of RAM, and a 12-core CPU. Total cluster size is 16 PB. It is configured to run 26,000 MapReduce tasks simultaneously.

As observed in the past, the average size of HDFS files is decreasing. This trend is sustainable as the cluster grows and becomes available for a larger variety of applications. The phenomenon is characterized by the decreasing block-to-file ratio, which has dropped from 2 in 2008 to 1.5 in 2009 and to 1.1 today.

DataNode's local storage capacity is increasing as cheaper 2 TB drives fall under the category of commodity hardware. Combined with the growing number of cores per processor and larger RAM sizes, this leads to more compact but powerful clusters.

While the clusters become more compact, the network bandwidth becomes the limiting factor of the cluster performance. Typical network bandwidth between nodes on the same rack is 1 Gbps, which converts into a 119 MB/s data transfer rate. The read rate for a single disk drive usually exceeds 60 MB/s. This means that if one runs a read-intensive job, such as DFSIO-read, then in order to saturate the network capacity of a single node, it is enough to have only two tasks reading from two different drives on that node. In practice the I/O rate of a combination of random reads and writes is lower, and only a fraction of those I/Os results in actual data transfers. Based on observations on busy Hadoop clusters, the average data transfer rate per client is 10 MB/s. In this case 12 clients accessing data from 12 different drives of a single node will saturate the node's network capacity. Therefore, adding more drives will not increase the aggregate throughput of the cluster, which means that dense local storage is beneficial only for the cluster that stores a vast amount of rarely accessed data, as in data warehouses.

Decreasing file sizes and compaction of the clusters: both of these trends drive the demand for higher Hadoop scalability.

MapReduce: Scaling the Framework

The simplicity of the MapReduce [5] computational model combined with its power to incorporate and utilize distributed resources of commodity hardware is the second driving force of Hadoop's popularity.

A MapReduce job is a two-stage computation, with each stage defined by a plain Java (or C++, or Python) program—a task. The input data for each stage is distributed across the nodes of the cluster, and the same task is run against different blocks of the input data on the node containing the data block. The job produces distributed output. This type of computation minimizes data transfer by moving computation to data and not vice versa.

The Apache Hadoop MapReduce framework has reportedly reached its scalability limit at 40,000 clients simultaneously running on the cluster. This corresponds to a 4,000-node cluster with 10 MapReduce clients—slots, in Hadoop terminology—per node.

The implementation of the Hadoop MapReduce framework follows the same single master architecture as HDFS. The single master is called JobTracker. It shepherds the distributed herd of slaves called TaskTrackers. The JobTracker serves two primary functions:

1. Job scheduling and resource allocation
2. Job monitoring and job life-cycle coordination

The first function is fundamentally centralized, but the second one is not. Coordination of many jobs running on thousands of TaskTrackers makes the single JobTracker a constraining resource for the entire cluster.

There are ongoing efforts ([10], [7]) to improve scalability of the MapReduce engine by delegating the coordinating function to different cluster nodes for different jobs. That way, even if the JobTracker fails the jobs will continue to run, as their lifecycles are controlled by other nodes.

This also intends to address another weak point of today's implementation—the static partitioning of cluster resources. In current MapReduce architecture the cluster is divided into a fixed number of map and reduce slots, which are uniformly configured per node. Each slot can be used for tasks of the assigned type (map or reduce) only and therefore cannot be reallocated to another type if the demand for the latter increases. Static cluster configuration also does not take into account the amount of resources—RAM, CPU, disk space, network bandwidth—required for different tasks, which may lead to underutilized resource usage for some tasks and starvation for others.

HDFS: Static Partitioning

Hadoop deployments have reached the architectural limit. With HDFS clusters running at capacity, the only direction for growth is a horizontal increase in the number of clusters. The demand to support smaller files, and the evolutionary growth of storage devices and computational power of servers, which allows aggregating more resources per cubic foot, are the two major factors contributing to the demand for higher scalability in distributed storage systems.

Current HDFS architecture by design assumes that a single server, the NameNode, dedicated to maintaining the file system metadata consisting of files, directories, and blocks, controls the work of other cluster nodes, DataNodes, handling the actual data blocks of files. The system is designed to scale linearly on the number of DataNodes, as they can process data transfers independently of each other. However, the NameNode is a single source of metadata information.

The HDFS scalability is limited solely by NameNode resources [12]. In order to process metadata requests from thousands of clients efficiently, NameNode keeps the entire namespace in memory. *The amount of RAM allocated for the NameNode limits the size of the cluster.*

The number of active clients is proportional to the size of the cluster. As the cluster grows, the increasing number of clients provides higher load on the NameNode. Being a single point of entry, *the NameNode's efficiency limits the aggregate cluster performance.*

It is clear that further scaling of HDFS requires a scalable architecture for its namespace.

One approach, called Federation [11], is based on the idea that multiple independent namespaces can share a common pool of DataNodes as a block storage layer. The namespaces representing isolated file systems, called volumes, are maintained by dedicated NameNodes—one per volume—and evolve independently of each other. Each DataNode maintains blocks of multiple volumes and reports those blocks to corresponding NameNodes. The cluster then is defined as a family of volumes sharing the same pool of DataNodes.

In order to conceive the isolation of the file systems from clients, the volumes are "federated" under client-side mount tables. The client-side mount table is a virtual file system, called ViewFS, which provides a common view of the cluster for a group of clients unified by common cause. ViewFS in general is a sequence of symbolic links—fully qualified HDFS paths—that can be passed over via a job configuration file to all tasks of that job in order to supply them with a common view of the world.

A federated cluster can store more data and handle more clients, because it has multiple NameNodes. However, each individual NameNode is subject to the same limits and shortcomings, such as lack of High Availability (HA), as a non-federated one.

The federated approach provides a static partitioning of the federated namespace. If one volume grows faster than the other and the corresponding NameNode reaches the limit, its resources cannot be dynamically repartitioned among other NameNodes except by manually copying files between file systems.

The Distributed Namespace Challenge

On the next evolutionary step, HDFS should become a distributed highly available file system without a single point of failure, which can:

- Store hundreds of billions of objects
- Support millions of concurrent clients
- Maintain an exabyte (10^{18}) of total storage capacity

The main motivation for building such a system is the ability to *grow the namespace*. The current namespace limit is 100 million files. Static partitioning will scale the federated namespace to billions of files. Estimates show that implementation of a dynamically partitioned namespace will be able to support 100 billion objects.

Service continuation and availability is another strong motivation for the system. A big HDFS installation with a NameNode operating in a large JVM is vulnerable to frequent full garbage collections, which may take the NameNode out of service for several minutes. “Bad” clients, producing a high number of metadata operations, can saturate the NameNode, effectively making it unavailable for other tasks. And, finally, a failure of the NameNode makes the file system inaccessible for up to an hour—the time it takes to restart the NameNode.

Building a large system compared to maintaining a number of smaller ones simplifies its operability. Currently, the effort of operating clusters of 400 nodes is roughly the same as for clusters of 4000 nodes. The cost of maintaining different clusters is proportional to the number of clusters.

Building such a system from scratch can take years, as this is how long creating viable file systems takes. A simpler approach is to construct the system from existing reliable components. For HDFS, one needs to find a component able to maintain a distributed namespace. It turns out that the Hadoop family has just the one.

Apache HBase [14] organizes data into big, sparse, loosely structured tables. The elements of a table are rows, having unique row keys. An HBase table can have an arbitrary number of columns, grouped into a small predefined number of column families. The columns can be created dynamically, but not the column families. Tables are partitioned into regions—horizontally across rows and vertically across column families. Regions are stored as (compressed) files in HDFS. HBase runs on a cluster of machines called region servers. Each region server caches a number of regions and serves this data to HBase clients. The goal is to provide a structured yet flexible presentation of data for random, near real-time read and write access to very big tables, consisting of billions of rows and millions of columns. Apache HBase is an implementation of Google’s BigTable [3].

HBase can be used to maintain the file system namespace, making it a scalable replacement for the HDFS’s NameNode. With this approach, files and directories become rows of a very large HBase table representing the entire file system. The file blocks will still be stored on and served from DataNodes. This will preserve the decoupling of data and metadata—one of the key principles of HDFS architecture.

Google used this approach to build its next-generation GFS Colossus [4]. Prototypes such as VoldFS and CassFS [6] have been built based on the same principles but using a competing database, with HBase metadata stores Voldemort and Cassandra, respectively. Pomegranate [9] implements it own tabular storage utilizing the same idea.

Of the many design challenges facing such systems, probably the main two are:

- Namespace partitioning
- Atomic rename

Namespace partitioning is the problem of mapping the hierarchical file system tree to the flat table structure. A naïve approach is to simply hash file paths, with the hash value defining the partition the file belongs to. This approach lacks the important principle of locality, as a simple listing of a directory requires accessing multiple partitions, which may be located on different nodes. Another approach is a Ceph-like [16] partitioning into full subtrees. This provides good *locality of reference*, even somewhat too good, assuming that the degree of locality of files in a tree is proportional to the number of their common ancestors. Somewhere in between is an approach which partitions the tree into small fixed-height tiles. For example, for height 2 and for a given directory D the tile contains D itself, all its children, and all grandchildren. The tiles can then be arbitrarily combined into partitions. This is similar to the way reiserFS partitions the namespace into fixed-size blocks.

Renaming is tightly related to the partitioning problem, because when the row (file) keys are based on file paths, even a simple case of rename—that is, changing a file's local name—may move the file into a different partition. And a directory rename in that case can lead to a massive cross-partition relocation of files in the subtree. Therefore, row keys should be based on unique file IDs (inode numbers) rather than paths. This still leaves unsolved a more general case of rename, required by POSIX semantics: an atomic move of a file from one directory to another. The problem is hard, as it requires a consistent update of multiple partitions potentially distributed across the cluster. A solution involves use of PAXOS-like [8] consensus-building algorithms. A "lazy" approach is to sacrifice this functionality, relaxing the semantics of rename to support only the simple case (in-place rename) in favor of higher scalability. Applications relying on the atomicity of cross directory moves will have to implement it internally. In many cases this is easier than building a generic solution.

The Final Dimension

An open source platform like Apache Hadoop,usually provides a generic tool to do things. Given the tool, companies and organizations initially benefiting from the ability to quickly adopt the system for their business use cases then tend to continue investing in testing, performance tuning, and refining the tool for their production needs. Ideally, this drives innovation, and the platform evolves into a highly tunable and adaptive system with various controls to make it fit many practical use cases.

Flexibility of the system adds another axis to the Hadoop scalability universe. I would like to thank my numerous colleagues in the Apache Hadoop community for contributing to this multidimensional endeavor.

References

[1] Apache Hadoop. http://hadoop.apache.org/.

[2] D. Borthakur, "Facebook Has the World's Largest Hadoop Cluster!": http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html.

[3] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," *Proceedings of the 7th Symposium on Operating System Design and Implementation,* November 2006.

[4] J. Dean, "Large-Scale Distributed Systems at Google: Current Systems and Future Directions," Keynote at Large-Scale Distributed Systems and Middleware, October 2009.

[5] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Proceedings of the 6th Symposium on Operating Systems Design and Implementation,* December 2004.

[6] J. Darcy, VoldFS and CassFS: https://github.com/jdarcy/VoldFS, https://github .com/jdarcy/CassFS.

[7] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A.D. Joseph, R. Katz, S. Shenker, and I. Stoica, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," Proceedings of NSDI 'll: 8th USENIX Symposium on Networked Systems Design and Implementation, March 2011.

[8] L. Lamport, "The Part-Time Parliament," *ACM Transactions on Computer Systems,* vol. 16, no. 2, May 1998, pp. 133–169.

[9] C. Ma, Pomegranate: https://github.com/macan/Pomegranate/wiki.

[10] A.C. Murthy, "The Next Generation of Apache Hadoop MapReduce," Yahoo! Developer Network Blog, February 14, 2011: http://developer.yahoo.com/blogs/hadoop/posts/2011/02/mapreduce-nextgen/.

[11] S. Radia, S. Srinivas, "Scaling HDFS Cluster Using Namenode Federation," HDFS-1052, August 2010: https://issues.apache.org/jira/secure/attachment/12453067/high-level-design.pdf.

[12] K.V. Shvachko, "HDFS Scalability: The Limits to Growth," *;login:*, vol. 35, no. 2, April 2010, pp. 6–16.

[13] K.V. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," *Proceedings of Symposium on Mass Storage Systems and Technologies*, May 2010.

[14] Apache, *The Apache HBase Book*, October 2010: http://hbase.apache.org/book.html.

[15] Tsz-Wo Sze, "The Two Quadrillionth Bit of Pi Is 0! Distributed Computation of Pi with Apache Hadoop," arXiv:1008.3171v2 [cs.DC], August 2010.

[16] S. Weil, S. Brandt, E. Miller, D. Long, and C. Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System," *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, November 2006.

Building a Cloud File System

JEFF DARCY

Jeff Darcy has been working on network and distributed storage since that meant DECnet and NFS version 2 in the early '90s. Since then he has been a key developer on the MPFS project at EMC, product architect at Revivio, and well-known blogger on various related topics. He is currently the founder and technical leader of the CloudFS project at Red Hat.
jeff@pl.atyp.us

Cloud file systems must address challenges that are not addressed by traditional network or distributed file systems. These challenges mostly revolve around isolation, identity, and privacy, but also include features such as adaptability to frequent changes in demand for capacity or capability. In this article, I'll elaborate on some of these challenges and describe how one project, CloudFS, attempts to address them.

Cloud Problems

To understand the requirements for a cloud file system, one must consider the special properties of cloud computing. While everyone who writes about the subject has their own list of properties that define "the cloud" or make it special, one property of particular relevance is that a cloud is shared. When you use resources in a cloud, especially a public cloud, you share the underlying physical resources with others whose intentions and resource needs are totally unknown to you. Typically, this sharing is between accounts rather than actual users; each *tenant* holding an account with a cloud provider might actually be a complex enterprise representing their own large and ever-changing set of end users. In a public cloud, such as those at Amazon or Rackspace, tenants are likely to be companies. In a private cloud, tenants might be departments or projects. In all of these cases, though, the important thing is that tenants don't trust each other. They might not even trust the cloud provider very much. In many situations—especially those involving medical or financial information—they're not even legally allowed to extend such trust to others. This extreme lack of trust implies that a cloud file system must ensure that tenants are isolated from one another to a far greater degree than would be the case using a traditional distributed file system.

Processors and disks aren't the only resources that are shared in the cloud. Another often-overlooked resource that can also be the subject of conflict is identity. Every online identity lives in a certain space within which it has some specific meaning. Once, each physical machine had its own identity space because each ID conferred only access to local resources. Many have returned to that model within their virtual machines today, because machine virtualization provides a similar level of isolation. In fact, it's entirely possible to run a traditional distributed file system across multiple nodes allocated within a cloud, relying on the cloud's existing machine- and network-level isolation to protect storage as well. Unfortunately, this approach is insufficient when the file system must be deployed as a shared service. A cloud provider might favor such a deployment to capitalize on the resource-utilization efficiency that comes from sharing resources between tenants with non-correlated peak demands (which is a core value proposition of cloud computing generally) and/or to provide an "added attraction" in a public cloud. The problem with identity in such a shared deployment is that it introduces a possibility of conflict involving reuse of the same ID by different tenants (Figure 1). If tenants are responsible for assigning IDs themselves, then two tenants can assign the same ID to different users—either intentionally or maliciously, whether IDs are numbers or strings—and present that ID to the shared service. As a result, client-provided IDs are insufficient as a basis for authentication or authorization in such a service.

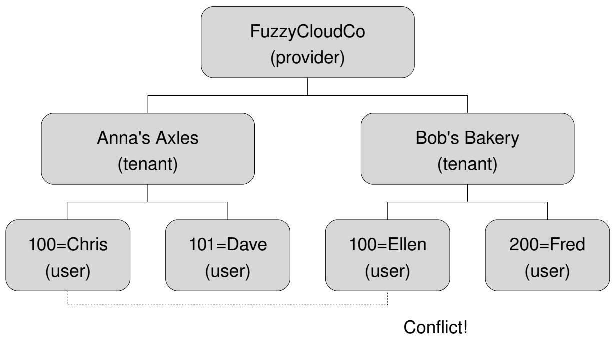


Figure 1: Illustration of a UID conflict between tenants. Anna's Axles and Bob's Bakery are unrelated tenants, but their users Chris and Ellen share the same UID.

One common approach to this problem in traditional environments is to say that machines in a network cannot simply assign their own user IDs. Instead, identity management is "outsourced" to something like Kerberos, establishing a new, centrally administered ID space which can be shared between clients and servers. How might this be applied to the cloud? Consider the scale of such a service at a large public cloud provider. There might be thousands of tenants, each with thousands or even millions of users. How many new user registrations per day would that be? How many ID-mapping operations per minute? Even if such a system could be implemented cost-effectively, tenants would resist any requirement to register their own users with the provider's identity service. They would object for privacy reasons, and also for operational-complexity reasons. This is especially true if they're already running their own identity management systems internally and would thus be forced to support two such systems side by side. Centralized identity management is as inapplicable in the cloud as client-side identity management.

If both of these options are precluded, what's left? The answer is the same as it has been for domain names, or for email addresses which depend on them—delegation. If a flat ID space won't work, use a hierarchical one and delegate smaller pieces to lower-level authorities—in this case tenants. Identity in the cloud has to be contextual, not just "user X" but "user X within tenant Y" as distinct from "user X within tenant Z" or any similar combination. Tenants would have complete freedom to manage their own identity space on their own machines, just as they do when using only their own services and just as they do with subdomains even in a shared higher-level domain. The only burden on the provider would be to maintain any stored end-user identities, such as the UID and GID associated with a file, so that it can be sent back later. It is not responsible for allocating these identities or for discovering them beyond the "which tenant" level.

Resource sharing might be the most important problem with which a cloud file system must contend, but it's far from the only one. Adaptability is another important feature for any cloud service, and providing that adaptability usually involves some kind of virtualization. For example, hypervisors allow many virtual machines to exist within one physical machine. In a similar way, a tenant's virtual file system might be carved out of a much larger physical file system spread across many machines and shared among many tenants. Just as the virtual machine's processor count or memory size can be changed at a moment's notice, so can the virtual file system's size and performance profile and redundancy characteristics. Besides being convenient, this preserves the basic cloud value proposition of "pay as you go" access to efficiently allocated hardware.

There, in a nutshell, are the problems a cloud file system must address: shared resources introducing issues of privacy and isolation, hierarchical identity, and adaptability to changing user needs. CloudFS, which will be discussed in the following section, represents one attempt to address these issues.

File System Virtualization

As mentioned earlier, it could be argued that current distributed file systems adequately solve the problems they were designed to solve. Examples such as GlusterFS [2], PVFS [3], and Ceph [4] allow the capacity and performance of multiple servers to be harnessed within a single file system. They handle things like distributing files across servers, striping data within files, replicating files, adding and removing servers, rebalancing and repairing, etc. Most importantly, they provide a POSIX file system API to all of this functionality, or "POSIX enough" to keep operating systems and most users happy. What some of them also provide is a convenient way to "mix and match" or even add functionality without having to rewrite what's already there. In GlusterFS, for example, most functionality—e.g., replication, striping, client and server protocols—exists in the form of "translators" which convert a higher-level file system request into one or more lower-level requests using the same API. These translators allow many local file systems on many servers to be combined and then recombined, providing layers of increasing functionality until they all combine into one translator providing one unified view on a client machine. This structure is illustrated in Figure 2.

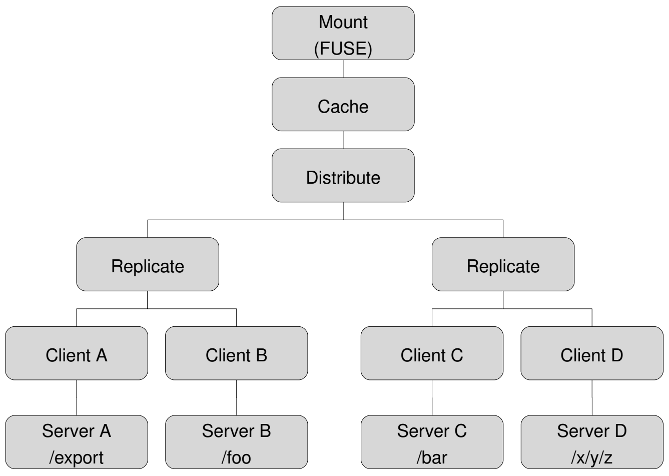


Figure 2: Translator structure in a typical GlusterFS deployment. Translators convert I/O requests from "above" into one or more "below" using the same API.

What we see in Figure 2 is several local file systems on several servers ("bricks" in GlusterFS terminology) being exported to clients, which first combine them into replica groups and then distribute files across those groups. Next, a cache translator is added; this is an example of a translator with a 1:1 mapping between input and output requests. Finally, the combined result is presented as a single mountable file system. Because they use a common interface, translators can readily be stacked in different orders and even be moved from one side of the client/server divide to the other. For example, the caching in the structure in Figure 2 could easily be done first instead of last, on the server side instead of the client.

When one seeks to add new functionality which is itself quite complex, as is the case with CloudFS, this kind of modularity and flexibility can speed development considerably. Because CloudFS is based on GlusterFS, this makes it relatively easy to provide a "virtual" file system for each tenant across a common set of "bricks" without having to build a whole new file system from scratch or modify the internals of a working and widely accepted file system.

In GlusterFS, a client's view is effectively the union of its component bricks. CloudFS takes this same set of bricks and turns each tenant's view into the union of per-tenant subdirectories on those bricks. If the whole file system is considered as a matrix of bricks along the X axis and tenants along the Y axis, then a server is managing a vertical slice and a tenant sees a horizontal one. This provides a complete and reasonably straightforward separation of each tenant's namespace (directories and files) from any other tenant's. This requires the addition of extra translators, as shown in Figure 3.

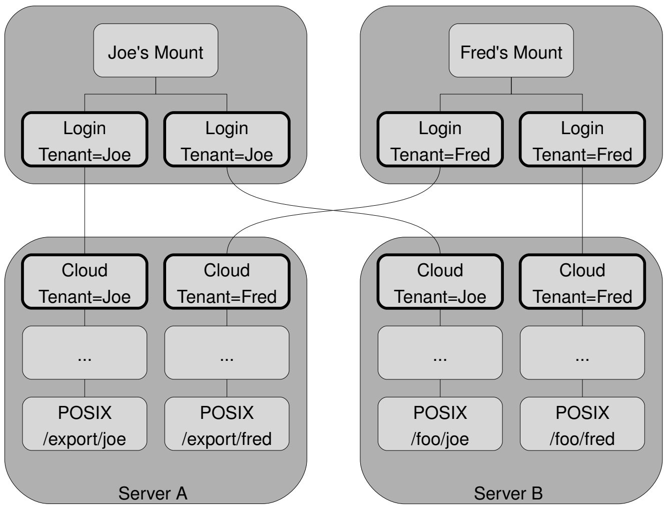


Figure 3: GlusterFS translator structure when using CloudFS. The translators with the bold outlines are provided by CloudFS.

The extra translators are added to the system by scripts which automatically rewrite both server-side and client-side configuration files according to the contents of a tenant list, including credentials for each tenant. (Actually, the credentials are identifiers for tenant-specific keys or certificates, to avoid having that sensitive information show up directly in the server or client configurations.) On the server side, the single translator stack for a brick is replaced by several per-tenant stacks. Each per-tenant stack leads down from a "cloud" translator, which provides authentication services down to a "posix" (local file system) translator, which is configured to use a per-tenant subdirectory of the original brick. On the client side, a "login" translator is added to do the client's side of the authentication handshake. The net effect of all this is to construct a tenant's horizontal slice of the aforementioned matrix for each client mounting with that tenant's configuration and credentials. In fact, each tenant might see a different horizontal slice across a different set of servers and replicate or distribute differently within its slice, providing some of the adaptability mentioned earlier.

Identity Mapping

So far we've succeeded in virtualizing each tenant's namespace. What about their identity space? A user's effective identity on a CloudFS server is actually the combination of their tenant identity (established when the tenant connected) and the tenant-provided UID. This identity can be stored directly in an inode or embedded in an extended attribute (e.g., for access control lists), or it can be associated with a request—and likewise for GIDs. To the maximum extent possible, the actual use of these values to enforce access control should be done by the kernel rather than by duplicated code in user space, but the kernel only understands a single ID space shared by all tenants. We therefore need to map a tenant ID plus a tenant-specific UID into a unique server UID for storage, and then map in the opposite direction upon retrieval. Fortunately, this mapping does not need to be coordinated across servers. Each server can safely use its own separate mapping table, populated automatically whenever it sees a new tenant/UID pair, so long as the mapping process is reversible locally. This is done by the "cloud" translator which sits at the top of each per-tenant translator stack on the server. All instances of this translator on a server share the same mapping tables, to avoid creating duplicate mappings, but otherwise (e.g., with respect to authentication) they're separate.

Privacy and Encryption

The remaining focus of CloudFS is ensuring tenant privacy, and the main tool we use for this is encryption. In fact, CloudFS needs to do two kinds of encryption: for data "in flight" on the network and for data "at rest" on disk. Techniques and tools such as TLS for doing in-flight encryption are fairly well understood, and CloudFS applies them in fairly straightforward ways, so here we'll focus on at-rest encryption.

For cloud storage of any kind, it's important for any tenant to consider whether they trust the cloud provider with their data (or with which data). "Trust" is a funny word, though, since it can apply to intentions, skills, or diligence. It's entirely possible to trust a cloud provider's intentions and skills which allow them to secure disks that are physically under their control, but not trust their diligence when those disks leave their control—e.g., when those disks are removed from service and sold. Stories about disks being sold online while still holding medical, financial, or even defense-related data are legion, and cloud providers go through a lot of disks. If you trust your cloud provider in all three of these ways, or simply don't care whether that data becomes public, then there's no need for at-rest encryption. In all other cases, though, at-rest encryption is necessary. CloudFS takes an even harder line: if the provider has keys for your data (and a surprising number of cloud-storage solutions require this), then they might as well have the data itself, so all at-rest encryption has to be done entirely by tenants using keys that only they possess.

Unfortunately, tenant- or client-side encryption presents its share of problems. Chief among these is the problem of partial-block writes. For encryption methods with the property that every byte within a (cipher) block affects the output, a read-modify-write sequence is necessary to incorporate the unwritten bytes. For other methods, the need to avoid re-using initialization vectors would require the same sort of sequence to re-encrypt the unwritten bytes (plus some mechanism to manage the ever-changing vectors). Adding authentication codes to prevent tampering involves many of the same problems. In all of these cases, some form of concurrency control is necessary to make the read-modify-write atomic even in the face of concurrent updates. Truly conflicting writes without higher-level locking can still clobber each other's data, just as they always have; this mechanism only prevents corruption that could otherwise be caused by the encryption process itself.

Currently, CloudFS handles this need with a combination of queuing and leasing. When a client needs to do a write that involves partial blocks, it first sends a request to the server to obtain a lease and retrieve the unaligned head/tail pieces. This lease is relinquished when the matching write occurs. Any conflicting writes received while the lease is valid will be queued behind the lease holder and resumed when the lease is either relinquished normally or revoked due to passage of time. Because there's no synchronous locking involved, the overhead for this approach is only the extra round trip for the initial lease acquisition—and even then, only when a write is unaligned.

Results

To validate the approaches described above, some simple tests were performed on machines available to the author through his employer. These consisted of nine machines which represent the low end of the node-count spectrum for CloudFS but also the high end of the per-node-capability range (each 24 cores, 48 GB of memory, 63 disks with hardware RAID, and 10 Gb/s Ethernet). Tests were done on Linux kernel 2.6.32-125.el6, including NFSv4 on a single server and with GlusterFS 3.1.2 on three servers. It should be noted that these are preliminary results for code still under active development. Results for higher node counts and more stable software versions will be published on the CloudFS project site [1] as they become available.

To test the effect of CloudFS's at-rest encryption on aggregate bandwidth, 1 MB writes were tested using iozone, with 12 threads per client machine. The results are shown in Chart 1. The first conclusion that might be reached from this result is that the underlying GlusterFS infrastructure is sound, outperforming NFS even on a per-server basis and easily using additional servers to scale beyond what the single NFS server could ever achieve. The second conclusion is that the encryption does exact a heavy toll on performance. However, the linear slope indicates that this is almost certainly a pure client-side bottleneck. To the extent that this is likely to be the result of insufficient parallelism within the encryption translator, this should be easily fixable. In the meantime, while performance is relatively poor, it is still adequate for many users in the cloud and scales well as clients are added.

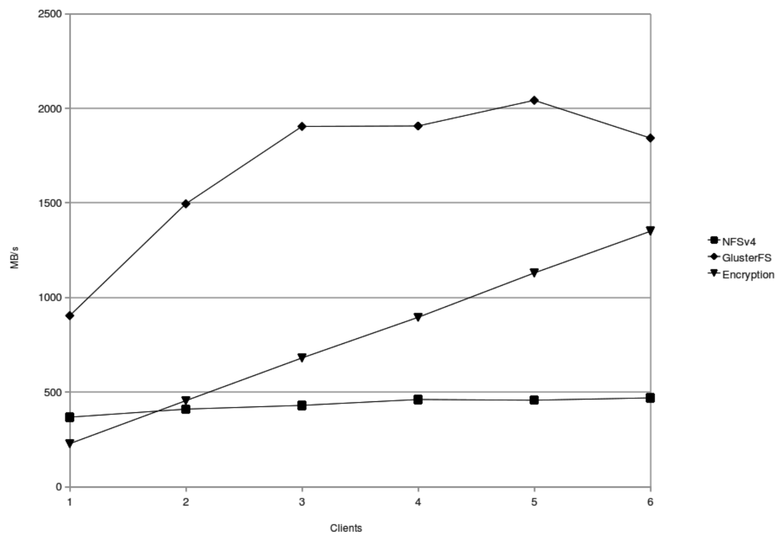


Chart 1: CloudFS encryption performance compared to NFSv4 and GlusterFS

To test the overhead of CloudFS's multi-tenant features, a different, more synchronous and metadata-intensive test was called for. In this case we used fs_mark to create many thousands of small files, again using 12 threads per client. The results are shown in Chart 2. This time, the performance is very nearly the same as for plain GlusterFS, and even better for much of the tested range. This is thought to be the result of lower contention on the servers, but bears further investigation.

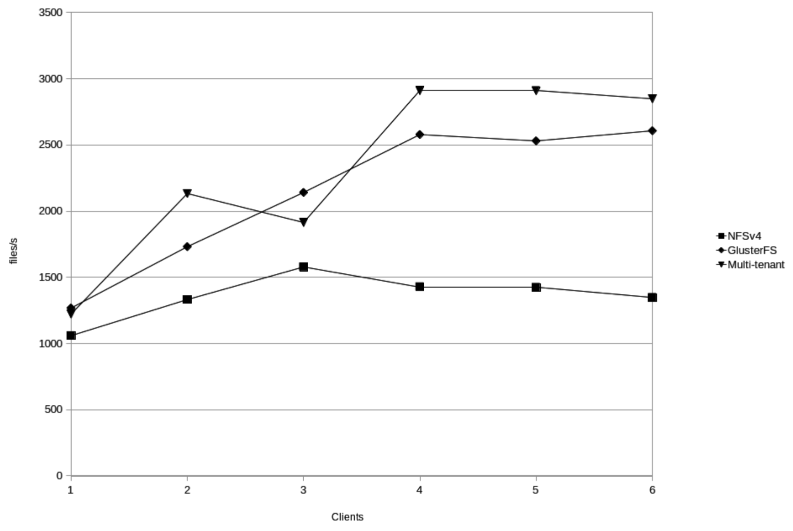


Chart 2: CloudFS multi-tenancy performance compared to NFSv4 and GlusterFS

Conclusions and Future Directions

The premises of CloudFS are that existing distributed-file system solutions already provide performance and scalability for cloud storage use and that additional features needed to make such use safe can be added in a modular fashion without excessive sacrifice in speed. Preliminary testing seems to bear out the first point quite well. With regard to the second point, the picture is less clear. A 25% performance degradation at n=6 is a matter for serious concern even if the starting point is good; a 75% degradation at n=1 is probably unacceptable to many users. On the other hand, the modular approach taken by CloudFS means that users who do not require this level of protection need not pay the price, and the data also suggests that the price can be lowered significantly with little effort.

In addition to ongoing work on the features mentioned here and the implicit goal of making configuration/maintenance ever easier, work has already begun on several other features also of value in a cloud environment. Chief among these are in-flight encryption, easier addition/removal of servers, and asynchronous multi-site replication. This last feature is likely to be a major focus for CloudFS going forward, both to address disaster-recovery needs and to facilitate migration between clouds. Location transparency and cost efficiency are often cited as advantages of the cloud model, but are lost if moving computational resources from one cloud to another requires waiting for an entire large data set to be transferred as well.

References

[1] CloudFS: http://cloudfs.org/cloudfs-overview.

[2] GlusterFS: http://www.gluster.org/.

[3] PVFS: http://pvfs.org/documentation/index.php.

[4] Ceph: http://ceph.newdream.net/publications/.

HARDWARE

System Impacts of Storage Trends

Hard Errors and Testability

STEVEN R. HETZLER

Steven R. Hetzler is an IBM Fellow at IBM's Almaden Research Center, where he manages the Storage Architecture Research group. He has spent 25 years in data storage research and development. Steven is currently focusing on novel architectures for storage systems and on applications for non-volatile storage. His most notable patents include split-data field recording and the No-ID headerless sector format, which have been used by nearly all magnetic hard-disk-drive manufacturers for a number of years. Steven was educated at the California Institute of Technology, where he received his PhD and Master's degrees in Applied Physics in 1986 and 1982, respectively. He joined IBM Research in November 1985 and was named an IBM Fellow in 1998.
Hetzler@almaden.ibm.com

The continuing increase in disk storage capacity has been extremely beneficial to the IT industry. However, the growth in capacity has not been accompanied by increases in the hard error specifications. We will examine how this affects system design. SSDs aren't immune to the effects of hard errors either. In fact, in many ways, designing systems to properly accommodate SSD hard errors is more complex than for hard disk-based systems. In this paper we will explore hard errors, develop some improved methods for specifying them, examine testing requirements, and propose some new specifications for SSDs.

Background

Hard disk (HDD) capacity has been increasing exponentially since they were first introduced. In the past few years, however, the demand for enterprise storage appears to have exceeded the ability of enterprise disks to deliver storage that meets customer budgets. This can be seen in Figure 1: the proportion of capacity-optimized disks in enterprise applications is increasing. Commonly, the performance-optimized disk will be what the industry calls enterprise class (10,000 and 15,000 rpm) disk. Capacity-optimized disks will be 7,200 rpm SATA-class. Note that such disks may not use the SATA interface, but are distinguished by their slower rotation speed and higher capacity compared with enterprise class disks. SATA-class disks come predominantly in two grades: enterprise and consumer.

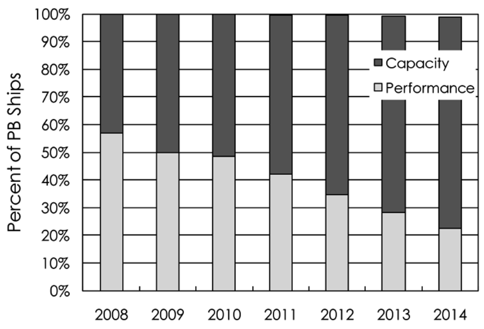


Figure 1: Percentage of PB shipped in enterprise storage systems by HDD class [1]

Unfortunately, there are side effects to the capacity increase in hard disks. The problem arises because not everything scales with capacity. This requires that system designs compensate for these effects. Let's start by examining the impacts that SATA-class disks will have on storage systems.

Hard Errors

Ideally, a storage device should always be able to return the data stored on it. When a storage device is unable to deliver a piece of data, it is referred to as a hard error. In such a case, the device has exhausted all means to recover the data. Thus, while the device is still operational, data has been lost.

Hard error specifications are one of the attributes that haven't scaled with capacity growth. However, the situation is obscured by the archaic units used to specify hard errors. Typically, storage devices are specified as less than one event per some number of bits transferred: e.g., <1 error in 10^{14} bits read [2]. We will refer to such a specification as a hard error interval, or HEI. While this may have been appropriate when disk interfaces were more primitive, it's not very informative today.

First, such a specification is not even statistically meaningful. The specification represents a minimum, which we all know isn't obtainable. The distribution is unspecified as well, so we can't tell if the mean is one error in twice the specified interval, or in 1000 times the interval. Thus, we can't model with the specification as is. The only reasonable approach is to operate as if it's a mean, and go from there. If the manufacturers are bothered by treating the specification as a mean, they could publish the distributions.

Second, a hard error results in an entire sector being lost, not one bit. Further, a block device can only transfer data in increments of a sector. Thus a per-bit specification is not as useful as a sector failure probability. Assuming a typical 512 byte sector size, we would have:

```
        Sector HE probability = 1/(HEI * 4096)
```

Third, the large exponents give an artificial impression that the devices are quite reliable. For example, 10^{14} bits seems incredibly large. However, there are 0.08×10^{14} bits in a terabyte! Thus, the reliability isn't quite what it seems.

A Better Hard Error Spec

I feel a specification should be sufficiently clear to allow one to determine its impact at a glance. Therefore, I propose that hard errors be specified as probability per TB transferred.

	SATA	**Ent. SATA**	**Ent. Hi Perf.**
Typical hard error interval (bits)[2], [3], [4]	10^{14}	10^{15}	10^{16}
Sector hard error rate	4×10^{-11}	4×10^{-12}	4×10^{-13}
Hard error rate (prob/TB)	8×10^{-2}	8×10^{-3}	8×10^{-4}

Table 1: Typical HDD hard error specifications. The SATA column refers to a consumer-grade SATA disk, the Ent. SATA column to an enterprise-grade SATA disk, and the Ent. Hi Perf. column to a high-performance enterprise disk.

This seems to be quite reasonable as a specification, and one might ask why the industry hasn't adopted it. I think the reason is clear—8% per TB doesn't sound very impressive. The 0.08% for high performance enterprise disk isn't terribly comforting, for that matter.

FURTHER ISSUES WITH CURRENT SPECS

Another serious issue for system designers is that the specifications as stated don't give any detail on the root causes of hard errors. As specified, it is related to disk read and write activity. Thus, we might assume that read and write operations create hard errors. This could be from head-disk contact, lube depletion, scratches, etc.

Alternatively, it might be just related to the drive operating, not tied directly to reading and writing. Such effects might include having a high bit error rate, defects in synchronization, defects in servo, process errors (off track positioning), etc.

In reality, all these likely contribute to differing extents, although manufacturers don't publicly release this information. From a system impact point of view, however, we typically need to use public information as a starting point.

Capacity-Related Specification

In storage systems, there are other metrics we can use to understand the impact of hard errors. I find it instructive to recast the hard error specification in terms of the drive capacity. Let's examine the probability of being able to read the entire drive capacity successfully—that is, without a hard error.

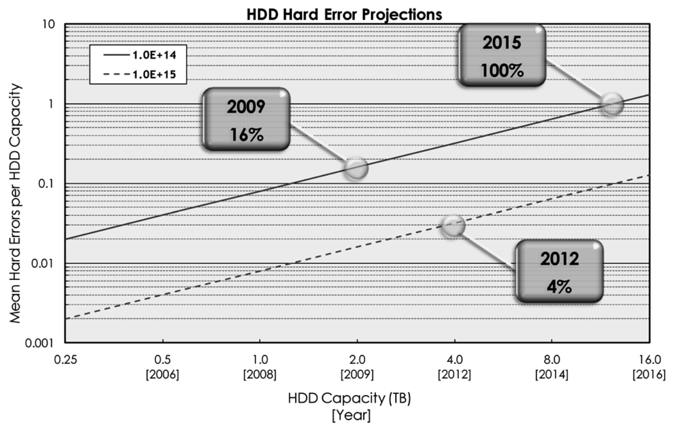


Figure 2: Plot of mean hard errors per HDD capacity vs. HDD capacity. The vertical axis is the log of hard error probability. The horizontal axis shows hard disk capacity in TB by the estimated year of first shipment. The solid line is for consumer-grade SATA disks, and the dashed line for enterprise-grade SATA disks.

In 2009, the largest SATA HDD capacity was 2 TB. At the specified hard error interval of $1x10^{14}$, the probability of encountering a hard error in transferring the entire capacity of the HDD would be 16%. At a capacity growth rate of 40% per year, one would therefore expect to encounter one hard error when transferring the capacity of the drive by 2015. It is interesting to ponder what this means. For enterprise-grade SATA disks, in 2012 the probability of a hard error when transferring the entire capacity of the drive would be 4%. As we shall see, this isn't dramatically different from the situation with consumer-grade SATA disks.

The impacts of such behavior depend on the application. Consider a typical consumer HDD used in a digital video recorder application. First, the result of a hard error might be the loss of a few frames of video. Further, the error rate of the input stream, such as from a satellite or cable, is higher—about 440/TB [5], and thus the contribution from the HDD hard error rate is negligible.

In a desktop PC application, it depends on the average data rate. If we assume that an average desktop HDD is used for 2,000 hours per year and that the time-average data rate for that period is 100 KB/s, this works out to 0.7 TB/year, or a 5% probability of one hard error per year. For reference, this example is the equivalent of 25 IOPS at a 20% duty cycle. In such a case, the impact of such hard error rates are likely to be seen soon, especially as HDD capacity and disk usage grow.

In storage systems, the situation is more acute, as the reliability depends on the ability to read the entire capacity of the disks.

A BRIEF REVIEW OF RAID-5 AND FAILURES

Because of its low cost and low performance overheads, RAID-5 remains a popular choice for protecting storage systems against disk failures. (We can safely ignore the effects of striping in our analysis here, as it doesn't affect the outcome.) RAID-5 uses a single parity disk to protect an array of disks. Since we have a full disk's worth of parity, a single failure will not cause data loss. There are three possibilities of dual failures we must consider in determining the probability of data loss. First, the array can lose a first disk, then lose a second disk before the missing data has been rebuilt onto a spare disk. This is commonly referred to as an *array kill event*. Second, the array can lose one disk, then encounter a hard error while rebuilding the missing data. This is commonly referred to as a *strip kill event*, where some portion of a strip's worth of data is lost. Third, the array can have two hard errors in the same parity strip. Having sector failures line up like this can be ignored to first order, since in hard disks sector failures do not exhibit a tendency to correlate between drives. The probability of two hard errors in a strip is order sector failure squared and the number of strips is also order sector failure (recall Figure 2); thus the resulting probability will be about 10^{-10}, which is too small to impact the failure rate.

QUICK RAID-5 DATA LOSS ESTIMATOR BY DISK CAPACITY (TB)

	1	2	4
Prob. disk loss/year	6.8%	6.8%	6.8%
Rebuild TB	7	14	28
Expected hard errors in rebuild	0.56	1.12	2.24
Prob. strip kill/year	3%	4.7%	6.1%
Prob. 2nd disk in rebuild	$1.6x10^{-4}$	$2.4x10^{-4}$	$3.9x10^{-4}$
Prob. array kill/year	$1.1x10^{-5}$	$1.7x10^{-5}$	$2.6x10^{-5}$
Prob.	3%	4.7%	6.1%

Table 2: Estimated failure probability per year for a 7+P RAID-5, assuming disks have a $1x10^{14}$ hard error specification and 1 MH MTBF. Each column shows the results for arrays built with disk drives of the indicated capacities.

We can easily create a simple RAID-5 loss estimator to see the impact of hard errors. Table 2 shows the results of the estimator for a system with seven data disks and one parity disk, where the disks have a stated reliability of 1 MH MTBF (one million hours mean time between failures). The latter equates to an annual failure rate (AFR) of 0.9% (suddenly a 1 MH MTBF doesn't seem that impressive). Using the AFR value, a binomial can be used to compute the probability of a first disk loss during a year (shown on line 2). The amount of data needing to be read to complete a rebuild after a data loss is seven here (the number of data disks). Given what we have previously shown regarding the hard error rate per TB, it should be clear what the result is going to be. Line 3 shows the expected number of hard errors encountered during the rebuild. Note that in the case of drive capacity > 2 TB, the expected number of hard errors is greater than one. Thus, we expect rebuilds to fail more often than they succeed! Line 4 shows the probability of strip kill per year, which is greater than 1% for all cases shown here. Line 5 shows the probability of a second disk failing during the rebuild process, which here is assumed to take about eight hours. The array kill probability is 100x smaller than the strip kill probability, and thus we need only consider strip kill here for the final probability of data loss. Switching to enterprise-grade SATA disks only reduces the strip kill rate by a factor of 10. So a system built in a year or so with 4 TB enterprise-grade SATA disks would still have >0.5% chance of failure per year, which is not very comforting.

It is interesting to note that we have created an array with a greater failure rate than the disks we built it out of (3% vs. 0.9%)! This is not quite a fair comparison, but it does say that such a design is of dubious reliability.

So it appears that RAID-5 is unsuitable for use with consumer SATA disks. However, as HDD capacities increase, if the hard error specifications remain constant (as they have for 10 years) [6], then enterprise-grade SATA disks and even high-performance enterprise disks will eventually produce the same result. Thus, the movement to stronger protection, such as dual-parity RAID-6, is inevitable.

SCRUBBING

Scrubbing is a technique where the data on the array is checked for errors (scrubbed) while all disks are functional. Any hard errors encountered can thus be corrected. While scrubbing should be able to limit the accumulation of hard errors, one has to wonder how valuable it is at large capacities. Consider a RAID-6 as eight data plus two parities, using 2 TB consumer-grade disks. The scrub reads 20 TB with an 8% per TB chance of a hard error, so we expect the scrub to leave us with about one new hard error in the array! An interesting question is how scrubbing should be implemented, when it will always leave hard errors behind. I liken it to washing a car during rain. The accumulated dirt can be removed, but the car will never be clean.

Importance

The enterprise storage market is increasing its use of lower-cost disks (such as enterprise-grade SATA). However, the data reliability requirements need to remain the same. Thus, action must be taken at the system level to understand and plan for the impacts. Clearly, stronger protection than RAID-5 is warranted for SATA disks of either grade. However, moving to RAID-6 will increase system costs and reduce performance. Scrubbing policies might also require adjustment.

Given the trend lines shown in Figure 2, it would seem beneficial for the HDD hard error rate behavior to be improved.

Hard Errors with SSD

It is tempting to believe that hard errors are a problem associated with magnetic recording and that solid state storage technologies will thus suffer to a far lesser extent. However, this isn't necessarily the case, and the situation warrants examination. For the following discussion, we will consider 2-bit MLC NAND flash as the underlying solid state storage.

NAND flash behaves quite differently from an HDD in response to storage operations (e.g., reading and writing). In a disk drive, the hard error rate has not been shown to significantly depend on how many times a data location is written, or on how old the data is. Experience has shown that the hard error rate may be treated as a constant for a device. The situation with NAND flash is quite different. NAND flash has both a finite endurance (that is, a limit on the number of times a location can be written), and a finite retention. Further, these parameters are coupled—the more a location is written (also called cycled), the shorter the data retention span. Therefore, the bit error rate may be expressed as a surface in three dimensions, as illustrated in Figure 3. The shape must be similar to what's shown. Since it is possible to cycle the device to complete failure even at short cycle times, the error rate multiplier must increase accordingly. Similarly, the data degrades over time; the error rate multiplier must increase with age. Since we expect the surface to be smooth, it will have a shape like that in Figure 3.

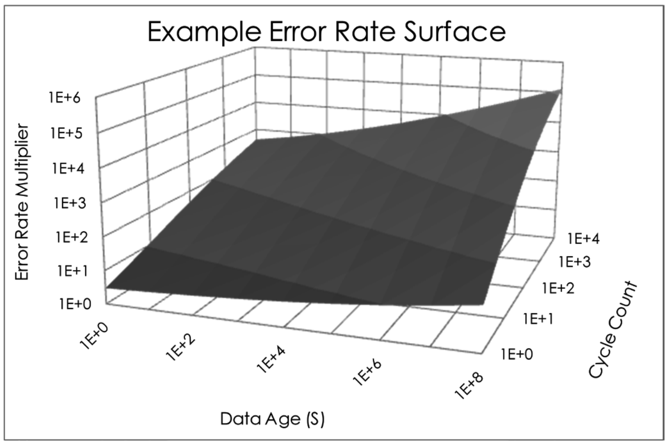


Figure 3: Example NAND flash error rate surface. The horizontal axis is data age in S, from 1 S to 3 years, the depth axis is the cycle count, and the vertical axis is the bit error rate multiplier, relative to the bit error rate at 1 cycle and 1 ms data age [7].

The behavior of flash is more complicated than described above. The bit error rate also depends on temperature and on the number of times a block has been read (read, disturbed). Thus, we have a five-dimensional surface.

System Reliability with SSDs

Reliability Targets

It is important to understand how to create a reliability target when designing a storage system. The targets should be expressed in a manner that reflects usage. Therefore, I propose that the storage system targets be specified as a target probability of a failure event per unit time. I choose this approach since it reflects how the user experiences reliability. Other methods of expressing reliability, such as per byte or per I/O, don't reflect user experience and are difficult for the user to measure (as we have seen for hard errors). Therefore the target should be expressed in a manner that clearly shows whether the system meets the target.

The design of targets is often based on the desired single user experience, such as single customer install. I prefer to use program-based targets, which cover the entire field population for the life of the program. This is how the business team of a storage system manufacturer will determine whether the program will meet its financial targets.

Inputs to a program-based target will include the install base and an estimate of the field failure the program can tolerate. The latter will depend on the type of failure. For example, does the failure result in a warranty event, a loss of access, and data loss, or will it significantly impact the customer's business?

A Simple Program-Based Reliability Target Estimator

Program-level targets are coarse enough not to require high precision. We only need precision to an order of magnitude here. Thus, for the simple estimator we will ignore effects such as duty cycle, read/write ratio, etc.

Table 3 shows an example for an enterprise storage system program using SSDs. Assume we are using enterprise-grade SSDs capable of 30,000 IOPS and that the desired field lifetime for the program is five years. Assume we plan to ship 50,000 SSD units each year. This gives a field population of 250,000 SSDs for the full program. Hard errors are sector-based events for most SSDs, and a typical 4 KB I/O is eight sectors. We can therefore compute the total program sector operations, arriving at $8x10^{18}$ sector operations for the program. Note how large this value is!

SSD unit IOPS	30,000	Enterprise SSD
Field lifetime	5	Years per product
Field SSD units	250,000	For full program
I/O size	4	KB
Total program sector ops	$8x10^{18}$	Program life usage
Field events/program	(0.1–50)	Depends on the event

Table 3: SSD-based program reliability estimator.

As mentioned earlier, the target number of field events depends on the event. If the event causes data loss, a value as high as 50 would equate to about one field data loss event per month. A value of 500 would be one a week, which is clearly too high for an enterprise program. Thus, 50 is an upper bound on acceptability. If the event causes a significant customer disruption, such as a data corruption event, then the target might even be less than 1. A business team might claim that the proper target is 0, but this wouldn't be practical. So a 10% chance of one such event in a program might be reasonable here.

Another important metric can be derived from this information—the number of unit-years of SSD operations. This program represents 1.5 M unit-years of SDD operation.

TEST CAPABILITY

Now that we have a field reliability target, we need to determine how to develop a test program to confirm that the system meets the target. HDD-based systems provide us with decades of experience to draw upon. A typical qualification test for an HDD program is to use 1,000 drives for 1,000 hours. Let's examine an SSD test based on these parameters. Assume that the device can operate at 80% duty cycle during the test, as we need to perform tests on the data. This leaves us with 24,000 IOPS per SSD. Thus, for the full 1,000 SSDs and 1,000 hours, we can test $7x10^{14}$ sector operations. However, this is $9x10^{-5}$ of the program sector operations. Thus, it will be very difficult to extrapolate the program behavior based on the results of such a test. (We haven't examined the confidence levels given 0 or even a few events seen during the test.) The bottom line is that this is only a 100 unit-year test, and we don't have much historical experience to draw upon.

SSD Specifications (Speed Kills)

One approach used to increase the confidence in such a test is to utilize an acceleration factor, such as temperature. There isn't room to examine the efficacy of such an approach in this paper (I have strong reason to suspect that such acceleration is questionable [8]), but we should examine SSD hard error specifications first.

SSDs are capable of significantly higher random I/O rates than hard disks. Given that SSDs are significantly more expensive than hard disks of a given capacity, it is reasonable to assume that they will be used only where the higher performance is required. Thus, we should consider how they behave in high random I/O workload.

The hard error specifications for SSDs are typically just copied from hard disk specification in the same market segment [9]. However, an SSD will be operated at substantially higher I/O rates than an HDD. Thus the hard error specification needs to be scaled accordingly. Let's examine this premise.

	Ent. SATA HDD	**Ent. Perf. HDD**	**SSD consumer**	**SSD enterprise**
IOPS	120	250	10,000	30,000
Equivalent MB/s	0.48	1	40	120
Hard error bit interval spec.	$1x10^{15}$	$1x10^{16}$	$1x10^{15}$	$1x10^{16}$
Mean years/hard errors	66	320	0.8	2.6
Proposed scaled bit interval spec.			**$1x10^{17}$**	**$1x10^{18}$**

Table 4. SSD and HDD hard error specifications and their impacts.

Table 4 is a comparison of enterprise-grade SATA disk, high performance enterprise disk, consumer-grade SSD, and enterprise-grade SSD. The IOPS row lists the sustained random I/O per second each device might attain. Clearly, the SSDs exhibit superior performance characteristics. Row 2 is a conversion of the IOPS rate to MB/s, assuming a typical 4KB I/O size. The hard error specifications are shown in row 3. The consumer SSD specification is the same as the enterprise SATA HDD, and enterprise SSD and high performance HDD also share the same specification. Given the values in rows 2 and 3, we can easily compute the mean years/hard error, as shown in row 4. It should be obvious given the IOPS ratios between devices, but the SSDs operating at these specifications will be expected to exhibit hard errors substantially more often than HDDs. The consumer SSD would not be expected to go a full year between hard errors.

It would appear that the SSD specifications weren't derived using the above analysis. Since the SSDs are about 100 times faster than the HDDs in the same market segment, they will need a hard error specification 100 times better just to maintain the same reliability as HDDs, as shown in the last row. Claiming SSDs are more reliable than HDDs would require an even higher specification than I have proposed.

Going back to the program example of Table 3, there are roughly 10^{19} sector operations in the program. Table 1 shows that the sector hard error probability for the enterprise SSD specification is $4x10^{-13}$. Thus we might expect something on the order of 10^{6} sector errors during the program. Such behavior is normally of little concern, as RAID can be used to correct operational hard errors. However, the situation with SSDs is more complex than with HDDs. RAID reliability computations like those illustrated in Table 2 make two key assumptions regarding failures: failures are assumed to be both independent of each other and independent of time. Unfortunately, neither is true with SSDs. Since flash has a wear-out mechanism where the bit error rate (thus the hard error rate) is a function of the number of times a location has been written, failures are time-dependent. Further, since RAID systems spread the data to evenly distribute the write accesses (identical in the case of a mirror), we expect the failures to correlate across SSDs in an array. Thus both assumptions are false, and we can't assume that RAID will be sufficient. We need to create a more sophisticated RAID model to compute reliability and a more sophisticated RAID design to accommodate SSDs.

If SSDs met my proposed scaled hard error specifications, then the strip kill exposure problem would likely be 100 times smaller. However, proving that SSDs meet such a specification will be problematic. We determined above that a 1,000 SSD 1,000-hour test could perform about $1x10^{15}$ sector operations. My proposed enterprise SSD hard error specification is a sector failure probability of $3x10^{-15}$; thus, the counting statistics in a 10^{15} operation test will be limited. (The nature of the error rate surface will necessitate even further testing to get endurance, retention, temperature, and read-disturb effects.)

SSD Testing

Since a flash cell has a finite (and small) write endurance, SSDs use wear leveling to spread the write load more evenly across the device. This is widely assumed to increase the reliability, as it substantially increases the amount of data that can be written to the device prior to wear out. However, there are side effects to wear leveling that negatively impact the reliability. One of the most significant is that it restricts the ability of the system integrator to test the device.

Wear leveling is essentially a virtual address layer that decouples the external logical block address (LBA) from the internal physical block address. Such devices necessarily perform background operations, such as reclamation, and so an LBA may be relocated without providing external information. This means that an external system can't know the physical location of a given LBA, its write count, or the data age. Therefore it isn't possible to construct a deterministic test using a wear-leveled device. The best one can do is to write the entire device sequentially until wear-out. This is time-prohibitive and still leaves many unknowns.

It is interesting to note that system integrators, who are very reluctant to employ hard disks that they can't test, don't seem to have similar reservations about SSDs. Essentially, they are forced to trust the vendors.

IN FLASH, ERRORS NEVER SLEEP . . .

Another side effect of wear leveling is that it enhances the cluster failure of sectors, since it has a tight distribution of write cycle counts. Coupled with the growth of the bit error rate as the data ages, wear leveling also makes the transition from a working device to one which is significantly out of specification much more rapid. This can be seen in Figure 4, which is from actual measurements of a commercial MLC SSD using 5,000 cycle rated flash. We can see that data of age 1000 hours (solid curve) meets the $4x10^{12}$ sector loss specification at about 3770 cycles. Now, if this data is allowed to age just 48 more hours, the hard error rate doubles! Looked at another way, the cycle specification at 1048 hours is 105 cycles shorter than it is at 1000 hours. Thus, we can see that having a large fraction of the sectors at similar cycle counts in this range can put a substantial fraction at risk in a very short time.

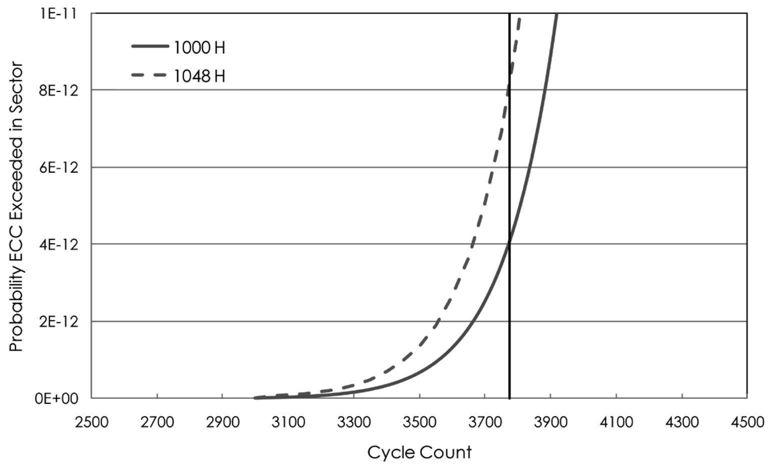


Figure 4: Measured probability of sector hard error vs. cycle count at different data ages. The vertical axis is the probability the ECC is exceeded for a sector (a hard error). The x axis is the write cycle for the sectors. The solid curve is for a data age of 1000 hours, and the dashed curve is for a data age of 1048 hours.

Conclusion

Disk capacities have been growing exponentially for decades, continuing to feed the market's insatiable demand for storage. However, the hard error specifications haven't kept pace. This has led to a situation where it would no longer be advisable to use RAID-5 with enterprise-grade SATA disks. If the hard error rates aren't improved, it will be only a few more years before high performance enterprise disks hit the same limit. While stronger RAID levels, such as dual parity RAID-6 can help, much of the additional protection will go towards hard error protection, as opposed to dual disk failures. It is also likely that scrubbing policies will require modification, as we will reach the point where scrubbing is unlikely to leave disks without hard errors.

It is widely assumed that SSDs are more reliable than hard disks, since they have no moving parts. As we have seen, this assumption isn't necessarily true and can't be verified with the data available. In fact, using SSDs at the JEDEC hard error specifications [9] could actually increase the time rate of failure in enterprise systems. This is because SSDs are deployed in higher-performance applications, but the specifications have been copied from hard disks which don't perform at these levels. I propose that SSDs require more stringent hard error specifications appropriate to their workloads.

There is a significant dearth of published information on the reliability of SSDs. One reason may be that the wear leveling used to increase device lifetime in SSDs has the side effect of preventing users from testing the devices. Properly designing a system using SSDs requires a deep understanding of the error rates. Thus, it would be beneficial for the industry to develop testable SSDs.

References

[1] N. Yezhkova and R. Villars, "Worldwide Enterprise Storage Systems 2010-2014 Forecast Update," IDC #226223, December 2010, p. 27.

[2] Caviar Black specification sheet, Western Digital 2879-701276-A11, March 2011, p. 2: http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701276.pdf.

[3] RE4 specification sheet, Western Digital 2879-701338-A05, September 2010, p. 2: http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701338.pdf.

[4] Ultrastar 15K600 data sheet, Hitachi Global Storage Technologies DSU-S156009EN-03, October 2009, p. 2: http://www.hitachigst.com/tech/techlib.nsf/techdocs/DAC6606EE8DDF5D7862576490028557B/$file/US15K600_DS_final.pdf.

[5] For example, the target error rate for satellite DVB-S2 is 1x10-7 packet error rate, or about 1 in 18 Gb; see Alberto Morello and Vittoria Mignone, "DVB-S2: The Second Generation Standard for Satellite Broad-band Services," *Proceedings of the IEEE*, vol. 94, no. 1, January 2006, p. 214.

[6] For example, DiamondMax D540X data sheet, Maxtor Corp. 9/01 6158v2 update, April 2002. This was a desktop drive with capacities up to 160 GB.

[7] Steven R. Hetzler, "System Impacts of HDD and Flash Reliability," IDEMA Hard Disk and Solid State Drive Reliability Symposium, May 2008.

[8] Steven R. Hetzler, "System Design Impact of Storage Technology Trends," FAST '11: 9th USENIX Conference on File and Storage Technologies, T4, February 2011.

[9] JEDEC Standard 218, "Solid-State Drive (SSD) Requirements and Endurance Test Method," September 2010, p. 7.

SYSADMIN

Testing the Transport Side of Highly Available Hosts

STUART KENDRICK

Stuart Kendrick works as a third-tier tech at the Fred Hutchinson Cancer Research Center in Seattle, where he dabbles in trouble-shooting, deep infrastructure design, and developing tools to monitor and manage devices. He started earning money as a geek in 1984, writing in FORTRAN on CRAY-1s for Science Applications International Corporation; worked in desktop support, server support, and network support at Cornell University; and reached FHCRC in 1993. He has a BA in English, contributes to BRIITE (http://www.briite.org), and spends free time on yoga and CrossFit.
skendric@fhcrc.org

When I configure Highly Available (HA) systems, I intend to configure them such that if either half goes down, the service remains accessible to the user. Too often, however, I find that I have configured them to be Highly Unavailable (HU), meaning that if either half goes down, the service becomes inaccessible.

Highly Available hosts typically sport redundant Ethernet NICs. Originally, I would test these by unplugging the cable feeding each NIC, perhaps while sending a continuous ping to the host's IP address to measure how well it handled the event. As it turns out, this test covers a limited selection of possible failure scenarios, not enough to determine whether the host has been configured HA or HU. In this article I describe a test protocol which more accurately assesses the host's configuration.

HA Transport in Datacenters

There are many ways to design a datacenter [1] to deliver Highly Available Ethernet/IP [2]. While the details vary widely—and have repercussions for host configuration and the parameters described here—they all share the same concepts: elements probe one another periodically to verify that the current network path is viable and, if it isn't, switch to an alternate path. For the purposes of discussion, Figure 1 illustrates one such design. Contrast it with what I will call a Single Point of Failure (SPOF) design, in which Switch B and Router B do not exist.

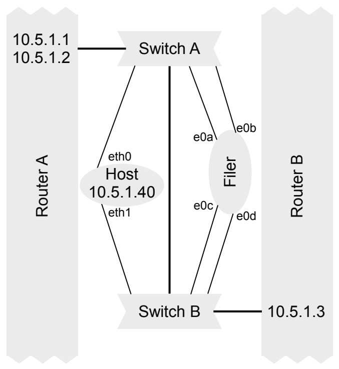


Figure 1: HA Ethernet/IP datacenter design

The Host is equipped with two NICs, one plugged into Switch A, the other into Switch B. Filer is a NetApp storage controller; we'll talk about it later. The two Switches are interconnected and also provide paths to the upstream routers (likely marketed as Layer 3 switches by their vendors, but for the purposes of this discussion, I'll use the term *router* to indicate a Layer 3 device). The Host is configured to rely on 10.5.1.1 as its default gateway. 10.5.1.1 is configured as a virtual IP address, currently owned by Router A. If Router A goes down, then Router B will acquire 10.5.1.1, thus hiding the loss of the default gateway from Host. We accomplish this magic via the use of a *gateway redundancy protocol*, e.g., VRRP, CARP, HSRP, or GLBP. Here is an example of HSRP in action on the wire; each router emits these Hellos once per second.

```
Delta T   Src          Dst        Protocol
0.55430   10.5.1.2 -> 224.0.0.2  HSRP Hello (state Active)
0.07153   10.5.1.3 -> 224.0.0.2  HSRP Hello (state Standby)
```

If the standby router (Router B) does not hear from the active router (Router A) for a configured amount of time (three seconds in our environment), then the standby router will change its state to Active, adopt both the MAC address and the IP address of the default gateway (10.5.1.1), and thus set up shop as the "go to" router on this subnet.

The Host can be configured to employ both NICs simultaneously (Active/Active) or to rely on one NIC while holding the other in reserve (Active/Standby). In both cases, though, it faces the same problem that Router A and Router B face: how does it know when its partner—in this case, one of its NICs—is defunct?

Sources of Failure

By default, the average host out of the box pays attention to the Ethernet link signal transmitted by the switch in order to determine whether it should consider a NIC viable. Regrettably, a range of hardware failures and human fat-fingering can result in the switch continuing to transmit link but no longer forwarding frames. For example, when the Supervisor/switching engine (the brains card) in a switch fries, the individual ports continue to transmit link, but the brains no longer forward frames arriving from hosts. Rebooting a switch, perhaps to load a new operating system, results in similar behavior during the boot process—as the switch reboots, it performs hardware tests on its cards, toggling link up and down several times, all the while tossing incoming frames into the bit bucket. Similarly, a line card can lose its connection to the switch's backplane, or a human can fat-finger a VLAN assignment, isolating a host from its intended conversation partners. *The lights are on, but no one is home.*

Figure 2 enumerates the failure modes we have experienced at our institution. Notice how, from a purely component-based point of view, Highly Available environments will tend to experience twice as many component failures as their non-redundant equivalents, because they contain twice as many parts. When compared to a SPOF design, HA environments also mean that:

- Operating staff spend twice as much time replacing fried components.
- When systems are configured in a Highly Unavailable way, there's twice as much downtime.

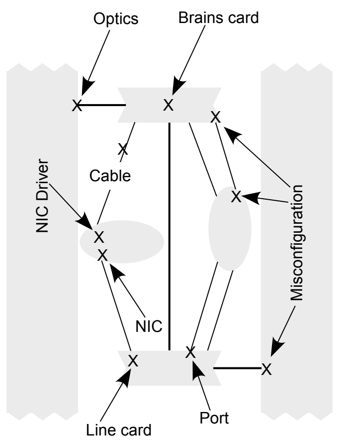


Figure 2: Sources of failure

Ethernet and IP both being *ship and pray* protocols [3], the transport infrastructure cannot recover from such errors: the burden for detecting and responding to such issues lies with the host. Hosts which rely strictly on link for determining the validity of a NIC will transmit frames into the bit bucket ad infinitum during one of these failure scenarios.

Polling

Host operating system designers and NIC driver developers commonly provide mechanisms for Ethernet NICs to exchange keep-alives with each other, with the upstream default router, or with the brains card on the nearby Ethernet switch. In this fashion, the host OS determines whether or not a given NIC has a viable path to the rest of the world (network) and then decides whether to continue forwarding frames across that NIC. However, operating systems do not ship with these features enabled; we system administrators must identify which configuration fits our environment and turn it on.

Linux

The Linux folks wrap their Ethernet HA options into their *bonding* driver [4]. In this example, I poll the IP addresses of the local routers to determine the viability of a NIC and of its path through the local network.

```
Host> cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
IPADDR=10.5.1.40
NETMASK=255.255.255.0
NETWORK=10.5.1.0
BROADCAST=10.5.1.255
GATEWAY=10.5.1.1
[…]
BONDING_OPTS='mode=active-backup arp_interval=1000 arp_validate=all \
arp_ip_target=10.5.1.2 arp_ip_target=10.5.1.3 primary=eth0'

Host> cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
```

```
[…]
MASTER=bond0
SLAVE=yes

Host> cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth0
[…]
MASTER=bond0
SLAVE=yes
```

Be aware that the `ifcfg-xxxx` format was different in older versions of the bonding driver; the format illustrated here became accurate with v3.2.0 or thereabouts. Type `cat /sys/module/bonding/version` to view your version and see bonding.txt for syntax variants appropriate for your version.

Restart networking via `/etc/init.d/network restart` or similar and verify that the bonding driver correctly reports key parameters; see bolded text below.

```
Host> cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.2.4 (January 28, 2008)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: eth0
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0
ARP Polling Interval (ms): 1000
ARP IP target/s (n.n.n.n form): 10.5.1.2, 10.5.1.3

Slave Interface: eth0
MII Status: up
Link Failure Count: 2036
Permanent HW addr: 00:19:b9:32:76:25

Slave Interface: eth1
MII Status: up
Link Failure Count: 1
Permanent HW addr: 00:19:b9:32:76:27
```

On the wire, the result is the following, exchanged every 1000 ms:

```
Delta T   Src               Dst            Protocol
0.50123   Dell_32:76:25  -> Broadcast      ARP Who has 10.5.1.2?  Tell 10.5.1.40
0.50124   Dell_32:76:25  -> Broadcast      ARP Who has 10.5.1.3?  Tell 10.5.1.40
0.50158   Cisco_64:3a:de -> Dell_32:76:25  ARP 10.5.1.2 is at 00:03:6c:64:3a:de
0.50162   Cisco_43:bc:00 -> Dell_32:76:25  ARP 10.5.1.3 is at 00:03:6c:43:bc:00
```

In this *active-backup* configuration, the *active slave* emits ARP Requests every 1000 ms. If it doesn't hear an ARP Reply from either 10.5.1.2 or 10.5.1.3 within 2000 ms (2 * arp_interval: see drivers/net/bonding/bond_main.c in the Linux kernel source), then the driver marks the *active slave* (eth0 in this case) as down and initiates a failover to the *backup slave* (eth1) [5].

In this design, failure to hear from either 10.5.1.2 or 10.5.1.3 covers all the failure scenarios illustrated above.

Windows

Under Windows, NIC manufacturers provide driver features which implement similar polling. Intel calls their approach *Teaming* [6]; the admin uses a GUI [7] to configure the parameters [8]. Let's talk about the choices.

I prefer *adapter fault tolerance* (e.g., Active/Standby) over active load-balancing (e.g., Active/Active), because I find packet capture easier. With Active/Active configurations, I need two sniffers, plus the headache of merging the two trace files together (rarely a precise process)—that is a lot of overhead, particularly when I am in a hurry to fix something that is broken. Furthermore, if the host actually needs to employ both NICs in order to deliver sufficient service (needs to transmit and/or receive across both NICs), then it is no longer Highly Available—the loss of either NIC will lead to service degradation.

I set *Activation Delay* to 100, which instructs the driver to leave a NIC disabled for 100 seconds after it has determined that the NIC is ready to return to production, this because cable and switch failures can be erratic, working for a few seconds, failing for a few seconds. By instructing the driver to wait a while before re-enabling a previously disabled NIC, I harden the host against this sort of flapping experience.

And of course I enable *Probes*.

Here is what Intel Teaming looks like on the wire, with the Active and Standby NICs each sending probes to one another.

```
Delta T   Src               Dst         Protocol
0.51795   Intel_e4:ea:72 -> Multicast   Intel ANS probe Sequence: 3098765056,
                                        Sender ID 256
                                        Team ID 00:11:43:e4:ea:72
0.51796   Intel_e4:ea:73 -> Multicast   Intel ANS probe Sequence: 3098765056,
                                        Sender ID 512
                                        Team ID 00:11:43:e4:ea:72
```

Broadcom calls their approach LiveLink, which uses the same ARP polling approach that Linux *bonding* uses, although LiveLink requires that each NIC have its own IP address, in addition to the shared virtual address. For details, poke around Dell's site [9] or consult yours truly [10]. I recommend updating to the latest drivers in order to dodge a series of nasty bugs—we're using v4.1.4.0 successfully.

NetApp

The ONTAP designers chose to layer their Ethernet High Availability scheme on top of IEEE 802.3ad, aka Link Aggregation Channel Protocol (LACP). Linux bonding, Intel Teaming, and Broadcom LiveLink all support this approach as well; ONTAP requires it. LACP was intended as a protocol to permit bundling multiple Ethernet links into a single pipe or channel in order to increase the throughput available between two switches or between a host and a switch. However, as the ONTAP designers realized, LACP ships with a built-in polling protocol—the host and the switch exchange periodic Hellos to ensure that their understanding of the channel specifications are synced.

```
Delta T  Src                 Dst            Protocol
1.05400  NetApp_00:45:44 -> Slow-Protocols  LACP Actor Port = 1 Partner Port = 368
1.07000  Cisco_06:b4:7f  -> Slow-Protocols  LACP Actor Port = 368 Partner Port = 1
```

The brains cards in Ethernet switches are responsible for emitting these Hellos. Thus, the host configured for LACP can determine whether or not anyone is home in the switch by listening for silence.

In ONTAP-speak, I create dynamic, multimode Virtual Interfaces (VIFs) [11] using the LACP protocol and then combine pairs into single-mode VIFs, where e0a and e0c are plugged into Switch A and e0b and e0d are plugged into Switch B, per Figure 1 [12].

```
Filer> rdfile /etc/rc
hostname Filer
vif create lacp dmmvif1 -b ip e0a
vif create lacp dmmvif2 -b ip e0b
vif create lacp dmmvif3 -b ip e0c
vif create lacp dmmvif4 -b ip e0d
vif create single svif1 dmmvif1 dmmvif2
vif create single svif2 dmmvif3 dmmvif4
```

Myself, I don't like entangling host and switch configurations—adds an additional dependency and yet another way for the switch admin to break the host. Furthermore, while the host can detect both cable failure and switch failure using this scheme, it cannot detect either a switch admin fat-fingering a VLAN assignment or the loss of the path between Switch A and Router A. Finally, LACP misconfigurations and bugs are hard to troubleshoot, because capturing the LACP traffic requires an in-line sniffer (LACP is a link-local protocol, invisible to Wireshark running on the host or port mirroring on the switch).

On the other hand, the LACP approach dodges the polling and configuration subtleties inherent in the competing techniques, relying as it does on the protocol's built-in Hello function. And for the ONTAP developers and testers, it eliminates an entire chunk of functionality (an ARP-based polling mechanism, for example) which they would otherwise have to implement and maintain—all steps in the right direction, as far as uptime goes. In the end, we like our NetApps for a range of reasons, and this is the only NIC HA approach ONTAP supports, so we do it.

Application-Layer Protocols

As an aside, if our application-layer protocol contains its own polling techniques, then we can dispense with all these kernel-level and driver-level shenanigans. For example, SCSI Initiators and Targets exchange frames, if only NOPs, every five seconds. When configured with *multipathing*, SCSI running over IP (iSCSI) and SCSI running over Fibre Channel (FC) have no need for these lower-layer fault detection protocols—SCSI itself detects the failure of a path and initiates failover to a backup path—in our experience, a robust technique.

Here we see the iSCSI Initiator (10.5.1.50) emitting NOPs to its two iSCSI Targets (10.5.1.61 and 10.5.1.62).

```
Delta T    Src           Dst          Protocol
0.04404    10.5.1.50 ->  10.5.1.61    iSCSI NOP Out
0.04450    10.5.1.61 ->  10.5.1.50    iSCSI NOP In
0.05009    10.5.1.50 ->  10.5.1.62    iSCSI NOP Out
0.05043    10.5.1.62 ->  10.5.1.50    iSCSI NOP In
5.04423    10.5.1.50 ->  10.5.1.61    iSCSI NOP Out
5.04451    10.5.1.61 ->  10.5.1.50    iSCSI NOP In
5.05118    10.5.1.50 ->  10.5.1.62    iSCSI NOP Out
5.05230    10.5.1.62 ->  10.5.1.50    iSCSI NOP In
```

Test Procedure

So how do we verify that all this stuff actually works? I have experimented with pulling cables, assigning ports to the wrong VLANs, and even inserting mini-switches between host and switch (in order to sustain link but still produce a bit bucket—I yank the cable marked with an 'X' in Figure 3).

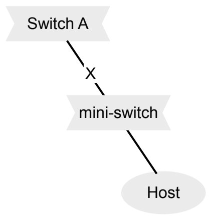


Figure 3: Manual testing

Each of these tests provides varying levels of validation, with the mini-switch test being particularly effective. However, the test I prize above all is rebooting each of the switches and routers in turn, because a reboot exercises a whole range of failure scenarios, from loss/restoration of link to the more brutal *the lights are on, but no one is home* condition, incurred while the switch ports are transmitting the link signal but the brains card is still performing hardware checks.

I like to measure the behavior of hosts using mass-ping [13], possibly because it is a brilliant and precisely honed tool for tracking IP connectivity or possibly just because I wrote it. mass-ping emits a stream of ICMP Echos, one per second, to a list of IP addresses or to entire subnet ranges, giving the operator real-time feedback on behavior and saving the test result to a CSV file, which a supporting tool can convert into a graphic.

```
Host> sudo mass-ping -s yes -c "Reboot Switch A" -n switch-a -w 900 -q 10.5.1.0/24
[sudo] password for skendric:
Sanity check...
Identifying live hosts...

Beginning with 144 live addresses
Pinging targets every 1 seconds with timeout 0.2 seconds, running for 15 minutes,
hit Ctrl-C to cancel...
144 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144
3   3   3   130   130 130 130 130 130 130 130 130 130 130 130 130 130 130 130 130
130 130 130 [...] 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144 144
144 144 144 144 144 144 144 144 144 144 144 144
```

The report at the end summarizes results:

```
# target        hits    misses
# -------------- -----   ------
aurora          897      3
jennite         897      3
madison         417      483
osiris          415      485
[…]
```

The graph-mass-ping script takes the CSV data file as input and produces a graphic illustrating when each host missed pings. The exclamation point represents ICMP Replies while the periods represent silence.

```
Title           Mass-Ping:  Reboot Switch A
Invocation      mass-ping -i 1 -w 900 -t 0.2 -q 10.5.1.0/24
Details         Run from Host on 2010-01-30 at 05:49 by skendric
Errors
Node count      144
Time count      900

Nodes                           Time
             05:49:05  05:49:15  05:49:20  05:49:25  05:49:30
aurora       !!!!!!!!!!!!!!!!!!!!...!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
jennite      !!!!!!!!!!!!!!!!!!!!...!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
madison      !!!!!!!!!!!!!!!!!!!!...........................................
osiris       !!!!!!!!!!!!!!!!!!!!...........................................
[…]
```

In this example, the datacenter contained 144 active IP addresses, 130 of which didn't blink when Switch A went down. Of those that noticed, aurora and jennite each needed three seconds to flip to their standby NICs, while madison and osiris stumbled and didn't recover.

Ideally, I run multiple simultaneous mass-ping invocations, from hosts located within the datacenter (one on each VLAN) and from at least one host located outside the datacenter, while performing each test. Large-format color printers (tabloid sized paper), tape, and a lot of wall space allows me to view mass-ping output for an entire datacenter. Periods during which all hosts miss pings suggest a systemic issue—unplugged switch-to-router uplink cable or misconfigured routing protocol. Periods during which many hosts miss pings but many do not suggest a misconfigured VLAN. And periods during which a few hosts miss pings but most do not suggest host-specific issues. In our environment, the most common causes of systemic failure have been fried optics and misconfigured router interfaces, while the most common causes of host failure have been bad NICs, unplugged cables, and buggy NIC drivers.

Some hosts throttle the rate at which they will respond to pings—get enough mass-ping sessions going and you can bump into that rate-limiter. For example, under ONTAP, check your settings for option `ip.ping_throttle.drop_level`.

Rubber Hits the Road

We started implementing redundant Ethernet/IP transport in 1998, intermittently testing it using manual techniques. In 2004 I automated the test process, writing code [14] which steps through the 15 or so redundant pairs of routers and switches at our institution, rebooting each one in turn, and watching, via pings, hosts on the other side. It runs via cron once per month. When it detects trouble, it halts and pages me. Each year, this automated process uncovers a handful of flaws in the switched/routed infrastructure plus numerous host-specific issues [15].

The most spectacular issue we have uncovered to date revolved around our NetApps. We had been skipping the automated testing of the larger datacenters, based on conflicts with projects and general anxiety. In fact, the previous test had occurred just prior to turning up our new centralized VMware cluster—seven Sun 4450s mounting a clustered FAS3020 back-end via NFS. Turns out that we had misinterpreted NetApp documentation, believing that ONTAP implemented an ARP-based probing scheme, similar to Linux bonding, and that this came for free, without specific configuration on our part. *Note to self: There is no free lunch.* We rebooted the first Ethernet switch to load the new OS, and VMs started crashing. More precisely, Windows crashes when it cannot write to disk; Linux enters read-only mode.

```
Apr 24 05:45:02 cairo kernel: Remounting filesystem read-only
Apr 24 05:45:02 cairo syslog-ng[2312]: io.c: do_write: write() failed (errno 30),
Read-only file system
```

In theory, detecting the frozen Windows guests was easy—our network management station reported them down. In practice, we had been forgetting to add VMs to the management station's lists. Worse, many of the Linux guests continued to respond to management station polls just fine . . . but, of course, at some point, being unable to write to disk impacted their behavior. *Note to self: Upgrade monitoring strategy to include polls which incur disk writes.* Finally, the VMware hosts themselves lost touch with the VMware console after this event, making them unmanageable via the GUI and requiring reboots of each host to restore this function. Crawling through the lists of VMs and rebooting the ones which were hung or read-only took hours, followed by a multi-day process of migrating guests off a host, rebooting the host, and migrating guests back onto it.

After that, we started learning about dynamic multimode VIFs and LACP. During our most recent test of a datacenter, the NetApps and VMware cluster rode through without blinking—we rebooted switches repeatedly, and they didn't break a sweat (dropping only the occasional ping). At this point, our filers literally don't care about the loss of an Ethernet switch.

Subtleties

Even without the fancy LACP configuration, most of our NetApps have ridden through the loss of a switch without a problem, because our switches tend to toggle link, if only briefly, when they reboot. So when Switch A reboots, for example, it drops link on all ports for a few seconds. ONTAP notices the loss of link and flips to a backup NIC attached to Switch B. The first switch does its stuff, returns to life, and begins to service traffic—though the Filer ignores it, happily using its "b" side NIC.

In the NetApps backing our VMware cluster, however, we had started to use the *favor* command, which instructs ONTAP to fail back to the primary NIC, once link is restored. That, of course, got us into trouble—line cards in a rebooting switch will transmit the link signal early in the reboot process, long before the brains card will forward frames. *The lights are on, but no one is home.* As a result, ONTAP would re-enable NICs attached to the rebooting switch too soon, sending traffic into oblivion. In my experience, this example illustrates several themes in the behavior of HA designs under stress:

- In general, HA systems handle losing a component more gracefully than returning a component to service.
- Failed components will sometimes oscillate on their way to returning to life or on their way to a permanent death.

Configuring HA systems to dawdle before returning a previously failed component to service helps protect against these issues.

Conclusion

Like everything else in this business, developing these techniques has taken me years of trial and error, and I expect to advance them further as I continue to better understand how to migrate transport infrastructure from Highly Unavailable to Highly Available configurations.

Note that I have focused on one particular Ethernet/IP datacenter design; the configuration choices I sketch here apply to that particular design. Different transport designs call for different configuration choices in bonding, Teaming, LiveLink, and LACP.

In the future, I hope to expand my validation toolkit to include read/write tests across storage devices and pings across Fibre Channel networks, to tackle the challenge of verifying the failure behavior of application-layer clusters, and to develop tools that proactively identify flaws in configurations.

References

[1] Cisco Datacenter Infrastructure Design Guide: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmigration_09186a008073377d.pdf.

[2] Trey Layton, Virtual Port Channels, Cross-Stack EtherChannels, MultiChassis EtherChannels: http://communities.netapp.com/blogs/ethernetstorageguy/2009/09/23/virtual-port-channels-vpc-cross-stack-etherchannels-multichassis-etherchannels-mec--what-does-it-all-mean-and-can-my-netapp-controller-use-them.

[3] Howard Goldstein, Storage Network Design, Performance, and Troubleshooting: http://www.hgai.com/index_files/Page488.htm.

[4] Linux Ethernet Bonding Driver HOWTO: http://www.kernel.org/doc/Documentation/networking/bonding.txt.

[5] Discussion of subtleties on the Bonding Development list: http://sourceforge.net/mailarchive/forum.php?thread_name=AANLkTimhWimxhQZ__i-eNU%3DYFJaXgQU_5J%3DLDUmO%3DFY0%40mail.gmail.com&forum_name=bonding-devel.

[6] Intel Advanced Network Services Software Increases Network Reliability, Resilience, and Bandwidth: http://www.intel.com/network/connectivity/resources/doc_library/white_papers/254031.pdf.

[7] How to Configure Teaming Modes on Intel Network Adapters: http://www.youtube.com/watch?v=GQ4WGGdlqpc.

[8] Network Connectivity Advanced Networking Services: http://www.intel.com/support/network/sb/cs-009744.htm.

[9] Configuring LiveLink for a Smart Load Balancing and Failover Team: http://support.dell.com/support/edocs/network/P29352/English/bacs.htm#configuring_livelink.

[10] Stuart Kendrick, Configure HA Servers in Datacenters: http://www.skendric.com/philosophy/Configure-HA-Servers-in-Data-Centers.pdf.

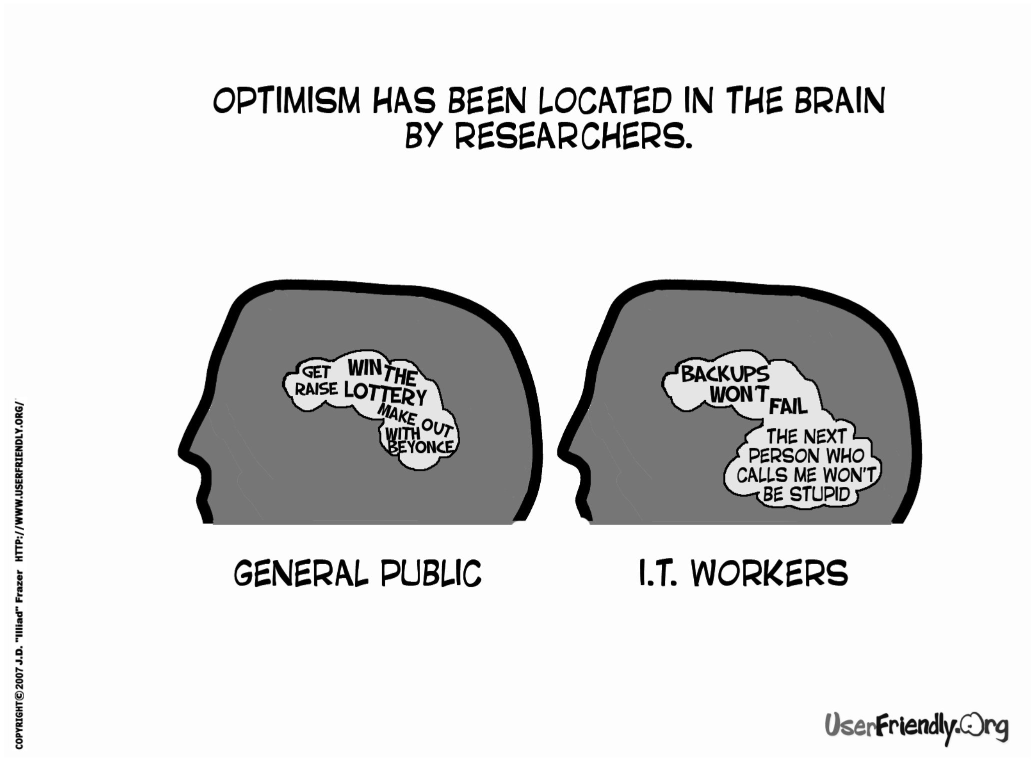
[11] Trey Layton, Multimode VIF Survival Guide: http://communities.netapp.com/blogs/ethernetstorageguy/2009/04/04/multimode-vif-survival-guide.

[12] Stuart Kendrick, Focus on NetApp: http://www.skendric.com/philosophy/Toast-Ethernet-IP.pdf.

[13] Mass-Ping: http://www.skendric.com/nmgmt/polling/mass-ping/.

[14] Red-Reboot: http://www.skendric.com/nmgmt/device/Cisco/red-reboot.

[15] Stuart Kendrick, "A Few Thoughts on Uptime," pp. 64-65: http://www.skendric.com/philosophy/A-Few-Thoughts-on-Uptime.pdf.


OPTIMISM HAS BEEN LOCATED IN THE BRAIN BY RESEARCHERS.
GET RAISE
WIN THE LOTTERY
MAKE OUT WITH BEYONCE
BACKUPS WON'T FAIL
THE NEXT PERSON WHO CALLS ME WON'T BE STUPID
GENERAL PUBLIC
I.T. WORKERS
COPYRIGHT© 2007 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
UserFriendly.Org

COLUMNS

Practical Perl Tools

Give as Good as You Get, My Tiny Dancer

DAVID 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

During our last time together, we had a chance to explore some of the features of the Web library for Perl, the seminal HTTP client distribution (more commonly called LWP). We saw how to fetch HTTP content from Web servers, POST data to them, and so on. I thought it might be interesting to look at the other side of the coin and explore another way to construct Perl applications that serve data to HTTP clients like those from my April column. I say "another" way because all the rabid fans of this column (I'm waving to both of you!) will recall our forays into the CGI::Application framework back in July and September of 2009. CGI::Application is still alive and kicking, but since then there have been a number of new frameworks released that some considered to be the new hotness. In this column we'll look at one of those frameworks, and, if polling numbers stay high, we'll look at a "competing" framework in the next issue.

NEWSFLASH: Before we get to that framework, a quick newsflash related to last issue's column. After the column was submitted (but perhaps before it was printed), a new 6.0 version of the LWP distribution was released. It was largely a revision related to which modules were bundled, but there was one change that is likely to make a difference to readers of this column. I'm indebted to David Golden who quoted one of the new parts of the LWP::UserAgent documentation in his blog:

> If hostname verification is requested, and neither SSL_ca_file nor SSL_ca_path is set, then SSL_ca_file is implied to be the one provided by Mozilla::CA. If the Mozilla::CA module isn't available SSL requests will fail. Either install this module, set up an alternative SSL_ca_file or disable hostname verification.

Short translation: you probably want to install the Mozilla::CA module if you plan to make https requests using LWP. You could set PERL_LWP_SSL_VERIFY_HOSTNAME to 0 to disable hostname verification, but that would be considerably less secure. Just a quick heads-up that hopefully will save you a bit of a surprise when you upgrade LWP. Okay, onward to the main attraction.

Dancer

One of the things I appreciate about CGI::Application is its (initially) simple model of the world. In CGI::Application, each page is associated with something it calls a run mode. Each run mode could consist of a subroutine whose job it was to produce the output for that run mode. Simple Web applications are indeed that simple, although as things get more complex in a CGI::Application application, so does the code and its control and data flow. For this column, let's look at another framework that aims for simplicity. Dancer started out as a straight port of a deservedly much praised Ruby Web application framework. No, not that one. Dancer is a port of Sinatra, a Ruby framework considerably simpler than Rails.

How simple? Although this example from the documentation tends to come up in any discussion of Dancer, you'll forgive me if I feel compelled to repeat it here as well:

```
use Dancer;

get '/hello/:name' => sub {
    return "Why, hello there " . params->{name};
};

dance;
```

Let's take this example apart in excruciating detail, because Dancer's simplicity is directly related to its concision. The first line is easy: we're just loading the module. The next lines define what Dancer calls a "route." Routes are specifications for incoming requests and how to handle them. They consist of the kind of request (in this case a GET), the path being requested (i.e., the part of the URL after the server name), and what to do with the request (i.e., what code to run when it comes in). The last two parts of the route definition are the more complicated aspects of the route lines above, so let's go into further detail.

In the example above, the specification says to look for incoming requests that match "/hello/:name". The first part, the part between the slashes, looks reasonable, but what's that ":name" part? Anything that begins with a colon is meant to indicate a placeholder accessed by that name (the doc calls it a named-pattern match). In this example, it means our code is looking for a request of the form:

```
http://server/hello/{something}
```

When it finds it, it places the component of the path represented by {something} into a parameter called "name". The code that gets run as part of this route looks up the "name" parameter in the params hash reference Dancer makes available to all routes and returns it as part of the subroutine's return value.

This is just one kind of pattern you can specify for a route. Dancer also makes full regular expression matches, wildcards, and conditional matches available in its specification. It also lets you write:

```
prefix '/inventory';
```

before each route handler, and that handler will add that prefix before the specified pattern. So "/shoes/:size" matches as if you specified "/inventory/shoes/:size" when that prefix is set. Dancer also lets you define a default route if desired.

If a pattern matches, it runs the code associated with the route. That code is responsible for providing a response to the incoming request. The example above is the simplest kind of response (essentially just a scalar value); we're going to get more sophisticated in just a few moments. Dancer provides hooks should you want to operate on a request before it gets processed or after the response has been generated. The documentation gives an example of code you could run in a hook to handle a request in a different way, depending on the logged-in status of the user making the request.

The final line in the example code ("dance;") may be the most poetic, but that just spins up the framework after all of the definitions are in place and starts accepting requests. If the Bob Fosse–like ending to your scripts doesn't really work for you (or if your boss is going to read your code and has a distinctly prosaic heart), you can use the more humdrum command "start;" instead.

So how does this thing work? First you need Dancer and its prerequisites installed on your system. Ordinarily, I wouldn't mention such an obvious detail, but I'm taken with the package's intro page, which points out that you can either install it on a UNIX system using the standard CPAN.pm/CPANPLUS invocations or use the very cool cpanminus script like so:

```
# wget -O - http://cpanmin.us | sudo perl - Dancer
```

(If you leave off the sudo invocation, cpanminus will install the whole kit and caboodle into a perl5 directory in your home directory.) I'm sure cpanminus will make another appearance in this column in the future.

As a quick aside, on an OS X machine using a version of Perl from MacPorts, I needed to set `PERL_CPANM_OPT='--local-lib=/Users/dnb/perl5'` in my environment and add `--no-check-certificate` to the wget command line to allow the non-sudo version of that command line to work. Once Dancer was installed, I could add `use local::lib;` at the beginning of my code to use the stuff installed in ~/perl5.

With Dancer on your system, congratulations, you now have a Web application complete with its own Web server, simply by running the script:

```
perl yourscript.pl
```

This will spin up a server listening on port 3000 on the local machine:

```
$ perl dancer1.pl
>> Dancer 1.3020 server 88541 listening on http://0.0.0.0:3000
== Entering the development dance floor ...
```

(and in another window)

```
$ curl http://localhost:3000/hello/Rik
Why, hello there Rik
```

This mini Web server is meant to be just for development. This would not be the way you'd actually want to deploy your Web app in production. The Dancer::Deployment documentation goes into the more robust methods for that (CGI/fast-cgi, Plack, behind a proxy, etc.).

Web Apps That Produce Output Are Generally More Interesting

In our exploration of CGI::Application, we took our time getting to the notion that one would want to use some sort of templating mechanism when writing Web applications. These mechanisms allow you to write static HTML into which dynamic content is inserted at runtime. This lets you create output that hews to a standard look and feel (e.g., same headers/footers/CSS styles) and is much easier than writing a bunch of print() statements yourself. For this intro, let's not pussyfoot around and instead dive right into using what Dancer has to offer in this regard.

Dancer lets you easily pick from at least two templating engines: its rather basic built-in engine and the pervasive engine for most Perl Web frameworks, Template

Toolkit. The Dancer engine (Dancer::Template::Simple) only does simple replacements. If you write:

```
<% variable %>
```

it will replace that string with the value of `variable`. If you decide to go with the other engine, there's a whole book on Template Toolkit and tons of online documentation if you'd like to explore the full range of template power at your disposal.

To use either engine, you place the template in a subdirectory of your application directory (more on directory structure later) called "views," with a name that ends in .tt. This template gets referenced in the subroutine you defined in your route specification, like so (as seen in another example from the documentation):

```
get '/hello/:name' => sub {
    my $name = params->{name};
    template 'hello.tt', { name => $name };
};
```

In this sample, we generate output by processing the hello.tt template. We pass in a parameter hash that replaces the variable "name" in that template with the value we retrieved as part of the path when the request comes in. If views/hello.tt consisted of:

```
<p>Why, hello there <% name %>!</p>
```

we'd get the same output from our `curl` command as before except that it would have paragraph tags around it.

Dancer provides, for lack of a better word, a meta-version of the view templating functionality called "layouts." Layouts let you create a single view that "wraps" or interpolates other views in order to produce the look-and-feel consistency we mentioned at the start of this section. You might create a layout that specifies the header and footer information for every page on your site. Such a layout would look like this:

```
(header stuff here)
<% content %>
(footer stuff here)
```

The content variable above is magic. When any template command gets processed, the layout view is returned with the results of the processed view inserted right at that spot. This allows you to avoid having all of that header and footer stuff copied into every view for your site. One way you could imagine using layouts would be to have a separate view template for every section of your Web site, each of which is inserted into the layout for the site before being sent to the browser.

Directory Assistance

In the previous section I mentioned that, by default, Dancer expects to see its templates in a views directory rooted off of the main application directory. There are a number of other defaults and expectations. Perhaps the easiest way to get them all out on the table, even though we won't have space to explore them all, is to see what happens when we use the helper script called "dancer" that ships with the distribution. Like many other Web frameworks, Dancer provides this script so that you can type one command and have an entire hierarchy of stub files for all of the different files you may need to construct an application in one swell foop. Here's Dancer's take on that process:

```
$ dancer -a usenixapp
+ usenixapp
+ usenixapp/bin
+ usenixapp/bin/app.pl
+ usenixapp/config.yml
+ usenixapp/environments
+ usenixapp/environments/development.yml
+ usenixapp/environments/production.yml
+ usenixapp/views
+ usenixapp/views/index.tt
+ usenixapp/views/layouts
+ usenixapp/views/layouts/main.tt
+ usenixapp/lib
  usenixapp/lib/
+ usenixapp/lib/usenixapp.pm
+ usenixapp/public
+ usenixapp/public/css
+ usenixapp/public/css/style.css
+ usenixapp/public/css/error.css
+ usenixapp/public/images
+ usenixapp/public/500.html
+ usenixapp/public/404.html
+ usenixapp/public/dispatch.fcgi
+ usenixapp/public/dispatch.cgi
+ usenixapp/public/javascripts
+ usenixapp/public/javascripts/jquery.js
+ usenixapp/Makefile.PL
+ usenixapp/t
+ usenixapp/t/002_index_route.t
+ usenixapp/t/001_base.t
```

Yowsa that's a lot of files. Let's see if we can break this apart so it becomes a little less scary.

```
+ usenixapp/bin/app.pl
+ usenixapp/lib/usenixapp.pm
```

is essentially where you'd find the script we've been writing so far. The former imports the latter like a module and starts Dancer.

```
+ usenixapp/views
+ usenixapp/views/index.tt
+ usenixapp/views/layouts
+ usenixapp/views/layouts/main.tt
```

Here is where the views we've already talked about are stored.

```
+ usenixapp/public/*
```

Dancer places all of the static files into a public directory. This include CSS, HTML error pages, jQuery library, and so on.

True to the Perl tradition, Dancer wants you to write tests for your code:

```
        + usenixapp/t/*
```

Yay, Perl (and Dancer).

This just leaves a few mysterious stragglers:

```
        + usenixapp/config.yml
        + usenixapp/environments
        + usenixapp/environments/development.yml
        + usenixapp/environments/production.yml
```

Dancer makes it very easy to construct and read YAML-based configuration files for your application. In these files, you can specify things such as the templating engine choice we discussed before, what level of debugging to use, how logging will take place, how sessions are stored, and so on. Dancer lets you keep a general config file plus separate config files that can be used depending on how you are using your application (is it in development? production?). Dancer can keep configuration information in the code itself, but, as you can see above, it strongly suggests that you keep that sort of information in dedicated files for this purpose.

I Could Dance All Night

Unfortunately, we're almost out of space, so we're going to have to get off the dance floor and just list some of the functionality in Dancer we won't be able to explore. The documentation is good, so hopefully my just mentioning these capabilities will encourage you to read further to see how these things are accomplished in Dancer. Dancer, like the other frameworks, provides a number of ways to keep "sessions" (i.e., some permanent state between requests, such as for the user who is logged into an application). It has useful logging support that allows the application to log debug messages. It can (with the help of a separate plug-in) make AJAX programming easier (routes can return JSON and XML easily). It can cache routes for better performance if you'd like. There's lots of spiffy stuff in this one distribution, so I would encourage you to check it out for at least your smaller Web application needs.

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

iVoyeur
A Handful of Gems

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

Fall is my favorite time of year. Not because of the leaves, or the pumpkins, or the other usual reasons (although the weather is nice). I love fall because fall is when the auditors leave. Beautiful fall, when they scurry back to whatever putrid swamps hatched them and I am free.

Just now, however, loathsome spring is beginning to settle in, and I close my window shades in dread of this my absolute least favorite time of year. Outside my window the pecan trees bud, the daffodils bloom, and I sense them stirring. Auditors across the country slithering to life inside the slick protective shells of their pupae, blinking their blind eyes and clutching their binders and clipboards close to their heart-like organs as they come fully to whatever passes as awareness for them.

Another 40 days, maybe less, and they'll descend on us like a blight. We'll mark their coming when they blot out the sun with appointments, reminders, and pre-engagement questionnaires. We'll know they're upon us when they dance from the shadows like marionettes, demanding answers to questions they don't understand. I'll be made to dance with them, these surreal pawns of an unseen overlord in a Shakespearian nightmare.

For months our dance will bring my professional life to a grinding halt. The dance will consume my productivity entirely, spinning us 10 hours a day up mountains of Word docs, and leaping us through fields of Dia diagrams [1], and all the while answering and answering and answering their strangely worded and context-less questions. Are my inputs validated? Do my authentications have two factors? Exactly how many bits do my encryptions have? Cha cha cha.

When they ask me what version of Active Directory I use, I will have to tell them that I do not use Active Directory. "Tsk tsk," they'll say to each other, marking things down on their clipboards. They'll exchange knowing glances and inform me of the need to remediate our lack of account management. I will then be forced to explain that we use OpenLDAP for account management. Their eyes will narrow, and they'll look at each other uneasily, suspicious that I'm making things up off of the top of my head. I know this will happen because it's happened every year for the past three, but the auditors have no memory. The auditors are freshly hatched each year and are not burdened by the weight of the past like you or me. The auditors don't need memories; they have clipboards and checkboxes instead.

That uneasy murmur in fact is about the only fun I'll have for the next few months. I know it's wicked of me, but I do so relish their discomfort when, for example, I reply "Snort" to their oddly phrased question about whether my networks contain IDS. A quirky answer befitting an odd question. Their pencils will hover above the checkbox, their indecision palpable. Is he making that up? No one would bring an IDS product to market called "Snort," would they? Unable to decide, they'll move on to the next question without marking down an answer just yet, only to be told that I update my IDS definitions with "OinkMaster." You should be there: it's delicious.

These tools—so lovingly created and whimsically named by my brothers and sisters in the nerdosphere—are like a handful of gems in my pocket. I pull them out and their brilliance dazzles the auditor hoards into confusion and dismay. I'm not working on any big monitoring projects at the moment, so I thought this month I'd share with you a few little gems I'll have in my pocket for the auditors this year.

Ganglia GWeb 2.0

My first article about Ganglia began with a lengthy rant about the sorry state of RRDTool display engines. The "spherical cow" [3] of RRDTool display engines has, in my mind, the following characteristics:

- It is polling-engine agnostic.
- It can read my RRDs from multiple directories, anywhere in the file system.
- It will show me graphs from any RRDs it can find without me having to configure anything.
- I can easily (in as few clicks as possible) tell it to create new graphs of aggregated data from the RRDs it knows about.
- It allows me to save these new graphs for later reference.
- It will provide a predictable URL to any graph it can show me, whether auto-detected or saved.
- It allows me to change the RRD options, such as size or date-range, of any saved or dynamically generated graph by modifying its URL.

Ganglia's polling engine and Web interface are designed to show graphs in the context of machine clusters, so the two will probably never be independent, but their ongoing Web interface redesign (dubbed GWeb 2.0 [2]) has addressed nearly every other one of my qualms. To the existing Ganglia Web interface it adds a search feature capable of displaying graphs matching regex search criteria, an aggregation feature that can aggregate data from any graphs that match a regex, and much more.

The displayed graphs now support specifying the time-range via a drag-box (e.g., Cacti [4]), and there are also text entry fields for specifying custom time ranges. Every graph I click on sends me to a linkable version of that graph. I can change the size, date-range, and myriad other RRD options (but not quite all of them) by modifying the URL parameters in these linkable graphs. This effort is really coming together nicely.

MonAmi

Speaking of separating polling engines from display engines, that same rant outlined some of the things I'd like to see in a hypothetically perfect polling engine:

- It speaks more than just SNMP.
- It provides sane defaults that are easily discovered and changed.
- It uses plug-ins to define data sources like SNMP daemons, external monitoring systems, and log files.
- If it stores data in RRDs, I can easily set whatever custom RRDTool attributes I want per device per metric.

I recently came across a fascinating little project called MonAmi [5], which seems to fit the bill. It aims to be a universal sensor network and all-around monitoring data middle man. MonAmi does not provide graphs or attempt to send notifications by design. Instead it is a pure data collection engine, using plug-ins to define data sources and endpoint destinations. It supports 16 data collection plug-ins, including plug-ins for the Apache Web server, Tomcat, and MySQL, and more generic system metrics such as file system, kernel process, and socket statistics.

Internally, MonAmi maintains data metrics in a local file system, and can aggregate and transmit the metrics it collects via transmission plug-ins to 12 different endpoint systems, including Nagios [6], Ganglia, MonaLisa [7], and GridView [8], as well as to log files and MySQL databases.

This, in my opinion, is an excellent design that wants only more plug-ins. Unfortunately, SourceForge lists its last code update as “2009-12-29,” and traffic on the forums appears to have died off in 2010. The code is stable, however; we’re currently using it on a few production hosts to move our Tomcat JVM metrics to Ganglia. I would really like to see someone pick up this little gem, polish it off, and love it.

JavaMelody

JavaMelody [9] is a stand-alone Java class implemented as a JAR file. It is easily integrated into your Web app by simply slipping it into the WEB-INF/lib directory and adding a few lines to the Web.xml. JavaMelody is about as lightweight as a JVM monitoring app can be. It doesn’t require that JMX be enabled, does not use profiling, and keeps no database of its own.

For your trouble you get 14 metric oodles of information about the running application; 28 application-specific metrics ranging from memory and CPU utilization to hit metrics such as HTTP sessions and active threads; and really interesting JDBC info such as SQL and Spring mean connection times. All of these are stored on the local file system in RRD format and graphed for you on a report page located at http://[servername]/monitoring.

But wait, that’s not all! The real-time Web-based report has a litany of text-based stats and information about the system itself, including process, thread, mbean, and JDBC connection viewers; JVM info including a full dump of all the class files and their versions; and tables of statistics about HTTP, Spring, and SQL connections.

I really wish JavaMelody would take the entire heap of this data and multicast it in XML format to gmond. It would be a windfall to have all of this centrally located in Ganglia. I’d dig into it myself, but I’m going to be busy for the next few months.

Snoopy Logger

If you have a centralized syslog infrastructure and you’re looking for a lightweight and easily manageable way to get an audit trail of every command executed on your Linux servers, Snoopy Logger is just the ticket [10]. Implemented as a shared library, which you place in ld.so.preload, Snoopy intercepts every call to execve() and logs the UID, PTY, and current working directory of every command executed to authpriv. Here’s a sample log line:

```
Mar 25 21:19:15 vlasov snoopy[12931]: [uid:1000 sid:5927 tty:/dev/pts/4 cwd:/
home/dave filename:/bin/cat]: cat /var/log/auth
```

If you're really serious you probably want something like GRSecurity [11] or SELinux [12], but in a layered approach, along with a file system integrity monitor, host-based IDS, and a centralized syslog infrastructure. It's an awesome little tool that does just exactly what it should, and with a name like "Snoopy" it's *bound* to go over with the auditors.

I hope you found something in there worth adding to your own bag of gems.

Take it easy.

P.S. Re-reading the intro, I realize that I may have been just a tad hyperbolic. The security auditors in this audience are, I know, just as frustrated with the ongoing security theatre as the rest of us and aren't to blame for any of it. The intro was intended to give you a smile, and also as a literary device to introduce some color into the subject matter.

P.P.S. Please don't flame me.

References

[1] Dia, a drawing program: http://projects.gnome.org/dia/.

[2] GWeb 2.0: http://ganglia.info/?p=343.

[3] Spherical Cows: http://en.wikipedia.org/wiki/Spherical_cow.

[4] Cacti Monitoring System: http://www.cacti.net/.

[5] MonAmi: http://monami.sourceforge.net.

[6] Nagios: http://www.nagios.org.

[7] MonaLisa: http://monalisa.caltech.edu/monalisa.htm.

[8] GridView: http://gridview.cern.ch/GRIDVIEW/dt_index.php.

[9] JavaMelody: http://code.google.com/p/javamelody/.

[10] Snoopy Logger: https://sourceforge.net/projects/snoopylogger/.

[11] GRSecurity: http://grsecurity.net/.

[12] SELinux: http://www.nsa.gov/research/selinux/.

/dev/random

ROBERT G. FERRELL

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

I stayed home from work for the past few days because I picked up a nasty virus while on vacation in southern Mississippi. Lying in bed thinking is always a ripping adventure for me, because my brain loves to take off on cross-country jaunts and tumble into half-hidden abysses from which there is no easy extrication.

Having been trained as a biologist, I tend to think in those terms no matter the subject. It was, I suppose, inevitable that I would eventually decide to analyze the overall suitability of the malware:virus analogy. To begin, we must first familiarize ourselves with the froo-froofery that lies on each side of this equation.

A biological virus is essentially a horrifyingly efficient little robot. It uses chemistry to track, latch onto, and inject a cell with its own genetic material. When the infected cell starts the reproduction process, it just keeps making copies of the virus until it pops and releases them, ready to start the process anew *ad nauseum*. Viruses don't, as a rule, have any payload per se other than their own DNA. They're just programmed to make more viruses, like those tiny DVD replicators that creep under the door into your house and use up all your blank disks to make endless copies of ancient "My Mother the Car" episodes while you're asleep. That doesn't happen to you? Never mind. My mother (Cord L-29 Special Coupe, light blue with wide whitewalls) always said this was an odd neighborhood.

Computer viruses, in contrast, almost always seem to have a less-than-pleasant payload attached. This can range from zombie-inducing botnet software, to utilities that damage the host system (intentionally or otherwise), to simple "gotchas." This means effectively that they more closely resemble pathogenic bacteria than viruses. Infectious bacteria frequently sport either endo- or exotoxins that wreak havoc on the host organism. The deleterious effects of viruses are mostly by-products of the fact that host cells can't really do their job once they're, you know, *lysed*.

Lounging about the domicile, fending off invisible legions of microorganismic doom, facilitates all manner of half-cocked cogitation. I found myself, for example, wishing I could use vi commands and regular expressions during face-to-face conversation. Jump lines, go back a paragraph, skip to the end, substitute, insert, delete all strings that contain "like" or "awesome"—that kind of thing. That would be, like, awesome. Especially :q!

What if life, I now feverishly speculate, were more like UNIX? I could bring up the process table at will to see what tasks were running in my background, and *kill -9* any I didn't want. I could look at memstat or iostat to examine what was bothering me or find out what I was supposed to be doing. Better, I could just mod my kernel and reload to solve any pesky medical issues. All of those little seemingly insoluble problems I encounter on a daily basis could be settled by calling up the man page. (Well, for the most part. I've encountered a few man pages that look as though they were written by an alien engineer describing the process for adding quark injection to your galaxy hopper's hyperdrive. I tried that once and it *really* gums up the intake manifold.)

As I drift lazily along the sludge-like stream of my overheated subconscious, the sky overhead now begins to darken ominously and thunder rumbles in the distance as we enter that forsaken region known only as "The Cloud." Here is a rare glimpse of the underworld, replete with blaze-eyed daemons, towering black grisly giants, and puissant phantasms eternally ripping and tearing at the firmament. Leviathan doom machines grind away tirelessly in bottomless pits filled with raw data, whilst toothless grinning gargoyles stand guard. Their cockpits glowing with hideous evil, jagged sable dive-bombers swoop and scream among the fluttering packets, fragile butterflies in the bombastic blitzkrieg. At every turn unblinking red eyes stare out from the darkened thickets, heavy with menace.

While Jim (I forgot to mention him) wrestles the saber-toothed beast known only as "QoS" to the oil-streaked tarmac, I dodge and weave to avoid the hordes of vicious stinging "features" that swarm suddenly from gaping fractures in the towering wall of fire separating us from the swirling maelstrom of fractal turmoil raging outside our beleaguered perimeter. Lava-like bit torrents erupt without warning from fissures that split and seal at random with a sound that resembles nothing if not the angry snort of some great sea serpent, enraged by the white-hot fluid belching from its orifices. Ships flying the flags of all the well-known anti-virus vendors pitch and rock anchored firmly in the harbor, helpless against the rapidly rising seaweed-choked swells rushing in from the Malware Sea. Children huddle against their mothers and wail inconsolably along the docks as their treasured stuffed penguins are dashed against the needle-sharp rocks of Scada Point.

Fire, brimstone, sobs, and swirling toxic fog fade at last into an idyllic scene of gently rolling green hills dotted with happy little thrumming servers, each the king of its own sovereign domain of countless unseen workstations. The peaceful serenity is broken only occasionally by a wandering slime monster, known to the local tribes as "The Ravaging Replicator," with a large anvil-like appendage that smashes servers at random and leaves them hemorrhaging data copiously over the verdant sward. Replicators, it turns out, have no natural enemies, so the only thing that can be done is wait until one experiences the inevitable page fault and then drive a sharpened poignard deep into its exposed BIOS.

All fevers eventually break, and when mine did the world reverted to a much less interesting place. On the bright side, the path to the bathroom is no longer fraught with deadly digital dreamworld diaspora. For this, at least, I am grateful.

Book Reviews

ELIZABETH ZWICKY, WITH EVAN TERAN AND RYAN MACARTHUR

The Visible Ops Handbook: Implementing ITIL in Four Practical and Auditable Steps
Kevin Behr, Gene Kim, and George Spafford
IT Process Institute, 2005. 97 pp.
ISBN 78-0975568613

Don't be fooled by the low page count; there's a lot of information stuffed into that space. It suffers somewhat from having been stuffed. Traditional book design takes a lot more paper, what with the bigger letters and the generous empty borders, but it makes a real difference in readability. However, once you get used to reading the small, densely packed type, you will find that it is a believable account of how to get an IT operation under control.

The authors clearly have experience with real organizations that they have had to actually fix, so they recommend practical steps (for instance, grandfathering every piece of software in use onto the approved list, and slowly paring it down as you move forward). It does what it sets out to do, which is give you a working compromise between ITIL and reality.

Although most system administrators should be able to find something useful here, it's most useful to straight-up IT organizations: people who work in companies that make widgets, or sell widgets, or insure widgets. It maps closely to established computer companies, but if you're at a start-up, you'll have to do some translation, and if you're at a university, the translation is going to have to be pretty liberal.

Visible Ops Security: Achieving Common Security and IT Operations Objectives in Four Practical Steps
Gene Kim, Paul Love, and George Spafford
IT Process Institute, 2008. 108 pp.
ISBN 978-0975568620

Security Metrics: Replacing Fear, Uncertainty, and Doubt
Andrew Jaquith
Addison Wesley, 2007. 299 pp.
ISBN 978-0-321-34998-9

Here are two books on the practicalities of implementing computer security in a big network, which is, sadly, a great deal more about committee meetings than it is about bad guys and exciting technology. Since computer security education is a great deal more about the technology, there is a painful gap to be bridged. *Visible Ops Security* does a good job of bridging those gaps on the purely political end; who do you need to talk to and how do you get them to talk to you? *Security Metrics* is aimed at giving you something to say to management that is also useful to you, which implies that it is strongly based in reality and therefore will not leave you with that vaguely slimy feeling you get from simply making up numbers that seem plausible and support your position.

I like them both a lot. *Visible Ops Security*, not surprisingly, has a lot in common with *The Visible Ops Handbook*, including the dense typesetting and the focus on traditional companies, which may be a drawback for people for whom auditors are mythical beings. Still, it's hard to beat something that suggests ways of getting cooperation. Furthermore, it has a good background on what it's like when security relationships aren't working, which you may find therapeutic—it's always nice to know you're not alone.

Security Metrics also has the voice of a real practitioner (genuine sarcasm, frustration, and silliness included) and not only talks about metrics good, bad, and dangerously fictional, but also gives advice on what to do with them once you've collected them. This includes extremely basic statistics and somewhat more advanced charting advice. As puzzling as this may seem if you're fluent in statistics, knowing about medians and quartiles can be life-changing if you've been stuck wondering why the mean is so poor at characterizing any data you care about, and remembering vaguely that there are supposed to be some other averages out there.

Neither book is going to give you a detailed recipe that you can immediately apply to your organization and be better, safer, and happier next week. This is at least in part because no recipe for doing this exists. Each book gives you some guidelines you can use to construct and implement a plan that will make your site safer and happier next year, which is quite audacious and useful enough.

Mining the Talk: Unlocking the Business Value in Unstructured Information

Scott Spangler and Jeffrey T. Kreulen

Pearson Education, 2007. 208 pp.
ISBN 978-0-13-233953-7

Book reviewing is full of surprises. A lot of them are sad, but every so often you find an unexpected pleasure, and this book is one. I was ready for a lot of things when I started reading it, but what I got was utter practicality; here's an algorithm I kind of knew about (clustering) with all the information you need to turn it into an effective tool for slicing and dicing types of data sources that don't lend themselves to exploitation with regular expressions.

You know that request queue that you're sure is trying to tell you something useful, but nobody can quite figure out what? (Somehow, those categories you predefined just aren't working.) These guys know why it doesn't work and what you can do that will work, so that with an entirely reasonable amount of effort you can sort the requests into categories and correlate those categories against whatever else you've got lying around in the request data. Getting from there to something you can fix is your problem.

What you get here is the algorithm and the practical advice on the tools you surround it with to make things work. There's a sample implementation you can download (it's Java, but set up for a PC), and a bunch of descriptions of the kinds of problems you can apply it to, and what happens when you do that. Some of the problem categories were of more interest to me than others, but I'm happy to have a shiny new tool to play with.

Picturing the Uncertain World: How to Understand, Communicate, and Control Uncertainty Through Graphical Display

Howard Wainer

Princeton University Press, 2009. 227 pp.
ISBN 978-0-691-13759-9

I like this book, but not as much as I would have liked the book that I was imagining, based on the title. This is a book about statistics, graphics, and understanding, something that fans of Edward Tufte will like, with a really nice chapter on how to make graphs that accurately depict significant differences and insignificant differences. It has practical moments, but for the most part, it's more about background than it is about advice and techniques. It's an enjoyable wander through statistics-related topics with particular attention to uncertainty, but not dense with useful illumination.

Web Application Obfuscation

Mario Heiderich, Eduardo Alberto Vela Nava, Gareth Hayes, and David Lindsay

Syngress, 2010. 282 pp.
ISBN 978-1597496049

Despite what you may assume from the title, this book is not about obfuscating Web applications themselves, but instead talks about things from the attacker's perspective. In other words, it answers the question, "How can I make my attack code get by the various types of filters put in place by Web applications?" Normally, I am a low level, native code guy, so this was a nice change of pace from my usual reading.

This book is broken down into a few distinct parts:

- Introduction (Chapter 1)
- Language-level attacks (Chapters 2–7)
- Bypassing Web application firewalls (WAFs) and client-side filters (Chapter 8)
- Mitigation techniques (Chapter 9)
- Future developments (Chapter 10)

This breakdown is actually a very nice way to structure the book. If you're looking for some information specific to PHP, for example, then you know exactly which chapter to skip to (Chapter 6). Odds are, you will want to read the whole book, but having the information available in an easy-to-look-up form is nice too.

Chapter 1 is nothing more than a brief introduction to the concepts that will be discussed. The authors review why Web applications need to filter and some of the approaches that developers may use. Since regular expressions are a cornerstone concept for filtering, an overview of how they work is covered as well.

The next few chapters are where the really interesting stuff is. The book does a great job of discussing and comparing the various ways different user agents handle invalid markup and code. Sometimes one stands out as handling a particular situation better than the rest, but inevitably they all fail in some way, allowing the attacker to bypass the filters and still get the user agent to do what they want.

The authors discuss many concepts which, to be honest, I would not have thought of myself. Of particular interest was the concept of making the last character of your injected HTML the first byte of a multibyte unicode codepoint. For several browsers, this caused the next character in the HTML to get "swallowed" up, effectively removing it from the markup! That's pretty cool.

Things get a little crazy (in a good way) once we start into JavaScript and VBScript. The authors slowly ease us into the basics, and then throw us into the deep end with writing JavaScript which uses hexdecimal and octal escapes and, finally, "non-alphanumeric JavaScript," which allows the creation of valid code that is truly incomprehensible to the average person.

In the end, there are two core concepts being expressed in the book. First, for all of the covered languages there are ways to write code which has non-obvious functionality, often in the form of alternate ways of expressing characters (e.g., hexadecimal encoding). Secondly, even though the Web is based on many standards, the implementation of these standards varies, often wildly, when given unusual situations to deal with. This is especially true for HTML and JavaScript.

—Evan Teran

The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler

Chris Eagle
No Starch Press, 2008. 608 pp.
ISBN 978-159327-178-7

At 608 pages, the mere size of this book is intimidating; I would call it a tome. Before picking it up I had heard it was the "IDA Bible," which prompted me to acquire it. My job sometimes involves reverse engineering malware, and so I use IDA. I was a self-starter and never took a training class or watched an online IDA training video. I wish I had picked up this book right when I started reverse engineering, because it not only provides a taxonomy of IDA features but is also a primer on reversing pitfalls and how to overcome them leveraging IDA.

The book is separated into six parts, all of which are extremely valuable. Most technical books I pick up today I read haphazardly, but this book is one of the rare ones I read cover to cover (like Stevens's *TCP/IP Illustrated*). Hard to use at times, but rewarding in the end, this book is undeniably packed with useful examples, interesting asides, expert commentary, and real-world use cases.

Eagle spends no time explaining assembly (if you are using a disassembler, I would hope you are already familiar with whatever instruction set you are working with), and most of the examples are focused on x86-based Portable Executables running on Windows (although there are Linux ELF examples).

The "Advanced IDA Usage" section, one of my favorite parts, walks you through the process of creating an IDS file for OpenSSL, something I've now done, and it has actually improved my reversing capabilities, because statically linked OpenSSL code is now detected automatically. "Real-World Applications," another favorite section, was an amazing surprise, discussing common anti-debugging and anti-static analysis techniques, and how to mitigate these with IDA. All of the examples are available on http://www.idabook.com.

The author even delves into the black art of creating your own processor modules and demonstrates this capability by producing a Python module. Think of a processor module as the implementation of an instruction set architecture in IDA. Understanding how to write a processor module can come in handy when working with malware that uses custom virtualization handlers, which make IDA's x86 processor modules worthless. This topic is tough to find good documentation on; before this book, reading the other processor modules was the only way to learn. Not anymore: crack open this book and you will be on your way to writing your own processor modules, custom loaders, and more.

Every little nook and cranny was accounted for here, and it has transformed the way I use IDA. Now I'm armed with a better understanding of every aspect of IDA and can harness each to improve analysis. The current edition covers IDA up to version 5.3. The current IDA is 6.0, and most of the content is still relevant. The section on console mode is obsolete because OS X and Linux now have QT-based user interfaces (as does Windows); having said that, I must concede that console purists might still exist. There was strong focus on well-known processors and file formats, but this book does not advertise methodologies for reversing random firmware images ripped from bizarre hardware. You will probably come away from *The IDA Pro Book* with a better handle on IDA's internals to aid your reversing effort.

—Ryan MacArthur

Conference Reports

In this issue:

FAST '11: 9th USENIX Conference on File and Storage Technologies 59

Summarized by Puranjoy Bhattacharjee, Vijay Chidambaram, Rik Farrow, Xing Lin, Dutch T. Meyer, Deepak Ramamurthi, Vijay Vasudevan, Shivaram Venkataraman, Rosie Wacha, Avani Wildani, and Yuehai Xu

NSDI '11: 8th USENIX Symposium on Networked Systems Design and Implementation 80

Summarized by Rik Farrow, Andrew Ferguson, Aaron Gember, Hendrik vom Lehn, Wolfgang Richter, Colin Scott, Brent Stephens, Kristin Stephens, and Jeff Terrace

LEET '11: 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats 100

Summarized by Rik Farrow, He Liu, Brett Stone-Gross, and Gianluca Stringhini

Hot-ICE '11: Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services 106

Summarized by Theophilus Benson and David Shue

FAST '11: 9th USENIX Conference on File and Storage Technologies

San Jose, CA
February 15–17, 2011

Opening Remarks and Best Paper Awards

Summarized by Rik Farrow (rik@usenix.org)

The FAST '11 chairs, Greg Ganger (CMU) and John Wilkes (Google), opened the workshop by explaining the two-day format. Past workshops extended two and one-half days, but by leaving out keynotes and talks, they fit the same number of refereed papers, 20, into just two days. There was also a WiPs session and two poster sessions with accompanying dinner receptions.

The Best Paper awards went to Dutch Meyer and Bill Bolosky for *A Study of Practical Deduplication* and to Nitin Agrawal et al. for *Emulating Goliath Storage Systems with David*.

The total attendance at FAST '11 fell just three short of the record, set in 2008, with around 450 attendees.

Deduplication

Summarized by Puranjoy Bhattacharjee (puran@vt.edu)

A Study of Practical Deduplication

Dutch T. Meyer, Microsoft Research and the University of British Columbia; William J. Bolosky, Microsoft Research

Awarded Best Paper!

Dutch Meyer gave a lively presentation of this paper, winner of a Best Paper award, about deduplication research in the form of a study of the files involved. Meyer started by saying that dedup is a good idea, but he is not sold on it because of fragmentation and loss of linearity against the potential storage savings. Storage space being cheap, this is counter-intuitive. He said that dedup is important in cases where the tradeoff achieved is high. This is what this paper focused on. At Microsoft Research, the authors were able to recruit 875 people to investigate their file system states once a week for four weeks, yielding around 40 TB of data.

Analysis of the files revealed that the median file size is 4K and that the files are linear. However, a lot of the bytes are concentrated in a few files, mostly of type .iso (disk images) and .vhd (virtual hard disks). He also talked about the different kinds of dedup techniques out there, such as whole file, fixed chunk, and Rabin fingerprinting. Applying these techniques, they were able to achieve 70% storage saving with Rabin, 60% with fixed chunk, and 50% with whole file. With support for sparse files, another 8% savings were achieved. Further analysis showed that there are two classes of files—small at 10 MB and emerging class of big 8–10 GB files of type .iso, .vhd, .null, etc. They observed that Rabin works well on the big files, such as .vhd. Hence the authors suggested a mixed technique of doing whole file dedup on most files and Rabin on some special files, to reduce the gap of 20% between whole file and Rabin performance. Meyer also announced that they are working to release the dataset.

During the questions, Michael Condict (NetApp) pointed out that people usually care more about the amount of space remaining than about space saved. Assar Westerlund (Permabit) wondered about lower performance than expected for 4K Rabin, and Dutch answered that this might be workload specific, with more bytes in .vhd than earlier, making whole file dedup gains harder. He speculated that techniques to open up the .vhd and examine subfile structures could lead to more gains. Ben Reed (Yahoo) asked about the methodology of the one machine numbers. Bill Bolosky, the co-author, replied that it was for each of the 875 machines. Reed asked whether the dedup numbers were evenly distributed or heavily biased, but the authors had not looked into that. Chris Small (NetApp) had a suggestion for the authors to look into the tradeoff between smaller chunk size for dedups vs. local compression and backup.

Tradeoffs in Scalable Data Routing for Deduplication Clusters

Wei Dong, Princeton University; Fred Douglis, EMC; Kai Li, Princeton University and EMC; Hugo Patterson, Sazzala Reddy, and Philip Shilane, EMC

Fred Douglis presented this paper on the performance in dedup clusters with existing high throughput single-node systems. The requirement for clusters was motivated by situations where backups do not fit in a single appliance, and hence dedup clusters are needed that provide throughput comparable to single node along with load balancing on the cluster nodes. Douglis described how fingerprint lookup tends to be the bottleneck and how cache locality and disk avoidance via containers and Bloom filters can help. He described their approach of using super chunks (a consecutive group of chunks) and deduping node by node at chunk level. This means that the same chunk might appear on multiple nodes, leading to opportunities for balanced storage and scalable throughput, because routing super chunks provides better throughput than routing chunks. He discussed their techniques of stateless (with low overhead) and stateful (with more overhead but better load balancing) routing protocols. In stateless, chunks are migrated when a node exceeds 105% of average disk usage. In stateful, chunks are routed to nodes where it matches the best, but it avoids flooding by adding more new chunks. If no clear winner is found in terms of match, the algorithm defaults to a stateless approach.

He then talked about how contents of each node are tracked using a Bloom filter and their strategy of reducing overhead by sampling with a rate of 1/8. He then described their evaluation to find the best super chunk feature and granularity. They used trace data from a production environment with three large backup sets and five smaller sets with individual data types and one synthetic dataset. They found that a hash of 64 bits works well for small clusters and that bin migration indeed helps to reduce skew and increase effective dedup. Stateful routing was found to offer further improvement. Larger superchunks gave better throughput. He also cautioned that dedup improves after migration but may drop between migration intervals. He talked about the Global Deduplication Array product, which was based on the research presented here. Finally, he talked about future directions: investigating conditions causing data skew, scalability over a broad range of cluster sizes, and supporting cluster reconfiguration with bin migration.

During questions, Michael Condict (NetApp) asked if bin content was accounted for to maximize performance. Douglis replied that they did it in the simulator, but in practice they did not notice sufficient improvement. Geoff Kuenning (Harvey Mudd) said that the probability of troublesome chunk boundaries with Exchange was low, and asked if the authors had fiddled with the hashing algorithm and chunk boundaries to get around that problem. Douglis replied that if the masks were changed, Exchange could work better but other cases might be hit. If they were to start from scratch, a whole chunk hash instead of a first few bits might have worked but at large cluster sizes, a hash is only one of the factors that have to be accounted for. Keith Smith (NetApp) asked about hardware cost tradeoff and said that perhaps bigger nodes could perform better than a cluster. Douglis said that was a possibility. Finally, Christian (NEC) wondered if the tradeoff of 8% decrease in dedup was worth the 50% increase in throughput for routing by chunk and super chunk. Douglis replied that they were more concerned about throughput, so it was acceptable in their case.

Specializing Storage

Summarized by Xing Lin (xinglin@cs.utah.edu)

Capo: Recapitulating Storage for Virtual Desktops

Mohammad Shamma, Dutch T. Meyer, Jake Wires, Maria Ivanova, Norman C. Hutchinson, and Andrew Warfield, University of British Columbia

Jake Wires introduced himself as a member of the storage team of XenSource (now acquired by Citrix) for about five years and currently in Andrew's group at UBC, focusing on how virtualization will affect storage. Then he introduced some interesting predictions from Gartner, one being that by 2013, 40% of desktops will be virtualized. He claimed that he was not trying to argue the accuracy of these predictions but did believe that virtualization is trying to take over the world. To demonstrate this trend, he cited some companies investigating and deploying virtual desktop infrastructure (VDI). For example, UBC tried to deploy 2000 virtual desktops this year. Administrators love VDI because it makes it easy for them to manage desktops centrally while reducing hardware and maintenance costs. And users will embrace it if VDI provides a familiar working environment and comparable performance. Jake then turned to the topic of how VDI changes storage.

He introduced the typical architecture of storage in VDI. A central storage system is used to store disk images for virtual machines, and gold images are used to create a virtual disk with copy-on-write. Such architectures enable sharing of gold images and fast cloning. To facilitate system upgrades, user directories are isolated from system directories in practice. In order to identify what can be done to improve the storage system in the VDI, they did a workload analysis in UBC. They profiled 55 Windows machines for one week by installing drivers which can capture file- and block-level operations. In total, they got 75 GB of compressed logs. After that Jake showed the day-to-day activity and analyzed some peaks. An important observation is that the fraction of accesses to user data is rather small. Most of the accesses are to system directories or metadata updates. He also showed that 50% of the data was rewritten within 24 hours and that the average divergence was 1 GB through a week.

His discussion turned to improving VDI scalability. The main finding from their analysis is that the life of written data is short, and virtual machine hosts usually have local disks which are not used. By using local disks as a persistent cache, VDI can coalesce a significant portion of writes to the central storage system. They also observed that reads to system files can be shared with multiple virtual machines within a host. Implementing local persistent caching which supports both write-through and write-back policies did a good job of eliminating duplicate reads. He also introduced the idea of a multi-host preloader which, because duplicate data is shared among multiple hosts, enables shared data to be multi-casted to all interested hosts. And, finally, the differential durability mechanism enables users to specify different synchronization policies for different directories. For example, user data uses write-through to provide a strong consistency while the page file is totally eliminated from being written back to the central storage system. Jake then provided a micro-benchmark of the preloader, showing that it does eliminate duplicate reads effectively. He also showed how they evaluated their system with trace replay. Write-back caching and differential durability eliminate I/O operations significantly.

Steve Kleiman from NetApp asked whether the timestamp was updated for each write, since it would double the number of writes. Dutch Meyer, who was sitting in the auditorium, said that it was turned on. Steve Byan from NetApp asked whether they noticed any drive scan activities. Jake said that they found some and also found some defragmentation activities. Keith Smith (NetApp) encouraged people from other companies and universities to ask questions and asked why the write-back window of 10 minutes was used and whether it came from stream analysis. Jake replied that they did analyze the data trace and found 10 minutes is a sweet point to coalesce the writes. A researcher from VMware asked about the overhead of cache mechanism, especially of memory. Jake replied that their cache was located in disk, not in memory. She asked about the CPU overhead of the cache mechanism. Jake replied that their mechanism did introduce a little overhead by looking up the cache in the local disk.

Exploiting Half-Wits: Smarter Storage for Low-Power Devices

Mastooreh Salajegheh, University of Massachusetts Amherst; Yue Wang, Texas A&M University; Kevin Fu, University of Massachusetts Amherst; Anxiao (Andrew) Jiang, Texas A&M University; Erik Learned-Miller, University of Massachusetts Amherst

Mastooreh Salajegheh began by introducing small embedded devices. Although these devices are small, the market for them is huge, so it is important to solve the challenges they pose.

The flash for these devices is small and embedded in the same chip as the CPU, but since the voltage requirements for the CPU and flash are different, they should ideally use separate power lines. In actual practice, however, they share the same power line, which results in excessive energy consumption.

Mastooreh said that the goal of their research is to reduce the energy consumption for embedded storage. Traditionally, there are two ways to set the voltage for embedded systems: pick the highest voltage, which results in a huge energy consumption, or dynamically scale the power, which results in complex design. They suggested another approach—to write to flash with low voltage. However, there are occasional failures with this approach, and she gave an example. Then she introduced three factors which may influence the error rate: operating voltage level, hamming weight of data, and wear-out history. She showed their testbed and the error rates for different settings. With different operating voltages, the error rate is nearly 100% at 1.8V, but it drops sharply when the voltage changes to 1.85V. The error rate reaches 0 well before 2.2V, which is the voltage suggested for the microcontroller. She also showed that the heavier the hamming weight, the smaller the error will be, since there are fewer 1s which need to be changed to 0s.

By designing an abstract model of flash memory and introducing the accumulative behavior of the flash memory, they developed several ways to cope with partial failure. One is in-place writes which repeatedly write to the same location. The idea is to recharge the cell repeatedly until it is written correctly. She illustrated the error rates for different voltages with different number of sequential in-place writes. The effect of in-place writes did show up when the number of writes was increased. The second approach is multiple-place writes. The basic idea is to write to multiple locations so that reliability is increased. She also showed the results of this approach.

After this, she compared energy consumptions for three operations—RC5, retrieve, and store—at different voltages: 1.8V, 1.9V, 2.2V, and 3.0V. The RC5 and retrieve operations required less energy than the store operation. She hypothesized that CPU-bounded workloads, which have fewer store operations, should save energy. Then she introduced their synthetic application, which reads 256 bytes, does an aggregation of these bytes, and writes the result back. For this workload, their scheme saves as much as 34% energy. At the end of her talk, she suggested two further improvements: store the complement of a data item if it has a light hamming weight; use a mapping table to map frequently used numbers to numbers which have heavier hamming weights.

A researcher from EMC was confused by the distinction between 1.8V and 1.9V and asked why 1.8 is a useful number and not the higher 1.9. Mastooreh answered that they were trying to find the voltage that worked best and were not interested in using a voltage where an error did not happen. William Bolosky from Microsoft Research suggested that for in-place writes, it might be possible to read the data back and check to see whether it is written correctly before a second write. Mastooreh agreed and said this is what they have done and that measurements reflect the savings from this feedback system. Another researcher was curious about whether the authors had investigated the temperature factor for error rate, since high temperatures improve flash memory reliability while low temperatures result in a higher error rate. Mastooreh replied that they did consider the temperature factor and did see a decreased error rate when they increased the temperature. She admitted that they had not put the chip in a fridge, but they expected the error rate to be increased at low temperatures. A researcher from EMC asked whether it is difficult to do a read after write to verify whether the data was correctly written. Mastooreh said that it is easy for flash memory (especially the smaller ones) to check the result of write operations. This researcher pointed out that the worst bound of rewrite may take more energy than what would be saved. Mastooreh agreed and said that it depends on the voltage the chip is working on. Another researcher asked whether the authors had any ideas about predetermining the number of in-place writes for applications. Mastooreh answered that it depends on the chip, and if the application developers tell them what error rate they want, they can set parameters for them.

Consistent and Durable Data Structures for Non-Volatile Byte-Addressable Memory

Shivaram Venkataraman, HP Labs, Palo Alto, and University of Illinois at Urbana-Champaign; Niraj Tolia, Maginatics; Parthasarathy Ranganathan, HP Labs, Palo Alto; Roy H. Campbell, University of Illinois at Urbana-Champaign

Shivaram Venkataraman began with the scalability wall of DRAM, which prevents it from scaling for future computer architectures. There are several new memory technologies such as Phase Change Memory (PCM) and Memristor, non-volatile memory devices in which the access time is projected to be in 50–150 nanoseconds, which is much nearer to DRAM access time than Flash's. With non-volatile memory replacing traditional storage architectures DRAM and disk, only a single copy of data will exist in the system. They called such storage architectures single-level stores.

Shivaram demonstrated that it is not easy to maintain the consistency of a binary tree based on these new stores. To deal with such challenges, he introduced "consistent and durable data structures," defining four properties for such structures: versioning, restoration, atomic change, and no processor extensions. He discussed how they implemented a consistent and durable B-tree based on these guidelines. They chose B-tree because it is a complex data structure and is widely used in file systems. In addition to keeping a data value in a node, they introduced start and end version numbers to keep version information. Then he walked through lookup and insert/split operations and explained how version number is used and how end version is calculated.

He then explained how they incorporated their consistent and durable B-tree into the key-value store system, Redis, producing their system, Tembo. It is rather easy to integrate their consistent and durable B-tree into other systems; in their case, it only required modifying 1.7% of Redis. They evaluated their system at three levels: micro-benchmarks, end-to-end comparison, and real cluster deployment. They found that for small value size such as 256 bytes, a hashtable with logging performs better than their consistent and durable B-tree. But when the data size becomes 4K, their B-tree achieves higher throughput. For end-to-end comparison, they used the Yahoo Cloud serving benchmark, Cassandra. Tembo outperformed Cassandra in-memory by 286%. He also mentioned additional details in their paper, such as deletion algorithms and analysis for space usage.

Edward Denali (UC Berkeley) asked about the advantages of the authors' approach when compared with transaction memory. Shivaram acknowledged that transaction memory is a way to provide consistent updates but it is too heavyweight for certain applications. A researcher from NetApp asked whether the garbage collector will produce more garbage. Shivaram replied that it will not, since they can safely remove all versions before the last consistent version. Margo Seltzer (Harvard) asked how their version compared to B-trees of 30 years ago. Shivaram said they didn't want to include old data. Seltzer persisted, pointing out that multiversioning concurrency control is really a very old technique. Shivaram replied that even for a single update, they can go back and safely recover, do rollbacks. Seltzer responded that multi-level concurrency could do this 30 years ago.

Ethan L. Miller from UCSC found this work very interesting and asked whether they did an error test in the middle of the flush process. Shivaram said it is one part of their current work to improve system robustness. A researcher from Stanford asked whether their key-value system handles node failure when they compared their system with other key-value systems. Shivaram replied that they replicate data to multiple nodes, using consistent hashing, but did not perform any experiments with failures. Michael Condict from NetApp pointed out that for read-only portions of programs and memory-mapped files, there is only one copy of data either in RAM or disk. So these are also a form of single-level store and there's no need to wait another two years for new memory technologies.

Flash

Summarized by Puranjoy Bhattacharjee (puran@vt.edu)

CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives

Feng Chen, Tian Luo, and Xiaodong Zhang, The Ohio State University

Feng Chen described SSDs, discussed their merits, and highlighted the concern of limited lifespan caused by limited program/erase (P/E) cycles. He talked about the prevalence of redundancy-based solutions for wear-leveling in Flash Translation Layers (FTL), but noted that prior work has been inconclusive regarding the benefits of this approach. Chen presented a formula for endurance of a flash-based system: Endurance = (CxS)/(VxE), where C = P/E cycles, E = Efficiency of FTL designs, V = Write volume per day, and S = available space. Thus, Chen pointed out that flash endurance can be increased if V is reduced and S increased. The insight here is that data duplication is common, and deduplication can help reduce V. In addition, coalescing redundant data will help increase S, thus leading to higher endurance.

Chen talked about the challenges in implementing such a scheme, such as availability of only block-level information and lack of any file-level semantic hints for deduplication. To overcome these, CAFTL uses a hash function and fingerprinting to identify duplicate blocks. However, the authors observed that most fingerprints are not duplicates; hence they need to store the fingerprints for the ones which are most likely to be duplicates. The other challenge is for reverse mapping from PBAs to LBAs (since there is an N-to-1 correspondence between the logical and physical block addresses because of deduplication); Chen described their solution, VBA (virtual block address). He also talked about an acceleration method of sampling of pages for hashing based on the idea that if a page in a write is a duplicate page, the other pages are also likely to be duplicates. He discussed various sampling strategies and talked about their content-based scheme, where the first four bytes of a page are selected for sampling. He then talked about the evaluation based on an SSD simulator for desktop and office workloads and on TPC-H benchmark, leading to a dedup rate of 4.6% to 24% and space savings of up to 31.2%.

This was a popular talk and there were a lot of people queuing up to ask questions, but only a few got to ask because of time constraints. Assar Westerlund (Permabit) asked about the estimate of storing overhead information on the flash and

RAM. Chen replied that it is on the order of 10s of MB for 32 GB flash drives. He said that other techniques, such as the FTLs described in the next paper, could be used. In addition, he said that SSD manufacturers maintain that the most significant production cost is flash chips. So, adding more RAM could be a solution. Peter Desnoyers (Northeastern U.) talked about SANForce controllers which appear to be doing dedup similar to the CAFTL scheme. Chen said that they had not looked at it, but agreed it was a nice idea and said that runtime compression could provide more space savings, at the cost of more complexity. Raghav (Penn State) asked how old VBA values are updated when a new unique value comes in pointing at the same VBA. Chen answered that there is an offline deduplication method and it is done when the flash is idle.

Leveraging Value Locality in Optimizing NAND Flash-based SSDs

Aayush Gupta, Raghav Pisolkar, Bhuvan Urgaonkar, and Anand Sivasubramaniam, The Pennsylvania State University

Aayush Gupta presented the authors' look at different dimensions of value locality and how to develop a new kind of SSD called CA-SSD based on that. He discussed temporal and spatial localities but focused on another form of locality, called value locality, where certain content is accessed preferentially and can be exploited by data dedup using content-addressable storage. He discussed out-of-place updates and loss of sequentiality of writes, challenges which need to be overcome. However, this analysis works only if real workloads exhibit value locality. To quantify this, he talked about a new metric the authors introduced, value popularity, which indicates the potential of a workload to be value-local. They found that 50% of writes have only 8% unique values, so real workloads are indeed value local. He then described the design of their system, which added a hash co-processor to prevent that from being the bottleneck, and also added persistent storage with battery-backed RAM. Simply stated, when new content arrives, compute the hash and lookup if it is new or existing; if existing, update the data structure, otherwise write the page.

To implement this, Gupta talked about maintaining two tables, iLPT (inverse logical to physical table) and iHPT (inverse hash to physical table), in addition to LPT and HPT. He then described the metadata management policy used in CA-SSD. Since three extra tables might not fit in RAM, they exploited the property of temporal value locality. They store only a few entries in the HPT and iHPT, and discard using an LRU (least recently used) policy. This raises the possibility of affecting the dedup rate but not the correctness of the policy; in any case, experimentally they did not observe any dedup degradation. Gupta then discussed their evaluation strategy using Web workloads. He said that workload writes and garbage collection writes were both reduced, leading to improved lifetimes.

Someone asked how their work differed from least popular value replacement work published earlier. Gupta said they were not aware of this. Jonathan Amit (IBM) wanted examples of value locality in real applications. Gupta pointed out spam deliveries to all users in a mail workload. Mark Lillibridge (HP) and Gupta then discussed the different kinds of localities. Mark also pointed out the need for a distinction between static and dynamic deduplication, and Gupta agreed. Dongchul Park (U. Minn) asked about the recovery procedure when power is cut down before all the tables can be updated, and Gupta replied that they roll back.

Reliably Erasing Data from Flash-Based Solid State Drives

Michael Wei, Laura Grupp, Frederick E. Spada, and Steven Swanson, University of California, San Diego

Michael Wei discussed the reasons why SSDs pose problems for erasing data with confidence. SSDs are new devices, and the existing erasing techniques are tailored for hard disks. The presence of FTLs poses another problem, since the OS is not aware of the actual data layout on the flash drives. There is the problem of incorrect implementation of FTLs, because there are many manufacturers. Since FTLs are cheap and easier to disassemble/reassemble, Wei contended that someone could steal data overnight from an SSD. Wei then presented some background on sanitization, following which he described their custom hardware platform to validate sanitization efficacy of various techniques and reported the results. Built-in commands were unreliable, since one disk reported data as erased when it was not. The technique of crypto-scramble encrypts data online but at the same time makes drives unverifiable, so Wei said they could not vouch for this technique. Finally, with software overwrites from various government standards, Wei said that sometimes two passes of overwrites were required; at other times, even 20 passes were not enough, thus rendering this technique unreliable as well.

Wei then discussed their technique of scrubbing—an enhancement to FTL for single file sanitization. He found that flash devices can be made to program pages in order, with some restrictions for single-page overwrites to work. With multi-level cells (MLCs), however, there is a limit on the number of scrubs possible, called the scrub budget. Three modes of scrubbing are possible: immediate, background, and scan. Wei then discussed the different sanitization levels achievable with the different modes. It was found that SLC scrubbing can be done in a couple of seconds, but MLCs take longer.

Nitin Agarwal (NEC) wondered if scrubbing causes disturbances in nearby cells. Wei said that scrubbing works by writing a single page to all zeroes and that it works fine for SLCs and for some MLCs. As a follow-up, Agarwal asked whether they had any idea what a reasonable scrub budget was. Wei said that this varied from manufacturer to manufacturer, but that they had not tested how many exactly. He added that they had not seen any corruption on adjacent pages. Keith Smith (NetApp) suggested integrating their approach throughout the file system stack to avoid problems where file systems did not store new data in place of the old data, resulting in stale blocks. Wei agreed that this was one way to improve the process, but said they did not have time to modify the file system. Michael Condict (NetApp) asked if the authors had explored techniques which would not involve increasing the error rates, such as immediate block copies. Wei said this would result in too much latency. Finally, Peter Desnoyers (Northeastern U.) pointed out that hardware to steal data from flash drives is available for a thousand dollars or so. In addition, he wanted to know the block size, but Wei did not know. Peter also wondered what hardware supportsthe authors would ask for if they could and Wei had suggestions for switching mechanisms in the FTL to use the three methods of sanitization along with new command sets.

The Disk Ain't Dead

Summarized by Dutch T. Meyer (dmeyer@cs.ubc.ca)

A Scheduling Framework That Makes Any Disk Schedulers Non-Work-Conserving Solely Based on Request Characteristics

Yuehai Xu and Song Jiang, Wayne State University

A work-conserving scheduler is one in which disk requests are scheduled immediately, while non-work-conserving schedulers instead choose to delay some requests when doing so may lead to a better schedule. Despite the potential delay, the tradeoff is sometimes attractive because it can better eliminate costly seeks in magnetic disks. Current non-work-conserving schedulers, like the Linux anticipatory scheduler, take advantage of process information to predict the locality of future requests, but that information is not available in some environments. In response to this deficiency, Song Jiang presented a new scheduling framework based exclusively on observed characteristics of incoming requests.

Their approach is called stream scheduling. The authors observe that requests can be grouped into streams based on locality. A stream is defined as a sequence of requests for which the judicious decision is to wait for more requests. Streams can be identified as sequences of requests where intra-stream locality is stronger than inter-stream locality, and which grow as requests within a certain range of each other are queued. Streams are ended when the time allotted to them expires, an urgent request arrives, or the stream is broken by a request that does not exhibit locality. Song also showed the evaluation of a Linux prototype stream scheduler, which was shown to have a factor of 2 or 3 performance increase in scenarios where multiple independent workloads are run in parallel.

Arkady Kanevsky of VMware asked how well stream scheduling scaled with number of streams. Song acknowledged that the wait times could grow long but said that in such a scenario, other non-work-conserving schedulers (such as AS and CFQ) would have problems as well. Currently, there's no general solution for this problem, but Song suggested that a high QoS requirement might still be met using multiple disks, as different disk heads could then serve different streams. Vasily Tarasov from Stony Brook University asked how stream scheduling scaled to multiple queues. Song answered that stream scheduling shows an advantage as the number of clients increase relative to the number of queues. However, an implementation artifact in NFS led them to measure this effect using multiple NFS servers, as opposed to multiple threads on the same server.

Improving Throughput for Small Disk Requests with Proximal I/O

Jiri Schindler, Sandip Shete, and Keith A. Smith, NetApp, Inc.

Jiri Schindler presented his work on proximal I/O, a disk access pattern for small updates to data sets that require both good sequential read performance and frequent modification. Existing disk layouts require in-place updates for efficient sequential access, while small updates are better served by log-style appends. Unfortunately, these two layouts are at odds. Log-style appends fragment on-disk layout, which makes serial reads inefficient. This type of mixed workload is extremely challenging to system designers and occurs in a number of practical workloads, such as when data is written in small transactions, then read in bulk for business intelligence purposes.

Schindler and his team developed a new approach that uses a flash-based staging buffer for writes, which are then committed to disk in batches. The writes are placed near related data to provide the required locality for reads, and the flash-based staging area provides the sufficient I/O density to be able to retire multiple user I/Os in a single disk revolution. Proximal I/O exploits hard disk drive mechanisms that allow for scheduling multiple writes per revolution within a span of hundreds of thousands of logical blocks of the disk interface. This combination of leveraging magnetic disk behavior characteristics and a small amount of non-volatile memory performs well and is well suited to file systems with no-overwrite semantics such as the write-anywhere method used by NetApp.

Schindler's performance evaluation of proximal I/O demonstrated that the flash could be small—just 1% of the total active working set. He also showed results against an aged file system and a 90% full disk. Despite these disadvantages, proximal I/O was able to provide RAID-1–like performance on a RAID-4 disk configuration that provides parity.

Someone asked how proximal I/O compared to free block scheduling (FAST '02). In free block scheduling, updates could be made to any free block, while proximal I/O employs a minimal staging area to keep the costs of flash low. Schindler concluded that the two techniques were useful in different applications. Aayush Gupta (Penn State) asked if flash lifetimes have been considered. Schindler replied that while writes were random from the application's perspective, they may be placed anywhere on a device, so the aging of flash memory will occur evenly.

FastScale: Accelerate RAID Scaling by Minimizing Data Migration

Weimin Zheng and Guangyan Zhang, Tsinghua University

Guangyan Zhang presented this research that attempts to accelerate RAID scaling, where a new disk is added to an existing RAID set. FastScale works on RAID-0, RAID-01, RAID-10, and RAID-1. Zhang's team started with several goals. While scaling a RAID set, they needed to ensure that data was spread across the disks uniformly, so no hotspots would be created. They also sought minimal data migration, so that the scaling could be done quickly. Finally, they needed to maintain fast addressing, to not impair the overall performance. To meet these requirements, they developed a new approach to migration and on-disk layout.

To reduce the cost of migration, FastScale groups blocks that must be migrated to new disks into groups that can be moved together. This allows multiple blocks to be read with a single I/O request. Blocks are then arranged in memory to form sequential sets of blocks that can be written linearly as well. In addition, by separating the migration process into phases which copy data, and phases which replace existing data, FastScale is able to lazily update the state of migration information. This limits the number of metadata writes required to ensure consistency in the event of a failure.

With a simulator, Zhang showed an 86% shorter RAID scaling time as compared to SLAS, a scaling approach proposed in 2007. In addition, the lower overhead of FastScale was shown to result in lower latency during the block migration process.

Bill Bolosky from Microsoft Research noted that during reorganization, the block address space in the RAID appeared to be made less linear. He asked if sequential I/O performance might be harmed by the approach. The author believed that the 64K block size was large enough to avoid many seeks, but Bolosky disagreed. He suggested they try the approach with a larger block size. Keith Smith from NetApp asked why the authors had not extended the technique to work with RAID-4, which he believed was similar in difficulty to what the team already supported, unlike RAID-5. The discussion was eventually taken offline, where Zhang noted that the percentage increase of RAID-4 scaling with FastScale would be smaller, because there was a constant overhead required to compute parity information.

Wednesday Poster Session

First set of posters summarized by Vijay Chidambaram (vijayc@cs.wisc.edu)

Improving Adaptive Replacement Cache (ARC) by Reuse Distance

Woojoong Lee, Sejin Park, Baegjae Sung, and Chanik Park, Pohang University of Science and Technology, Korea

Sejin Park presented a new block replacement algorithm for second-level caches. The Adaptive Replacement Cache (ARC) algorithm dynamically balances recency and frequency. However, it doesn't take into account the reuse distance of I/O requests and hence doesn't perform well for second-level caches. The authors propose an enhancement to ARC called Reuse-Distance Aware ARC (RARC). RARC maintains a history buffer over all I/O requests and uses a sliding window on the history buffer in order to determine which I/O blocks are kept in the recency queue. The size of the sliding window is equal to the size of the recency queue in RARC.

Email-based File System for Personal Data

Jagan Srinivasan, EMC; Wei Wei, North Carolina State University; Xiaosong Ma, Oak Ridge National Laboratory and North Carolina State University; Ting Yu, North Carolina State University

Xiaosong Ma proposed engineering a highly available storage service based on email services such as Gmail, building upon the reliability of cloud storage. The authors create the abstraction of virtual storage disks on top of email accounts and employ RAID techniques such as space aggregation, data striping, and data replication. They have implemented the

Email-based File System (EMFS), which exports a subset of the POSIX interface and is built on FUSE. They evaluated its performance on the Postmark benchmark. Results show that the performance of EMFS is comparable to that of commercial solutions such as JungleDisk.

Solid State Disk (SSD) Management for Reducing Disk Energy Consumption in Video Servers

Minseok Song and Manjong Kim, Inha University, Incheon, Korea

Minseok Song proposed a scheme for using solid state disks to reduce disk energy consumption in video servers. The zip distribution of block requests allows the use of small SSDs as effective caches for the disks, thereby allowing the system to run the disks at a lower speed. This enables the video servers to reduce their power consumption without a drastic reduction in performance. This raises the question of which videos should be cached on the SSD. The authors formulate this as an integer linear problem, with the solution minimizing energy consumption. Simulation results show up to 33% reduction in power consumption when a 256 GB SSD is used.

Object-based SCM: An Efficient Interface for Storage Class Memories

Yangwook Kang, Jingpei Yang, and Ethan L. Miller, University of California, Santa Cruz

Yangwook Kang pointed out that using SCMs in the storage hierarchy today leads to the choice between major changes to the file system and suboptimal performance. The authors aim to solve this problem using an object-based model, enabling integration of a range of storage class memories into the file system and drop-in replacement of different SCMs. The object-based model also enables performance optimization for each SCM, since more information is known about the I/O requests in this model. The authors have implemented a prototype in Linux and evaluated it using the Postmark benchmark. Results show significant performance improvements when the object-based model is used.

Latent Sector Error Modeling and Detection for NAND Flash-based SSDs

Guanying Wu, Chentao Wu, and Xubin He, Virginia Commonwealth University

Guanying Wu explained that the increasing density of NAND Flash SSDs leads to higher probability of latent sector errors. Latent-sector errors in NAND Flash are not well understood, due to the limited population of SSDs in the field. The authors propose a new model for latent sector errors in NAND Flash SSDs that takes into account the write/erase cycles in NAND flash, retention, and electrical disturbances. The authors also propose an optimized disk scrubbing strategy based on their comprehensive model, which uses different policies such as localized and staggered scrubbing.

Polymorphic Mapping for Tera-scale Solid State Drives

Sungmin Park, Jaehyuk Cha, and Youjip Won, Hanyang University, Korea; Sungroh Yoon, Korea University, Korea; Jongmoo Choi, Dankook University, Korea; Sooyong Kang, Hanyang University, Korea

Sungmin Park proposed a scheme to manage the mapping information for tera-scale solid state drives. For such large drives, the size of the mapping information itself is on the order of gigabytes and new techniques to manage the mapping table in SRAM are required. The proposed mapping scheme, polymorphic mapping, exploits the spatial locality of requests to store only part of the complete mapping information in SRAM, thus reducing the space needed for the mapping. This is done at different granularities from the block to the page. The authors evaluate their system using several real-world workloads. The results show that polymorphic mapping outperforms DFTL in most cases.

DBLK: Deduplication for Primary Block Storage

Yoshihiro Tsuchiya and Takashi Watanabe, Fujitsu Limited

Yoshihiro Tsuchiya presented the Deduplication Block Device (DBLK), a primary-storage device with in-line block-level deduplication. DBLK uses an innovative multi-layer Bloom filter (MBF) in order to index the data in the system. The MBF works like a binary tree, with a Bloom filter present at each node. The system uses techniques such as bitwise transposition in order to optimize the Bloom filter access. Micro-benchmarks show that DBLK's performance is comparable to the base RAID system, and that using the MBF to index metadata leads to reduced latency.

Implementing a Key-Value Store based MapReduce Framework

Hirotaka Ogawa, Hidemoto Nakada, and Tomohiro Kudoh, AIST, Japan

Hirotaka Ogawa presented SSS, a MapReduce system based on distributed key-value stores. SSS seeks to eliminate the shuffle and sort phase of MapReduce using the properties of the key-value stores, while enabling intuitive access to the MapReduce data. Building SSS on distributed key-value stores makes Map and Reduce tasks equivalent, enabling combinations of multiple maps and reduces in an individual workload. The authors evaluated their prototype using the wordcount benchmark. They demonstrated performance comparable to Hadoop. The results also show that an optimized version of SSS is almost three times faster than Hadoop.

Hot and Cold Data Identification for Flash Memory Using Multiple Bloom Filters

Dongchul Park and David H.C. Du, University of Minnesota—Twin Cities

Dongchul Park pointed out that current schemes for identifying hot data in flash memory focus only on the frequency of data access, ignoring the recency of the data access. The authors present a new classification algorithm, window-based direct address counting (WDAC), which also takes into account the recency of data access. Multiple Bloom filters are used in order to efficiently capture recency and frequency information. The key idea is that each Bloom filter uses a different recency weight to capture fine-grained recency. Experimental results show that WDAC improves performance by 65%, with lower overhead and memory requirements.

Second set of posters summarized by Deepak Ramamurthi (scdeepak@cs.wisc.edu)

An FCoE Direct End-to-End Connectivity Scheme

Michael Ko, Li Ke, Wang Yuchen Wu Xueping, Yang Qiang, Liu Lifeng, Meng Jian, and Yang Qinqin, Huawei Symantec Technologies Co., Ltd.

Michael Ko explained how the forwarder becomes a bottleneck in a Fibre Channel over Ethernet (FCoE) network, when it is required to mediate all data and control information, including those between FCoE end nodes. The authors propose the introduction of a fabric login wherein each end node is assigned a unique Fibre Channel identifier. All data plane functions between the end nodes use this identifier to communicate directly, thereby bypassing the FCoE forwarder. This eases the load handled by the FCoE forwarder. As native FCoE devices become commonplace, the forwarder is no longer a time-critical component and can be implemented in software.

What makes a good OS page replacement scheme for Smart-Phones?

Hyojun Kim, Moonkyung Ryu, and Umakishore Ramachandr, Georgia Institute of Technology

Hyojun Kim explained the need for OS-level support for better management of low-end flash storage used in smartphones. The authors present data to show that write request ordering is a key factor in the performance of flash storage, with sequential writes offering better performance than random writes. They present two write ordering-aware page replacement algorithms—Sorted Clock and Sorted Clock-WSR (Write Sequence Reordering)—and quantitatively show that they perform better than replacement policies that do not respect write ordering.

Accelerating NFS with Server-side Copy

James Lentini, Anshul Madan, and Trond Myklebust, NetApp, Inc.

James Lentini presented a new NFS server-side file copy interface that does away with the two-step approach by first fetching the file from the server and then writing it again to the destination location on the server. This single message contains the source and destination information, which saves a lot of client-side and network resources. Applications of such a mechanism include VM backup, cloning or migration, and file backup restoration.

Cluster Storage System, RAID and Solid State Drive Emulation with David

Leo Arulraj, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Leo Arulraj presented interesting extensions to David, a lightweight storage system emulator that currently allows the emulation of large disks by much smaller disks. Incorporating complex RAID setups into the emulator and on-the-fly generation of parity is one possible extension. Extending David to emulate SSDs is not trivial, because of the difference in latencies of I/O from an SSD and a disk.

Tamias: a privacy aware distributed storage

Jean Lorchat, Cristel Pelsser, Randy Bush, and Keiichi Shima, Internet Initiative Japan, Inc.

Jean Lorchat presented a distributed storage system that enables users to share data while giving them fine-grained control over the sharing process. The authors use Tahoe-LAFS, an open source storage system, as a foundation for their system, and build in additional features such as capabilities and user identification. They achieve capabilities through public-key cryptography. They maintain a directory of public keys of friends and use this information for access control.

Mercury: Host-Side Flash Caching for the Datacenter

Steve Byan, James Lentini, Luis Pabón, Christopher Small, and Mark W. Storer, NetApp, Inc.

Steve Byan and his team propose Mercury, a system that uses flash storage as a cache in a datacenter environment. With increasing amounts of flash being integrated into shared VM servers, they can be used for storage purposes. But using them as primary storage would break the shared storage model of a datacenter. The authors have built a block-oriented write through cache with this flash storage. Their results show that this leads to a very healthy reduction in mean I/O service time as well as in the number of requests sent to the server.

DADA: Duplication Aware Disk Array

Yiying Zhang, University of Wisconsin—Madison; Vijayan Prabhakaran, Microsoft Research

Yiying Zhang presented DADA, a duplication-aware disk array which identifies and retains the duplicated blocks on primary storage systems. This additional knowledge is put to use in different ways. Disk background activities like scrubbing are made faster by scrubbing only a unique block among duplicates and skipping the rest. Recovering only unique blocks improves RAID reconstruction time and hence improves availability. She also explained some of the avenues explored in choosing a representative block among the duplicates, which included a simple random selection policy to more complex heuristics based on region density of unique blocks.

MAPL: Move-anywhere proximate layout

Peter Corbett, Rajesh Rajaraman, Jiri Schindler, Keith A. Smith, and Pelle Wåhlström, NetApp, Inc.

Keith Smith presented MAPL, a journaling file system that tries to preserve the physical sequentiality of logically related blocks of data. By achieving this, the file system is able to provide good performance even for sequential-read-after-random-writes workload. MAPL allocates regions of the disk to a file. Space is over-provisioned for a file so that a portion of the file, called the *snapshot reserve,* is used to accommodate logically overwritten blocks of data in a snapshot. Once a region fills up, the snapshot data is copied out in bulk to a different region of the disk. Good performance is achieved by a sequential read of a full region for actual random reads. The cost of the copy-out of snapshot data is also amortized.

Scaling Well

Summarized by Xing Lin (xinglin@cs.utah.edu)

The SCADS Director: Scaling a Distributed Storage System Under Stringent Performance Requirements

Beth Trushkowsky, Peter Bodík, Armando Fox, Michael J. Franklin, Michael I. Jordan, and David A. Patterson, University of California, Berkeley

Beth Trushkowsky started her talk by introducing the elasticity to scale up and down provided by clouds and turned to talk about her work on how to dynamically allocate storage resource for workload changes. The challenge for this is that they must satisfy service level objectives (SLO). Then she showed a trace of Wikipedia workload which shows that hits are rather bursty. If they were to replay it on a system with 10 storage servers, the hotspot would affect one of those 10 servers. For this hypothetical example, the system would need to be over-provisioned by 300% to handle a spike of this magnitude on any data item. So it is important to leverage the elasticity from cloud storage.

Beth then introduced SCADS, an eventually consistent key/value store that has three features: partitioning, replication, and add/remove servers. All the data is kept in memory, which avoids the latency of disk accesses. With SCADS, it seemed quite straightforward to use classical closed-loop control for elasticity. But she illustrated that closed-loop control cannot handle oscillations from a noisy signal well, and smoothing can result in a long delay to react to spikes. In order to solve these problems, the authors proposed a new control framework called model-predictive control (MPC). This framework uses per-server workload as a predictor of upper-percentage tile latency and needs a model to map the workload to SLO violations. The MPC knows the current state of the system and can determine actions to be taken to meet constraints. Beth then showed how to transition from the classic closed-loop control to the MPC model. The upper-percentage tile latency is replaced with a workload histogram, and the controller refers to the performance model to decide actions. The authors built the performance model by benchmarking SCADS servers on Amazon's EC2. Using this performance model, the MPC can decide when and where to migrate the data, but it still does not know which piece of data is hot and should be migrated. So, they introduced the idea of finer-granularity workload monitoring. The benefit from this approach is that hot data can be identified and moved back and forth much more quickly. With both the performance model and fine-grained monitoring, it can determine the amount of data to be moved, what the hot data is, and when to coalesce servers. Beth also pointed out that they used replication for prediction and robustness; she referred interested audiences to read their paper for more details.

Next she illustrated how their system decides the amount of data and the idle server to be moved. Then she introduced the experiment setup and workload profiles for Hotspot and Diurnal. The aggregate request rate appeared to be flat, but the per-bin request rate increased sharply at the beginning for the hot bins. For the other 199 bins, the per-bin request rate remained almost constant. Their system dynamically replicated hot data among more servers, which succeeded in keeping the 99% tile latency almost constant.

Fred Douglis from EMC asked whether it is possible to use past behavior to predict future workload. Beth replied that the experiment they showed today involved observing the workload and reacting directly to it but that it is possible to use yesterday's workload trace as input to the controller. Brent Welch (Panasas) asked about the cost model of moving data around the servers. Beth agreed that if moving data is too costly, you might need to trade off the decision about scaling up or scaling down. Brent continued to point out that the cost models of SLO and moving data are different and encouraged Beth to look into that model in the future. Margo Seltzer (Harvard) said that from their experience, they can see a stable behavior for minutes or hours but the variability of day-to-day behavior is huge. She asked whether the authors have noticed this phenomenon and how this variance will affect their system's behavior. Beth replied that they did several things to deal with this. Although they did not evaluate the performance on different days, they built their model on different days and their experiments ran successfully. They used replications to improve the stability of their system. Margo asked what factor of replication they used. Beth replied they used a replication factor of 2 in their result. Xiaodong Zhang from OSU said moving data is expensive and caching is more effective. Beth replied that their system keeps data in memory and functions as if it were adding another cache.

Scale and Concurrency of GIGA+: File System Directories with Millions of Files

Swapnil Patil and Garth Gibson, Carnegie Mellon University

Swapnil Patil started his presentation by noting that checkpoints in supercomputing require each thread to write states periodically into a new file. As a result, it requires huge directories. He also pointed out that future massively parallel applications and diverse small file workloads may need huge directories. File systems today provide parallel access to the file data but not to the metadata. So they decided to build a scalable and parallel file system directory subsystem. Their goal is to support high file insert rates in directories and complement existing cluster file systems by providing POSIX-like semantics. Swapnil illustrated how small and large directories are stored on three servers: the smallest directory is hosted by a single server, while the largest directory is split into three partitions based on a hash scheme, and each partition is stored at one server. Then he discussed how the GIGA+ client and server components work and interact and showed that GIGA+ can achieve 98K file creates/s, which far exceeds the HPCS goal of 32K. He also compared the scalability of GIGA+ with HBASE and Ceph, and GIGA+ has much better scalability than both of them.

With the observation that directories start small and most remain small, directories in GIGA+ start with one partition on one server. When the number of files in a directory reaches a threshold, the hash space is split into two equal halves and half of the files are moved to a second server. It continues this split repeatedly when the directory size keeps increasing. The uniqueness of the split process in GIGA+ is that splits can happen concurrently at multiple servers without any synchronization. Swapnil described the performance of GIGA+ during the incremental splitting phase as dropping once until all the servers are used, then scaling linearly. He also mentioned that interested readers should look at the paper for additional details.

Assar Westerlund (Permabit) pointed out that for checkpoints, the directories do not shrink. He asked whether their system will rebalance for shrinking. Swapnil replied that currently they do not consider shrinking, but it is possible to deal with this. Servers can shrink their directories and the client mapping will be updated lazily. Thomas Schwarz from UCSC pointed out that Ceph's performance stats cited in the paper were five years old and that Ceph has been actively developed and has many improvements; why didn't the authors compare the scalability of the latest version with GIGA+ directly? Swapnil said that by default, Ceph does not support distributed directories. They tried to run it but, due to time pressure, they could not complete their evaluation. They do believe the latest Ceph will have better scalability than the old version. A researcher said he understood the goal of this work and asked why not just ask the programmers not to create so many files in one directory. Swapnil replied that some of the applications are legacy and it is better to provide file system support for them. What will the performance be when reading or deleting all the files? Swapnil replied that doing a read of millions of files is interesting, because you will get two-day vacations. A researcher from NetApp asked whether it will cross partitions when renaming a file. Swapnil said it is possible and they can reuse the atomic renaming function provided in the cluster file systems.

AONT-RS: Blending Security and Performance in Dispersed Storage Systems

Jason K. Resch, Cleversafe, Inc.; James S. Plank, University of Tennessee

Jason Resch introduced the basic idea of dispersed storage, which is to computationally massage data into multiple pieces and store them at different locations. It allows owners to reconstruct the original data by using any K out of N pieces. K and N can be arbitrarily chosen. The reliability of a storage system is improved since only K pieces are needed to reconstruct the original data, and it allows for disaster recovery without replication. It also provides for strong security, as discussed later.

The conventional approach to storing data securely is to encrypt it, but this requires users to protect and store their keys. Another problem with this approach is that users will lose their data if they lose their keys, and increasing the reliability of keys by replication opens more opportunities for attacks. In 1979 Adi Shamir and George Blakely independently discovered a better way, called Secret Sharing, in which a secret is divided into N shares. You can get the secret with any threshold (K) number of shares. However, you cannot get any information about the secret with fewer than K shares. This provides a high degree of privacy and reliability. Actually, encryption is a specific case of secret sharing, where N = K = 2. However, Secret Sharing has several drawbacks which prevent it from being used for performance- or cost-sensitive bulk data storage. The first drawback is the storage overhead. For Shamir's scheme, it requires N times storage and bandwidth, while for Blakely's method it is even worse: N*K times. The encoding time is another drawback. It grows with N*K. To deal with these two challenges, Michael O. Rabin designed another method, called Information Dispersal Algorithm (IDA), which can achieve efficiency, security, load balancing, and fault tolerance. Storage requirements are reduced to be (N/K) times of original data and (N/K) can be chosen close to 1. However, the security of Rabin's method is not as strong as Shamir.

In 1993, Secret Sharing Made Short (SSMS) was designed by Hugo Krawczyk by combining Shamir's Secret Sharing with Rabin's IDA. It works as follows: first, the input is encrypted with a random encryption key; the encrypted result is then dispersed using Rabin's IDA, while the random key is dispersed using Shamir's Secret Sharing. The result of this combination is a computationally secure secret sharing scheme with good security and efficiency.

After this introduction to related work, Jason discussed their new scheme, called AONT-RS. It combines Ron Rivest's All-or-Nothing Transform with Systematic Reed-Solomon encoding. The security and efficiency properties are similar to SSMS. However, it has another four properties: faster encoding, protected integrity, shorter output, and simpler rebuilding. In the All-or-Nothing Transform it is trivial to reverse the transformation once one has all the output, but it is very difficult to invert without all of the output. So by combining an All-or-Nothing Transform with Reed-Solomon, a computationally secure secret sharing scheme was achieved. Jason then showed a complete view of their system. AONT is used to pre-process the data and IDA is used to create N slides. Without a threshold number of slides, it is impossible to recreate the original data. If there are some bit flips in the encrypted packets, then you will get a different hash value and a different key to decrypt the packets. You can easily tell this from the decrypted canary value.

The observed performance and the expected performance derived from their performance model for AONT-RS, Rabin, and Shamir were shown. The authors implemented two versions of AONT-RS, optimized for performance and security, by using different ciphers and hash functions. Jason described a real-world deployment of their system for the Museum of Broadcast Communications which spans three power grids.

Mark Lillibridge (HP Labs) asked why not encrypt the data and only distribute the key with secret sharing, since keys are much smaller. Jason replied that in order to provide high availability and reliability of the data, they had to ensure the security of both the keys and the data. If the data is stored in a single or unsafe storage system, it may be corrupted or deleted, leaving you unable to restore the data. Bill Bolosky (Microsoft Research) asked why they used a secure hash in their AONT and suggested CRC might be enough. Jason said that secure hash ensures the integrity of their data and agreed that if AES were used to do encryption, the CRC could be enough to do the hash. Assar Westerlund (Permabit) suggested that it might be more helpful if the authors could present their threat model more clearly and pointed out that with their scheme, it would be possible for the storage providers to get together to figure out the information. Jason acknowledged this and said they had described their threat model in their paper. Their threat model is for enterprise storage which wants to maintain control of their data and does not want to recover data from node failures. Organization managers do not need to worry about encryption keys which can be taken and used by their employees.

Making Things Right

Summarized by Dutch T. Meyer (dmeyer@cs.ubc.ca)

Emulating Goliath Storage Systems with David

Nitin Agrawal, NEC Laboratories America; Leo Arulraj, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Awarded Best Paper!

Leo Arulraj presented David, an emulator for evaluating new storage systems in this Best Paper award winner. Arulraj explained that researchers face a dilemma in trying to measure future storage systems with current hardware. David is designed around the observation that most benchmarks measure performance but require only valid, not exact, file content. The emulator works by storing metadata but throwing away and regenerating fake data content as needed. In doing so Arulraj showed how they could accurately emulate a 1 TB disk using an 80 GB disk, or a multiple disk RAID with only RAM.

A key requirement of the system was being able to accurately classify blocks as data or metadata. Two such methods were provided. For ext3, David uses implicit classification, where the stream of requests is monitored and knowledge of the file system's operation is used to make the correct classification. For Btrfs, an explicit classifier was shown, which passes hints through the file system stack. In addition to classification, David must also model device behavior. In evaluating the system, Arulraj showed how they could model both device and I/O request queue behavior. Together these techniques allow a file system evaluator to accurately emulate, and thus measure, the performance of a new file system on faster and larger hardware than they have access to. As future work, the team intends to model a storage cluster with just a few machines.

During the question period, Arulraj explained that they must exert great effort to make the correct classification. Nitin Jain from Riverbed Technology asked if file system–level caching complicates block classification decisions. Arulraj explained that no decision will be made about an unclassified block until it can be made definitively. Geoff Kuenning from Harvey Mudd asked if explicit classification would incorrectly identify data blocks written to the journal as metadata if full journaling was enabled. Arulraj explained that they would need to use knowledge of the file system's operation in addition to explicit classification in that case. Vasily Tarasov from Stony Brook University asked about positioning David lower in the stack, below the I/O request queue. Arulraj explained that being above the request queue provides an opportunity to speed up benchmarking through emulation.

Just-in-Time Analytics on Large File Systems

H. Howie Huang, Nan Zhang, and Wei Wang, George Washington University; Gautam Das, University of Texas at Arlington; Alexander S. Szalay, Johns Hopkins University

Howie Huang presented research that addresses the difficulties of quickly generating statistics about a large file system. As a motivating example, Huang imagined a border patrol agent tasked with checking travelers' laptops for pirated media. The analytics engine developed at George Washington University, called Glance, delivers approximate results to aggregate query processing and top-k style queries. This gives quick answers to questions like, "What is the total count of a specific type of document?" or "What are the top 100 files in the file system that meet some criteria?" respectively.

The system works by walking the tree structure of a file system on a random path while evaluating the query. Each random walk constitutes a trial and the expected result of each of those trials is the actual value in the file system. The problem with this approach is that high variance can result in inaccuracy. Huang elaborated on two techniques to decrease variance. In the first, his tool visits all directories high in the tree to eliminate high-level leaf nodes that might end a trial early. In the second, the random walk is expanded to include random subdirectories in a breadth-first search manner. The system was evaluated on an impressive set of traces, including file systems up to 1 billion files, assembled from a Microsoft file system study, a pair of 2 million file traces from Bell Labs, a 2.4 million file NFS trace from Harvard, and a synthetic file system generated by impressions.

The Q&A was lively. Michael Condict from NetApp noted that the accuracy of the results should be independent of file system size, as is true in calculating a margin of error of a poll. Margo Seltzer from Harvard asked how the work compared to online aggregation estimates research in the database community. Bill Bolosky from Microsoft Research noted that many distributions in storage are highly skewed; he pointed to his own work published earlier the previous day, which showed that 40% of bytes were stored in 0.02% of files. Both Bolosky and the author agreed that you would need many more samples to capture distributions such as this.

Making the Common Case the Only Case with Anticipatory Memory Allocation

Swaminathan Sundararaman, Yupu Zhang, Sriram Subramanian, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

The common paths in a file system's code are usually well tested by users and can be considered hardened against bugs. Unfortunately, rarely executed paths such as error handling routines do not see the same degree of use. When this error handling code has bugs, it can cause serious file-system and application-level errors, sometimes even silently corrupting data. Sundararaman's presentation detailed their approach, Anticipatory Memory Allocation (AMA), and how it can ensure that memory allocation errors are handled correctly inside the operating systems in the context of file system requests.

Sundararaman started by showing how more than a half-dozen Linux file systems responded to fault injection during memory allocation. Each showed some potential to arrive at an inconsistent or unusable on-disk state. Sundararaman and his colleagues proposed to use static analysis to determine the location and factors that determine the size of each memory allocation on the request path, then runtime support to preallocate this memory at the start of the request and to draw from this preallocated pool so that the memory allocation does not fail. This has the advantage of easy recovery, because early in request handling there is little or no state to unwind. It also has the effect of centralizing all of the recovery code, so failures need only be handled in one place. At two points in the talk, Sundararaman relayed the project's mantra: "The most robust recovery code is recovery code that never runs at all." Before closing, the author showed how his team had minimized the memory overheads by reusing objects in the cache and recycling memory in the preallocation pool.

Jiri Simsa from CMU asked how long the semi-automated approach to determining allocation sizes took, and how human error might be handled. Sundararaman explained that the approach took just a few seconds and could be fully automated. He also explained that it does assume that finding bounds for the memory sizes was decidable, based on the reasonable structure of file system code. Nitin Jain from Riverbed Technology thought it would be interesting to see the difference in the number of outstanding requests that can be sustained, given the memory overhead of the approach, and the author agreed. Danilo Almeida from Likewise Software asked how the tool tracks allocations to the associated memory waiting in the preallocated pool. This led to an explanation that tracking allocations happen inside each request and the called memory-allocation function and its input parameters help determine the type and size of object that needs to be returned back to the calling function.

Work-in-Progress Reports (WiPs)

First set of WiPS summarized by Avani Wildani (avani@soe.ucsc.edu)

New Cache Writeback Algorithms for Servers and Storage

Sorin Faibish, Peter Bixby, John Forecast, Philippe Armangau, and Sitaram Pawar, EMC

Sorin Faibish presented a new cache writeback algorithm for servers, storage, and file systems. Their previous work, published in SYSTOR2010, defined the algorithm and demonstrated that it provided a 30% improvement to performance on their system. The goal of this project is to adapt the algorithm to make it usable in a general environment. They want to change the paradigm of using metrics to prevent I/O bottlenecks and stoppage in systems, some ideas for which were presented at LFS 2010.

So far, they have used measurement tools to pinpoint the problem and have written a Linux utility that will be made publicly available. They have identified room for improvement in ext3, ext4, and ReiserFS. Many applications use aggressive caching, but they are slowed by flushing to disk. They would like to solve memory swap, which is a big issue for cache writeback. Their end goal is to completely move memory swap on Linux. Their results show that the cache writeback took 3200 seconds with 8 GB of memory, 4300 with 4 GB, and never finished at 2 GB of memory. This shows that their intuition is correct. They also demonstrated how their method worked in simulation.

OS Support for High-Performance NVMs using Vector Interfaces

Vijay Vasudevan and David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

Vijay Vasudevan began by describing how we have a shrinking I/O gap, and yet the methodologies we use have not caught up to the hardware. Historically, the CPU-I/O gap has increased sharply, but recently this has changed because of the move to multi-core processors and the limit of single thread execution. Additionally, NVRAM is very fast and has become widely available. Modern systems are capable of over a million random I/Os per second instead of thousands, but we still use the same technologies to support these systems! Vasudevan defines this problem as the shrinking I/O gap.

You could reduce access latency, but more and more people are increasing device parallelism. Instead, the authors suggest issuing enough concurrent I/Os to saturate devices whose queue depths are increasing over time. Vasudevan introduced a vector operating system interface where independent threads calling files are replaced by a vector of OS calls. Combining threads into a vector limits context switches, and interrupts, and even reduces the path resolution operations if the files are in the same directory. The questions that remain are, "What are the interfaces?" "What is the impact on the latency?" and "What algorithms will best maintain the queue depth?"

During Q&A someone asked, "If operations are sent simultaneously, how do you aggregate?" Vasudevan replied that this is based on the operating system.

PreFail: Programmable and Efficient Failure Testing Framework

Pallavi Joshi, Haryadi S. Gunawi, and Koushik Sen, University of California, Berkeley

Pallavi Joshi discussed failure testing in large, distributed systems composed of commodity machines. In the cloud era, these systems are prevalent and failures in these systems are both frequent and diverse. The state-of-the-art for testing for such failures is testing single failures and testing multiple random failures to see if the system correctly recovers. Tests against single failures are straightforward, requiring one test execution per failure. For multiple failures, however, there are too many potential sequences of failure to test every sequence separately, necessitating sub-sampling.

Joshi proposed improving on the random selection typically used to do this by identifying the failure paths that most need testing and avoiding retesting failures that lead to the same recovery behaviors. He outlined an example of testing points where the number of test sequences required to provide full coverage is significantly smaller than the combinatorial maximum. Developers can use this to easily identify the code snippets to test failures and define failure coverage.

Don't Thrash: How to Cache your Hash on Flash

Michael A. Bender, Stony Brook University and Tokutek, Inc.; Martin Farach-Colton, Rutgers University and Tokutek, Inc.; Rob Johnson, Stony Brook University; Bradley C. Kuszmaul, MIT and Tokutek, Inc.; Dzejla Medjedovic, Stony Brook University; Pablo Montes, Pradeep Shetty, and Richard P. Spillane, Stony Brook University

Pradeep Shetty presented an alternative to Bloom filters for efficient cache indexing. Bloom filters are common for efficient indexing but do not scale well out of RAM. Bloom filters are used in BigTable, deduplication, network computing, bioinformatics, and a variety of other applications. These organizations handle the Bloom filter scaling issue using techniques such as MapReduce, LFS/merge trees, etc.

There is an underlying issue to merging two Bloom filters. Particularly, it is difficult to recover hashes and rehash them in a larger Bloom filter. It is necessary to reduce random I/Os, but even buffered Bloom filters result in random I/Os. The solution that Shetty proposed is to use compact hashing, abandoning Bloom filters entirely. Compact hashing, a CPU-bound vs. an I/O bound technique, preserves hashes and stores them in increasing order, allowing iteration. This is defined as a "quotient filter." Similar to merging two sorted lists using merge sort, the quotient filter merges serial cache. With its lookahead array, compact hashing has great scaling properties with inserts and deletes occurring at the bandwidth of the disk. The primary negative is that compact hashing takes twice the disk space of an equivalent Bloom filter. Otherwise, compact hashing is 3000 times faster than Bloom filters, 100 times faster than buffered Bloom filters, and has a comparable lookup throughput.

A File System for Storage Class Memory

Xiaojian Wu and A.L. Narasimha Reddy, Texas A&M University

Wu discussed how to design a file system for storage class, byte-addressable memory. A new file system is needed for SCMs because current file systems are not designed for non-volatile memory devices. For one, they have a very large overhead. Typically, this overhead is written off because I/O latency is so high on standard disks. SCMs, however, have significantly faster I/O, so this overhead presents a problem. A memory-based file system will also not suffice for SCMs because memory-based file systems assume that the devices are using a page cache that will be lost on restore.

Their goal is to keep the file system as small and simple as possible to keep overhead down. They propose to do this by utilizing the memory management module in the operating system, removing the block management in traditional file systems, and simplifying the file system design by allocating space contiguously per file. Instead of building over blocks, they build over the memory manager. Their code is 2700+ lines vs. 27000+ lines for ext3, and they have a small overhead compared to memory-based file systems.

Repairing Erasure Codes

Dimitris S. Papailiopoulos and Alexandros G. Dimakis, University of Southern California

The speaker presented a novel application of a new theoretical result in information theory to better rebuild erasure-coded systems. The speaker briefly introduced erasure coding, essentially a method of calculating parity over k data blocks such that any k blocks of your data+parity set can be used to restore the original data. While any MDS code can be used, Reed-Solomon is common.

Assume all n blocks of a particular piece of data are on different nodes. On rebuild, k nodes need to be spun up to reconstruct the original data. The data from these k blocks needs to be sent to a centralized location to calculate the value for the lost disk. However, if the code used is regenerating, the speaker claimed that the rebuild could occur with (n-1)*B/(n-k)*K blocks using a matroid property. For parity blocks, they read more than they send and calculate intermediate results.

Rewriting the storage stack for Big Data

A. Byde, G. Milos, T. Moreton, A. Twigg, T. Wilkie, and J. Wilkes, Acunu

This talk was presented by the Acunu company, and it describes their motivation for rewriting the middle of the storage stack to better suit modern applications and storage hardware. The speaker began by showing an old storage stack, and then pointed out that it has not changed much in many years. While there has been innovation at both ends, representing applications and hardware, there has been little work to redefine the middle.

Applications want new stuff, but they need to work around all the assumptions in the stack because the abstraction given to them is not immediately compatible with their goals. They termed this an impedance mismatch: the abstraction is fundamentally wrong. As a result, some applications built over this outdated stack are slow, unstable, and unpredictable.

Acunu proposes rethinking the stack by throwing away the file system interface and replacing it with a shared memory ring buffer in the stack. This allows a range of improvements such as extremely fast versioning, control of the page cache, cache-oblivious algorithms, the ability to rewrite RAID, etc. Fast versioning means no B-trees, since they are slow on disk. Performance is linear at eight cores, and it is maintained for billions of inserts. They currently have a product that sits above the hardware and provides a memory monitoring stack. They will issue patches for Cassandra and Voldemort so they can run over their existing stack.

Acunu is hiring (http://www.acunu.com/jobs). They will also be in beta soon.

Second set of WiPs summarizerd by Vijay Vasudevan (vrv+usenix@cs.cmu.edu)

T2M: Converting I/O Traces to Workload Models

Vasily Tarasov, Koundinya Santhosh Kumar, and Erez Zadok, Stony Brook University; Geoff Kuenning, Harvey Mudd College

When evaluating or benchmarking systems, researchers today commonly generate a workload and monitor output parameters. But benchmarks themselves have a complex set of parameters and languages to express the workload; these benchmarks are becoming more expressive over time. Alternatively, people use I/O traces from real-world workloads, which are typically more credible. Many researchers release these real-world workloads and datasets, and as a result these trace repositories are growing larger.

The key idea of this work, presented by Vasily, is to convert traces to benchmark workload descriptions. Their system takes in traces such as system call traces, file system traces, etc., and generates an intermediate workload model, which hooks into adapters for different benchmark frameworks. These adapters take in the model and write the proper parameters for each output benchmark tool. For example, they have currently used system call traces and hooked it into Filebench.

Skyline-enabled Storage System

H. Howie Huang and Nan Zhang, George Washington University

This talk presented S3, the Skyline-enabled Storage System, designed to automatically manage heterogeneous storage systems by moving files around among devices. Devices have different characteristics (capacity, bandwidth, IOPS), and choosing the right set of files to move can be difficult. When a new large disk is added to a system, one would like to move large files. This request can be expressed as a top-k SQL-like query. If you add flash memory, you have a top-k query with a different scoring function based on the characteristics of the device. The scoring function therefore varies by device.

To efficiently execute such top-k queries, they proposed maintaining skylines. Without knowing the scoring functions, which are based on device characteristics, you can get the answer from the skyline data. Their work-in-progress includes how to identify dimensions of interest, constructing the optimal skylines, and using approximate processing to improve efficiency.

Load-Aware Replay of I/O Traces

Sankaran Sivathanu, Georgia Institute of Technology; Jinpyo Kim and Devaki Kulkarni, VMware Inc.; Ling Liu, Georgia Institute of Technology

Jinpyo began by asking, "Why do we need a load-aware trace replayer?" When traces are replayed on different systems, the characteristics of the system can show a different I/O profile, making performance debugging and analysis more difficult. Jinpyo provided an example of a database program replayed on another system, showing two very different behaviors.

They proposed using a load-based metric called "outstanding I/O requests" (OIO) to evaluate I/O load-balancing algorithms on large-scale cluster systems. Using load-based replay of original traces from a hypervisor and replaying it on a guest OS showed similar, reproducible behaviors across different systems. Their future work includes generalizing to distributed environments and improving replay accuracy.

Flash SSD Oriented IO Management for Data Intensive Applications

Yongkun Wang, Kazuo Goda, Miyuki Nakano, and Masaru Kitsuregawa, The University of Tokyo

Kazuo talked about optimizing flash I/O. One major direction for flash has been to develop log-structured or copy-on-write techniques to avoid random writes. But few have explored whether these techniques achieve the potential performance of the device. The goal of this work is to look at how to optimize I/O to achieve the device's full potential. They identified the performance gap from random writes vs. sequential writes using DB benchmarks, and they found that LFS gives improvement but does not reach the potential of pure sequential writes.

They proposed "eager scheduling" using database checkpoint cues, deferring write requests, and performing write scheduling techniques such as coalescing, LFS-like address conversion, and block alignment, all within a small buffer until the checkpoint. They demonstrated a trace driven experiment showing that even a small buffer improved performance. Simply deferring the writes without the aforementioned write scheduling techniques wasn't as helpful. Their ongoing work is focusing on how more information from the application and device specifications can improve performance.

A Novel Nested Qos Model for Efficient Resource Usage in Storage Servers

Hui Wang and Peter Varman, Rice University

Server consolidation, bursty workloads, and single-level QoS make performance guarantees hard to achieve. Capacity provisioning provided by burst rate or long-term average are both insufficient: one has low server utilization and the other provides no strong guarantees.

In response, Hui proposed a nested QoS model. Each class is characterized by a traffic envelope and response time limit. If a request falls within that class, it is provided a guarantee of that time limit. The initial results showed that a nested QoS system could reduce the provisioned capacity significantly in terms of IOPS compared to a single-level QoS model, and could provide comparable performance.

Reading from a Write Optimized Parallel File System Under Highly Concurrent Shared File Workloads

Adam Manzanares, Los Alamos National Laboratory; Milo Polte, Carnegie Mellon University; Meghan Wingate and John Bent, Los Alamos National Laboratory; Garth Gibson, Carnegie Mellon University

Adam explained that HPC apps have thousands of clients writing to a single file (an N-to-1 writing pattern), which performs poorly on parallel file systems for a number of reasons. In response, they built an interposition layer atop a parallel file system that converts N-to-1 writes to N-to-N. Doing this improved write performance significantly: an example graph showed a factor-of-50 improvement. But this made subsequent read open times much worse. On opening a file read-only, all processes individually try to reconstruct a file offset map, which significantly degrades performance because each of N clients needs to aggregate N index files, which requires N^2 work.

They explored three different possible solutions. First, they developed an aggregation technique that uses a root process to perform the reconstruction exactly once, then distributing the resulting map to all the processes. While this improves performance over the naive approach, it unnecessarily serializes the work onto one process. This work can be parallelized and distributed over many nodes, which they called the “parallel read” approach. Finally, they noted an opportunity to write the map out to the file system on file close because the data would likely be in memory already, which they called the flatten-on-close approach. In their evaluation, the flatten-on-close was actually the best for read opens, but not great for writes, so the best option for a write-optimized file system was the parallel-read approach.

Their future work focused on several research opportunities: first, a parallel file system that uses decoupled log-structured files presents a unique opportunity to reconsider the downsides of log structured file systems; second, they are investigating a scalable key-value store to hold the file offset map, a feature they think will become important in the exascale era.

OrangeFS: Advancing PVFS

Michael Moore, David Bonnie, Walt Ligon, Nicholas Mills, and Shuangyang Yang, Clemson University; Becky Ligon, Mike Marshall, Elaine Quarles, Sam Sampson, and Boyd Wilson, Omnibond Systems

Michael talked about OrangeFS (a continuation of PVFS), a parallel open-source file system aimed at obtaining high performance. Michael described several goals for OrangeFS. First, they want to move file systems to fault-acceptance rather than just fault tolerance. As systems scale, failures increase. The systems that use parallel file systems exist in environments that require high availability, need replication, etc.

Support for distributed directories: applications often put lots of files in a single directory, a challenging requirement for most parallel file systems. They are changing the granularity of their directory implementation and are using techniques from GIGA+, presented the day before at FAST ’11.

Capabilities: assumptions about trust in HPC environments have changed, so security must be improved. Their system uses a capability-based system to verify trust before performing operations.

OrangeFS clients: they are developing for Windows and have a working implementation.

Redundancy: they are duplicating objects between servers, assigning different roles for metadata and data, and working on integrating hierarchical storage, replication for performance, and off-site replication once this is done.

Flash the Second

Summarized by Rosie Wacha (rwacha@gmail.com)

Exploiting Memory Device Wear-Out Dynamics to Improve NAND Flash Memory System Performance

Yangyang Pan, Guiqiang Dong, and Tong Zhang, Rensselaer Polytechnic Institute, USA

The reliability of NAND flash memory cells degrades over time and data becomes increasingly noisy with bad data. In order to accommodate noisy media, manufacturers incorporate additional NAND flash memory cells to store error correction codes (ECC). With each program/erase cycle, charge traps are accumulated in the cells, resulting in higher error rates. Especially at the beginning of a device's lifetime, the cells storing ECCs are largely underutilized. Tong Zhang presented a mathematical channel model that treats data retention as a Gaussian process with mean and variance scaling with P/E cycling and time in a power law fashion. Using this model, he introduced two techniques that better utilize the additional flash memory cells.

The first technique improves program speed by adaptively adjusting the voltage range, dV, for each bit. In the early stages, a larger dV is used to speed up the program and introduce some additional tolerable errors. The other technique improves technology scalability by allocating some redundant cells to compensate for defective ones. Because the defect rate in different blocks will be different, individual blocks will have unique P/E limits and wear leveling algorithms will handle this issue. Trace-based simulations were done using the SSD module in DiskSim. The program speed was increased by a factor of 1.5 by varying dV throughout the device lifetime, without violating ECC redundancy. For technology scalability, 30% higher endurance was shown with differential wear leveling compared to uniform wear leveling.

Brent Welch from Panasas asked if they could simulate the reduced lifetime of the device due to writing with higher voltages. Zhang said yes, it's a mathematical model and he could modify the parameters that are based on underlying physics. Another question was whether using redundant cells earlier and with higher voltage than necessary causes additional wear on cells that you would later need for redundancy. Zhang answered that all cells are erased in every P/E cycle anyway and because the erase cycle is a stronger contributor to wear-out, the impact on the overall cycle limit is small.

FAST: Quick Application Launch on Solid-State Drives

Yongsoo Joo, Ewha Womans University; Junhee Ryu, Seoul National University; Sangsoo Park, Ewha Womans University; Kang G. Shin, Ewha Womans University and University of Michigan

Yongsoo Joo presented an application prefetcher for solid state drives (SSDs) called FAST (Fast Application STarter). The amount of time it takes for an application to become responsive upon launch is a critical indicator of user satisfaction. Hard disk drive (HDD) access latency has scaled linearly while CPU, DRAM throughput, and HDD have scaled exponentially. Several software schemes have addressed this issue, including Windows sorted prefetch, which prefetches the set of blocks historically accessed during the launch of each application. They measured a 40% improvement in application launch time using sorted prefetch. This work eliminates seek delay by moving applications to SSDs, resulting in some improvement in application launch time. However, traditional HDD optimizers based on minimizing disk head movement are not very effective on SSDs. Sorted prefetch on SSDs results in a 7% improvement in application launch time.

FAST overlaps CPU computation with SSD accesses by initiating application prefetching first and starting CPU computation when data is available. This is possible because there is determinism in the set of blocks requested by most application launches. FAST is implemented in Linux and it first uses the blktrace tool to determine the blocks necessary to the application, then replays block requests according to the observed launch sequence to generate the prefetcher, and finally executes that prefetcher simultaneously with the original application by wrapping the exec() system call. The challenge in prefetching the LBA from the inode information given by blktrace was solved by building a partial inode to LBA map for each application using a system call log. They measured a 16–46% reduction (average: 28%) in Linux application launch time, with higher reduction found for applications with a higher ratio of SSD to CPU usage and better distribution of SSD access over the launch process. This work could be ported to smartphones, with expected improvement of up to 37% based on measured cold and warm start times of 14 iPhone applications.

Assar Westerlund (Permabit) asked why not use strace to capture all the file-level I/Os such as read() system calls and replay them instead of converting the block-level I/Os from the blktrace into the file-level I/Os. Joo answered that, although it is possible, capturing all the read system calls generates an extremely huge trace size even though very few of the read() calls will actually trigger the block requests. Therefore, it is not an efficient approach in terms of the function call overhead and the prefetcher size. Ethan Miller from UCSC asked about how the cold start process was done and whether they used the application to do any work. If you use an application, it will involve a different set of blocks than if you simply open it. Joo answered that they flushed the page cache and only prefetched the blocks accessed during the measured cold start. This limits the approach in that it associates a fixed set of blocks for launch with each application.

Cost Effective Storage using Extent Based Dynamic Tiering

Jorge Guerra, Florida International University; Himabindu Pucha, Joseph Glider, and Wendy Belluomini, IBM Research Almaden; Raju Rangaswami, Florida International University

This work investigates the area of tiered storage systems using SSDs and shows that SATA and SAS drives still hold a useful place in the hierarchy. The two questions are how to choose the most cost-effective set of devices and how to best use those devices. Jorge Guerra discussed several initial approaches that motivated the design of the final data placement algorithm, called Extent-based Dynamic Tiering (EDT). EDT monitors the workload for each data extent and migrates extents in 30-minute epochs. The goal of EDT is to reduce both the initial system cost and the operational cost. SATA drives are best used for idle data, SAS drives for sequentially accessed data, and SSDs for randomly accessed data. The total utilization for a given extent on each tier is computed using the time utilization, a measure of both IOPS and access sequentiality, and the capacity utilization, a measure of the fraction of space used to store that extent. By choosing the lowest-cost tier for each extent, the tiered system response time is 43% better than an all-SAS system while using about half the power.

Another issue is how to initially configure a tiered system using this approach to minimize costs and maximize performance. Guerra discussed a configuration advisor based on four steps. The approach is to use a representative I/O trace, divide the trace into fixed-length epochs, estimate the minimum cost per epoch, and, finally, estimate the overall set of resources needed across all epochs. For each epoch, the minimum-cost hardware set is computed based on all extents accessed in that epoch. Then looking at all epochs, the maximum number of resources of each tier-type is needed for the overall configuration.

The system is implemented in Linux and evaluated with SSDs, SAS, and SATA drives and a power meter. The paper contains results for SPC1-like benchmarks as well as several Microsoft traces, but Guerra presented only the server workload results. EDT reduced the capital needed to build the system by 36% from an all-SAS solution. More importantly, the response time is 60% better than SAS and 50% better than their initial approach. Because EDT migrates data to SSD when it's most frequently accessed, power is reduced by 57%.

Fred Douglis from EMC clarified that EDT is not strictly better than the simpler IOPS-DT approach in all workloads, then asked about data migration and whether EDT prefers to not move data. Guerra answered that yes, at every epoch there are several optimizations to minimize data movement. Ajay Gulati from VMware asked if they evaluated configurations using only SSD and SATA. Guerra answered that some of the traces have highly sequential access periods and require SAS to match necessary throughput. Westerlund asked about factoring the operational cost into the cost model. Guerra showed that power is significantly reduced but hasn't yet incorporated that into the total system cost model.

Thursday Poster Session

First set of posters summarized by Shivaram Venkataraman (venkata4@illinois.edu)

A File System for Storage Class Memory

Xiaojian Wu and A.L. Narasimha Reddy, Texas A&M University

Xiaojian Wu presented a poster on designing a new file system for non-volatile memory devices. Based on the virtual memory subsystem, the poster proposed a new layout for files and directories to avoid overheads associated with repeatedly accessing page tables. When asked how this differs from existing in-memory file systems like tmpfs, Xiaojian said that existing file systems are not optimal for Storage Class Memory and that the newly proposed file system provides better performance and metadata consistency.

RAMCloud: Scalable Storage System in Memory

Nanda Kumar Jayakumar, Diego Ongaro, Stephen Rumble, Ryan Stutsman, John Ousterhout, and Mendel Rosenblum, Stanford University

Nanda Kumar Jayakumar presented RAMCloud, a cluster-wide in-memory storage system. The storage system is designed for applications which require low latency accesses and was based on a log-structured design. Updates to the system are appended to an in-memory log and these updates are then asynchronously flushed to disk to ensure reliability. Nanda explained that the design was also optimized for high throughput and acknowledged that while the cost of building such a system might be high, there were many real-world applications for which this would be affordable.

Rewriting the Storage Stack for Big Data

A. Byde, G. Milos, T. Moreton, A. Twigg, T. Wilkie, and J. Wilkes, Acunu

Andy Twigg presented a poster on changing the storage stack in operating systems to enable high performance storage systems. Andy explained that distributed storage systems like Cassandra and HBase were built on top of legacy designs like file system buffer caches. Instead, the group proposed using a new, fully persistent, versioned B-tree which would provide fast updates and also be optimized for SSDs. When asked if some of their changes would conflict with the design of Cassandra, Andy acknowledged that they had made some modifications to Cassandra and would be releasing patches for that.

DiskReduce: RAIDing the Cloud

Bin Fan, Wittawat Tantisiriroj, Lin Xiao, and Garth Gibson, Carnegie Mellon University

DiskReduce, presented by Wittawat Tantisiriroj, explored the possibility of integrating RAID with distributed file systems like HDFS. With traces from Facebook, Yahoo, and a research cluster at Carnegie Mellon University, the authors found that in most clusters, around 80% of the files were smaller than 64 MB, the block size used in HDFS. Hence they implemented RAID encoding algorithms for all the files in a particular directory and found that there was little performance degradation and no significant change in the data loss rate.

OrangeFS: Advancing PVFS

Michael Moore, David Bonnie, Walt Ligon, Nicholas Mills, and Shuangyang Yang, Clemson University; Becky Ligon, Mike Marshall, Elaine Quarles, Sam Sampson, and Boyd Wilson, Omnibond Systems

Michael Moore presented the poster on OrangeFS, which is an effort to improve PVFS, a parallel file system deployed in clusters. When asked about the similarities between metadata layout in OrangeFS and the paper on GIGA+ presented at the conference, Michael clarified that the implementation in OrangeFS was a simplified version of what was discussed in the paper. He also said that they were implementing OrangeFS to be "failure accepting," which meant that the file system was robust against events such as disk failures.

New Cache Writeback Algorithms for Servers and Storage

Sorin Faibish, Peter Bixby, John Forecast, Philippe Armangau, and Sitaram Pawar, EMC

The cache writeback algorithms used in the Linux kernel were the subject of the poster presented by Sorin Faibish. Based on multiple workloads, the poster presented evidence on how the existing cache writeback algorithms were detrimental to the performance and reliability of the system. The poster also proposed many new cache writeback algorithms; when asked which of those performed better, Sorin explained that the effort was meant to raise awareness about the problems associated with swapping and that the final solution could be any one of them.

dBug: Systematic Testing of Distributed and Multi-Threaded Systems

Jiri Simsa, Garth Gibson, and Randy Bryant, Carnegie Mellon University

Jiri Simsa presented the poster on dBug, a testing framework for distributed and multi-threaded systems. The poster described how dBug worked by intercepting system calls made by the program and had been used to find bugs in Stasis, a transactional storage system. Jiri also pointed out that these were not limited to multi-threaded programs and that dBug could also be used to find any implementation errors in protocols used in RPC-based message-passing systems.

Second set of posters summarized by Yuehai Xu (yhxu@wayne.edu)

PreFail: Programmable and Efficient Failure Testing Framework

Pallavi Joshi, Haryadi S. Gunawi, and Koushik Sen, University of California, Berkeley

Pallavi Joshi introduced a testing framework (PreFail) to address the challenges on how to systematically explore failures of large-scale distributed systems. Through PreFail, testers can easily express failure exploration polices of various complexities. Evaluation showed that the framework can produce improvements up to a factor of 21, depending on workload and failure type.

Repairing Erasure Codes

Dimitris S. Papailiopoulos and Alexandros G. Dimakis, University of Southern California

Alex Dimakis surveyed recent developments in the information theory community on designing new erasure codes that have efficient rebuild properties. He gave us an example about how to set the extra repair traffic to theoretic minimum cut-set bound when the coding was using maximum distance separable (MDS) erasure codes such as Reed-Solomon codes.

T2M: Converting I/O Traces to Workload Models

Vasily Tarasov, Koundinya Santhosh Kumar, and Erez Zadok, Stony Brook University; Geoff Kuenning, Harvey Mudd College

Traces are cumbersome to replay and do not scale well as a system becomes more complicated. Vasily Tarasov said their objectives are to investigate the possibility of automatically creating workload models from existing traces and to produce scalable and easy-to-use models while maintaining realism of the traces. Currently, three main directions are considered: the set of parameters extracted from the traces, the level at which the traces need to be generated, and the language used for expressing the traces.

Skyline-enabled Storage System

H. Howie Huang and Nan Zhang, George Washington University

Nan Zhang presented the idea of enabling efficient and automated management over a large-scale file system, in which the key prerequisite is the ability to process top-k queries in an efficient manner. They borrow the idea of the skyline operator from the database community and extend it to automated management of large file and storage systems. Their approach is to maintain a list of skyline files in the system that can support efficient processing of all top-k queries, regardless of what the scoring function might be.

Load-Aware Replay of I/O Traces

Sankaran Sivathanu, Georgia Institute of Technology; Jinpyo Kim and Devaki Kulkarni, VMware Inc.; Ling Liu, Georgia Institute of Technology

Jinpyo Kim introduced an I/O load-aware replay mechanism, so that the load profile of the original application/system can be matched with that of the replay system, even when the environments of these two systems are different. Number of pending requests is the key metric used for making the load-aware replay keep the trace performance behaviors. Evaluation showed that the objective is achieved pretty well.

A Novel Nested Qos Model for Efficient Resource Usage in Storage Servers

Hui Wang and Peter Varman, Rice University

Hui Wang proposed a nested QoS service model to provide flexible QoS performance guarantees to clients. This model places requests into different classes according to a user's SLO (service level objective) in terms of response time. A class tag is then set for each request to guide its scheduling. Evaluation showed that resources required for implementing the nested QoS model is several times smaller than that for a single-level QoS, while the service quality experienced by the clients is minimally degraded.

Reading from a Write Optimized Parallel File System Under Highly Concurrent Shared File Workloads

Adam Manzanares, Los Alamos National Laboratory; Milo Polte, Carnegie Mellon University; Meghan WingazMellon University

Adam Manzanares improved the read bandwidth of the Parallel Log-Structured File System (PLFS) by introducing several index aggregation optimizations. PLFS is a write-optimized file system that was designed for shared file workloads. PLFS transforms a logical shared file into independent log-structured components. Planned future work includes a thorough investigation into the read performance of PLFS using representative scientific workloads and the development of a scalable shared indexing mechanism for PLFS.

Analysis of Workload Behavior in Scientific and Historical Long-Term Data Repositories

Ian F. Adams, University of California, Santa Cruz; Mark W. Storer, NetApp; Ethan L. Miller, University of California, Santa Cruz

In this work, Ian F. Adams presented results of their recent study on the archival system's characteristics through analyzing access behavior data. The results are useful to guide the design of future archival systems. They discovered that storage systems appear to be becoming more disk-centric and the assumption of write-once read-maybe doesn't hold universally. Additionally, they discovered that it is extremely challenging to acquire useful datasets. To this end, data-centric tools for long-term tracing are needed.

NSDI '11: 8th USENIX Symposium on Networked Systems Design and Implementation

Boston, MA
March 30–April 1, 2011

Opening Remarks and Awards Presentation

Summarized by Rik Farrow (rik@usenix.org)

David Andersen (CMU), co-chair with Sylvia Ratnasamy (Intel Labs, Berkeley), presided over the opening of NSDI. David told us that there were 251 attendees by the time of the opening, three short of a record for NSDI; 157 papers were submitted and 27 accepted. David joked that they used a Bayesian ranking process based on keywords, and that using the word "A" in your title seemed to increase your chances of having your paper accepted. In reality, format checking and Geoff Voelker's Banal paper checker were used in a first pass, and all surviving papers received three initial reviews. By the time of the day-long program committee meeting, there were 56 papers left. In the end, there were no invited talks at NSDI '11, just paper presentation sessions.

David announced the winners of the Best Paper awards: "ServerSwitch: A Programmable and High Performance Platform for Datacenter Networks," Guohan Lu et al. (Microsoft Research Asia), and "Design, Implementation and Evaluation of Congestion Control for Multipath TCP," Damon Wischik et al. (University College London). Patrick Wendell (Princeton University) was awarded a CRA Undergraduate Research Award for outstanding research potential in CS for his work on DONAR, a system for selecting the best online replica, a service he deployed and tested. Patrick also worked at Cloudera in the summer of 2010 and become a committer for the Apache Avro system, the RPC networking layer to be used in Hadoop.

Speed, Speed, and More Speed

Summarized by Colin Scott (cs@cs.washington.edu)

SSLShader: Cheap SSL Acceleration with Commodity Processors

Keon Jang and Sangjin Han, KAIST; Seungyeop Han, University of Washington; Sue Moon and Kyoungsoo Park, KAIST

Despite the importance of secure end-to-end communication, SSL deployment on the Internet is limited due to the heavy computational overhead it places on servers and the high cost of hardware accelerators necessary for achieving decent performance in practice. Working towards a vision of widespread SSL adoption, Keon Jang presented SSLShader, a proxy designed to provide high-performance SSL acceleration in a cost-effective manner.

Similar in vein to PacketShader—previous work from KAIST—SSLShader is based on the insight that commodity GPUs offer massively parallel computation at low cost; SSLShader uses opportunistic offloading to push parallelizable computation to a GPU during periods of high load. Combining batch processing, pipelining, and adapted cryptographic algorithms, SSLShader achieves better performance for RSA, AES, and SHA1 than high-end hardware accelerators. After implementing the aforementioned optimizations, the authors found that their system's performance was bottlenecked only by data copy overhead and inefficiencies in the Linux networking stack.

Grant Tarp (ACL), Paul Newman (VMware), and Jason Li (IA) asked questions related to whether SSLShader would perform similarly in different circumstances: running alternate cipher suites, using cheaper GPU cards, or servicing multiple Web sites. Keon answered affirmatively, speculating in turn that GPUs would offer acceleration for any cipher suites that use parallelizable algorithms; lower-quality GPUs would not strongly affect performance, since data copy rates are a bottleneck; and multiple Web sites could scale if they shared the same keys. Dan Halperin (University of Washington) incisively observed that many RSA optimizations are not used in practice, since they induce greater vulnerability to side-channel attacks. Finally, Aditya Akella (University of Wisconsin—Madison) asked Keon to identify the general lessons from this work that can be extended to other intensive applications. Keon replied that opportunistic offloading and batching are helpful in minimizing latency and maximizing throughput under varying loads.

ServerSwitch: A Programmable and High Performance Platform for Datacenter Networks

Guohan Lu, Chuanxiong Guo, Yulong Li, Zhiqiang Zhou, Tong Yuan, Haitao Wu, Yongqiang Xiong, Rui Gao, and Yongguang Zhang, Microsoft Research Asia

Awarded Best Paper!

Guohan Lu presented ServerSwitch, a fully programmable and high-performance network switch for prototyping datacenter network (DCN) designs. ServerSwitch comes out of four observations: rich programmability is required for the growing number of DCN designs that depart from traditional protocol formats; commodity Ethernet switching chips are becoming increasingly programmable; PCE-I interfaces provide high throughput and low latency communication between CPUs and I/O subsystems; and commodity multicore servers offer heavyweight packet processing capabilities.

By leveraging these observations, ServerSwitch explores the design space of integrating a high-performance packet-forwarding ASIC with a powerful fully programmable server. After implementing a software stack for exposing the configurability of their switching chip, the authors provided proofs-by-example that their platform supports many DCN designs. The examples were chosen to exhibit a wide range of functionality needs, including low-latency control-plane processing, support for arbitrary header formats, and in-network packet processing.

Given that several switch designs also use the PCE-I interface to connect CPUs and programmable switching chips, Sangjin Han (KAIST) wondered how ServerSwitch differs from previous work. Guohan clarified that ServerSwitch evaluated the design on various DCN designs. Anirudh Badam (Princeton) asked whether the PCE-I would be a bottleneck if other devices concurrently performed I/O. Guohan replied that existing motherboards provide many more PCE-I lanes than required for ServerSwitch's needs. Lastly, Chong Kim (MSR) questioned why a specialized switching chip was used rather than a simple TCAM.

TritonSort: A Balanced Large-Scale Sorting System

Alexander Rasmussen, George Porter, and Michael Conley, University of California, San Diego; Harsha V. Madhyastha, University of California, Riverside; Radhika Niranjan Mysore, University of California, San Diego; Alexander Pucher, Vienna University of Technology; Amin Vahdat, University of California, San Diego

Alex Rasmussen presented TritonSort, a record-breaking large-scale sorting system. TritonSort aims to improve the efficiency of increasingly prevalent data-intensive scalable computing (DISC) systems such as MapReduce or Dryad. In addition to achieving impressive performance, TritonSort offers a number of lessons for balanced system design and scale-out architectures in general.

The authors observed that although DISC systems scale remarkably well, they are often afflicted by poor per-node performance. Consequently, the design of TritonSort was carefully defined in terms of the notion of balanced hardware/software architecture, where all resources are driven to nearly 100% utilization. With I/O-heavy workloads in mind, the team developed a staged, pipeline-oriented dataflow software system running on a static selection of hardware resources. In order to keep disks running at nearly optimal speeds, TritonSort implements an external sort that proceeds in separate routing and sorting phases. The team's delicate engineering ultimately resulted in a factor-of-six improvement in per-node efficiency over the previous sorting record holder.

Alex's talk encouraged a lively question-and-answer session. Siddartha Sen (Princeton) asked how the system balances system parameters both at the onset and in real time. Alex explained that both of these opportunities for improvement are currently being investigated. Anirudh Badam (Princeton) wondered how much DRAM could be decreased without sacrificing balance. Alex noted that memory size determines average write sizes, affecting throughput and variance. Steve Hand (Cambridge) inquired about the energy-efficiency of TritonSort. In response, Alex cited a previous sort competition where TritonSort performed very well in terms of energy efficiency, largely due to its sheer speed. Mike Freedman (Princeton) pointed out that modern disks exhibit high variance in disk throughput and wanted to know the extent of TritonSort reliance on homogeneous disk characteristics. Alex acknowledged that the disks used in TritonSort were fairly homogeneous, but noted that one could profile disks beforehand to handle heterogeneous disk performance.

Arkady Kanevsky (VMware) wondered how the mechanisms in TritonSort, which have been highly optimized for sorting, could be pieced together to play a role in a more general high-performance processing system. Alex noted that a version of MapReduce based on TritonSort has already been built. He also identified the logical disk distributor and the buffer manager as components that are widely applicable. Finally, Matai Zaharia (Berkeley) asked whether performance would be impacted if the system allowed disks to be written to and read from simultaneously. Although Alex couldn't provide any measurement results, he conjectured that the predictability of disk seeks would be negatively impacted by loosening the constraint.

Performance Diagnosis

Summarized by Andrew Ferguson (adf@cs.brown.edu)

Diagnosing Performance Changes by Comparing Request Flows

Raja R. Sambasivan, Carnegie Mellon University; Alice X. Zheng, Microsoft Research; Michael De Rosa, Google; Elie Krevat, Spencer Whitman, Michael Stroucken, William Wang, Lianghong Xu, and Gregory R. Ganger, Carnegie Mellon University

Raja Sambasivan presented Spectroscope, a request flow-based debugging tool for distributed systems. Spectroscope determines the cause of performance degradation by identifying changes in the timing and structure of request flows. The tool uses several heuristics to pinpoint persistent mutations, link them to their precursors, and rank them according to their overall performance impact.

Spectroscope begins with request-flow graphs built using end-to-end tracing tools such as Magpie (OSDI '04), X-Trace (NSDI '07), or Google's Dapper. The flow graphs are then binned into categories according to path structure and expected timing. This categorization step seeks to minimize intra-category variance and any difficulties in this process help to identify aspects of the distributed system which increase the variance of user-visible response time. To determine which categories actually contain mutations, Spectroscope employs hypothesis testing to compare each category's distribution of response times. After two distributions with a mutation are identified, Spectroscope iterates over the edges in the request flow to localize the mutation to a particular RPC or function call.

After the request flow mutations are identified, Spectroscope presents a user interface which assists the developer in investigating the problems. The interface ranks the mutations with a simple "greatest impact" heuristic: the number of requests affected by the mutation, multiplied by the slowdown in response time. The goal of this ranking is to direct the programmer's attention to the underlying cause, as there may appear to be more than one problem in the system, or one problem may yield many mutations.

Sambasivan presented the results of two evaluation case studies. In the first study, Spectroscope helped resolve four previously unsolved performance problems in a distributed file system, Ursa Minor (FAST '05). In the second, the tool was used to diagnose the cause of different benchmark results recorded by a new Google distributed system when tested in multiple datacenters. Spectroscope was able to identify the slow datacenter's communal BigTable instance as the culprit, acquitting the developers of the new system.

Rodrigo Fonseca (Brown University, session chair) asked if it is possible to identify problems using the system, rather than simply searching for causes of known problems. Raja answered that problems are identified by comparing recent paths against historical data, comparing the performance with strict SLAs, and relying on gut intuition.

Profiling Network Performance for Multi-tier Datacenter Applications

Minlan Yu, Princeton University; Albert Greenberg and Dave Maltz, Microsoft; Jennifer Rexford, Princeton University; Lihua Yuan, Srikanth Kandula, and Changhoon Kim, Microsoft

Today's cloud applications make extensive use of the network. With the growth of abstractions for network programming, not all application developers fully understand the network stack, particularly more complicated topics such as congestion control, delayed ACKs, and the need for Nagle's algorithm. This lack of understanding can lead to performance problems, which can be diagnosed after the fact by extensive logging. However, application logs are too specific to identify general network problems, switch-based logging is generally too coarse-grained, and network-wide packet sniffers can be very expensive.

Minlan Yu presented a system called SNAP, a Scalable Network-Application Profiler, which runs continuously on the end hosts in a datacenter. Yu argued that TCP stacks already gather many aspects of network-application interactions and is thus an appropriate layer at which to do diagnostic logging. For example, the flow-control algorithms rely upon how much data applications want to read and write, and the congestion-control algorithms build up on measurements of network delay and congestion. SNAP works by gathering the TCP-level statistics described by RFC 4898 and collected by many operating systems, such as number of bytes in the send buffer, congestion window size, number of dropped packets, etc. Yu noted that the CPU overhead of collecting these statistics for 1,000 connections every 500 ms was only 5%. Each end-host then transmits the collected statistics to a central server which combines the statistics with information about the network topology and the mapping of connections to applications to diagnose problems such as excessive retransmissions or undersized send buffers.

Yu presented results for running SNAP for a week in a datacenter with 8,000 machines running 700 applications. SNAP was able to identify TCP-level problems with 159 applications, including six which suffered from TCP incast. In one particularly egregious case, a record-processing application was able to process 1000 records per second if there were an even number of packets per record, but only five records per second if there was an odd number of packets per record. By using SNAP, the datacenter operators were able to tune the performance of the TCP stack.

During the question period, Yu was asked about SNAP's correlation model. SNAP assumes linear correlation across connections or a set of connections, which might be oversimplified. She was also asked about SNAP's application to non-TCP protocols, and responded that it could be extended to other protocols, but TCP was a good candidate because the gathering of statistics was already well-defined. Elias Weingärtner (RWTH Aachen University) asked if they could map TCP data to different applications running simultaneously on the same host. Yu said that they map data to each connection, so they can link it to the correct application.

Nothing but Net

Summarized by Brent Stephens (brents@rice.edu)

Efficiently Measuring Bandwidth at All Time Scales

Frank Uyeda, University of California, San Diego; Luca Foschini, University of California, Santa Barbara; Fred Baker, Cisco; Subhash Suri, University of California, Santa Barbara; George Varghese, University of California, San Diego

Frank Uyeda began this talk by describing the current state of datacenters. Big datacenters have thousands of hosts, a multitude of applications, and high-speed networks. Debugging and tuning network performance in a datacenter requires fine-grained information on network latency and bandwidth. This work focuses on how to identify short-lived bursts in a datacenter with low monitoring overhead.

Bursts can be identified by sampling utilization at fixed intervals. However, correctly determining the time scale is difficult, and a full network trace is needed in order to resample utilization at distinct time scales. The existing tools for sampling are not well suited for the task. Both tcpdump and NetFlow provide too much data, sampled NetFlow has the potential for false positives, and SNMP counters provide insufficient information and time resolution.

This work presents two techniques for supporting bandwidth reports at any time scale that can be generated after the fact without requiring a full packet trace: Exponential Bucketing (EXPB) and Dynamic Bucket Merge (DBM). These techniques scale down to microseconds, require orders of magnitude less storage than a full packet trace, and provide complex stats and visualizations. These techniques involve sampling at all end hosts in the network at the smallest time scale. EXPB allows for querying the max and standard deviation of bytes per interval. This is accomplished by keeping packet statistics in buckets growing in time by powers of two. The query is served using the closest computed time scale.

A drawback of EXPB is that it does not retain time domain information, which is required for median and percentile information. DBM performs time-series summarization, which supports queries regarding median, percentiles, and visualization of bandwidth over time. When DBM has more samples than buckets, buckets are merged together. Buckets may be merged by smallest byte count (DBM-mm) or smallest variance difference (DBM-mv) or by minimizing the difference between the highest and lowest samples (DBM-mr). DBM has a tunable tradeoff between accuracy and storage.

EXPB and DBM were evaluated by replaying 500 GB of traffic traces obtained from the TritonSort project. The base sampling time was 100 microseconds, and queries were performed at 52 time scales. Both EXPB and DBM require orders of magnitude less memory than storing packet traces, which use six bytes per packet. DBM-mr was found to be the most accurate variant of DBM and was best for time scales greater than 2 ms, with EXPB being better for time scales less than 2 ms. Both techniques can be combined for the best results. Future work includes performance culprit identification.

Wyatt Lloyd from Princeton asked if it is possible to collect stats in an event-driven manner instead of time-driven. Uyeda replied that he is not sure about the best sampling interval. Wyatt then asked if there are basic principles for other basic monitoring tasks and if Uyeda will provide a software package. Uyeda replied that this work is not fundamental to bandwidth. Any time series of data can use these techniques. Other folks have been working with latency, and this approach is complementary. As far as providing a package, they'd love to release that but are not on track to do that right now. Cheng Huang from MSR asked for any thoughts on what stats they can't capture with this technique; he feels that some are missing. Uyeda replied that there are some things you can't keep with EXPB, but DBM can be used to sample many. The question is how much error is induced.

ETTM: A Scalable Fault Tolerant Network Manager

Colin Dixon, Hardeep Uppal, Vjekoslav Brajkovic, Dane Brandon, Thomas Anderson, and Arvind Krishnamurthy, University of Washington

Colin Dixon explained that before he was a graduate student, he was an enterprise network administrator; the things that he wants from a network include a NAT, firewall, traffic prioritization, Web cache, VPN, and IDS/DPI system. These things require several proprietary middleboxes, but this solution is inconsistent and not scalable. For example, intrusion detection systems are deployed at the edge of the network. This makes it difficult to catch internal threats that do not cross the network edge. Increasing the deployment can fix this but requires coordination between the intrusion detection systems. Their work proposes a different approach: the end to the middle (ETTM). The idea is that we do not need physically centralized hardware to make logically centralized decisions. Instead, we can implement virtual middleboxes. Intrusion detection on ETTM gives pervasive monitoring that is not possible at the network edge.

ETTM makes many assumptions. ETTM is meant to be deployed on enterprise networks under a single administrative domain. Hosts must have trusted computing software, be multicore, and be virtualized. The switches in the network are required to provide ACL, such as can be provided by OpenFlow or 802.1X.

The goal of ETTM is to be able to extend network functionality in a pervasive, consistent, fault-tolerant way that automatically scales with network demand. The challenges with accomplishing this goal are being able to trust commodity PCs, performing consensus on commodity PCs, and allowing administrators to deploy new features. The PCs can be trusted by adding a trusted platform module to the hypervisor, extending 802.1X to use the TPM instead of keys, and creating an attested execution environment (AEE) to run the network tasks. Consistent decisions are accomplished with Paxos, and the AEE is exposed to allow developers to deploy services.

A micro-virtual router that invokes filters on packets is run in the AEE. The virtual router has two implementations: one in Open vSwitch that can perform line-speed packet header operations, and the other in iptables, which is slow but can perform general operations.

ETTM was used to implement a NAT, DPI/IDS, Web cache, traffic prioritization, worm scan detection, and firewall. Most features could be implemented in only 100s of lines of code. Consensus with Paxos can be performed with a group size of 20 in under 1 ms on a wired LAN, and 1700–8000 updates can occur per second. The NAT application is able to saturate a gigabit Ethernet link.

How does ETTM affect computing power? Enterprise networks have low bandwidth requirements, so ETTM should scale down well. Brad Karp from University College London asked about privacy and whether hooking an Ethernet hub up to the network allows you to snoop packets. Dixon replied that you would want something like WPA2 encryption for wired networks. Ashok Anand from the University of Wisconsin—Madison asked if ETTM could be simplified if the network was programmed instead of the end-hosts. Dixon replied that there is a whole bunch of work that does this. This work mostly focuses on path selection rather than the complete suite of things you want on your network. Everything they've done is related, and he intentionally stayed away from programming the network. How does mobility change things? If all the hosts on the network are mobile, things change, but if it's a small set, then you could just tie the mobile hosts to the local AEE. Wojciech Golab from HP Labs asked why unsafe progress is allowed in Paxos in the case of a large-scale failure. Dixon replied that in general operation of the network he expects that this would never occur, and unsafe progress is allowed because two safe lists are easier to merge than a jumble of information.

Design, Implementation and Evaluation of Congestion Control for Multipath TCP

Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley, University College London

Awarded Best Paper!

Damon Wischik started his talk with some ancient history. Circuit switching used dedicated circuits for each flow, but this wasted excess capacity. Packet switching was introduced to allowed flows to utilize excess capacity to adapt to traffic surges. Multipath is packet switching 2.0. Packet switching allows for flows to utilize excess capacity on a single link, and multipath allows for a single flow to utilize excess capacity across multiple links. In circuit switching, flows cannot harm each other. To remove a circuit, you need a transport control protocol. If you want to aggregate links, then you need a new transport protocol to share the aggregate. MPTCP is a control plane for a collection of links that makes them behave like a single large pool of capacity. MPTCP is run at both the sender and receiver as a replacement for TCP. In this work, they formulated design goals and test scenarios and implemented a MPTCP that achieves these goals. There are further questions to be addressed with multipath TCP. How much of the Internet can be pooled, and what does this mean for network operators? How should we fit multipath congestion control to CompoundTCP or CubicTCP?

There are five other design goals for this work. Number 1 is that MPTCP should be fair to regular TCP at shared bottlenecks, and design goal 2 is that MPCP should use efficient paths. If each MPTCP sent its traffic on its least congested paths, this would be the most efficient allocation with respect to paths. Design goal 3 is that the throughput of MPTCP should be fair compared to TCP. In the situation where a user has a 3G network and WiFi access, design goal 2 could be satisfied by only sending on the 3G network, but 3G has a high RTT, which would give lower throughput. Design goal 3a is that MPTCP users should get at least as much throughput as a single-path TCP would on the best of the available paths. Design goal 3b is that MPTCP should take no more capacity on any link than a single TCP flow. At this point, design goal 1 has been subsumed by goals 2 and 3. Design goal 4 is to adapt quickly, and design goal 5 is not to oscillate.

MPTCP is based on TCP, so it is important to understand the basics of TCP. TCP maintains a congestion window which additively increases on every packet received and multiplicatively decreases on loss. MPTCP maintains a congestion window w for each path. For route r, w_r increases for each ACK, and w_r decreases for each drop by $w_r/2$. Specifically, on every ACK on subflow r, for each subset $S \subseteq R$ that includes path r compute:

$$\frac{\max_{s \in S} w_s / RTT_s{}^2}{(\Sigma_{s \in S} w_s / RTT_s)^2}$$

then find the minimum over all such S, and increase w_r by that much.

Michael Freedman from Princeton asked about the number of RTTs that short-lived flows take to converge compared to TCP. Wischik replied that MPTCP will converge very quickly to the correct bandwidth and use more RTTs to balance. Guohui Wang from Rice University asked how the end host knows how many paths there are in the network. Wischik replied that MPTCP assumes a different address for each path, but if you have ECMP in the network, you can open up multiple flows on different ports and hope to get different paths. The last question was what multiplexing 3G and WiFi in a very mobile network would look like. Wischik replied that in the simple case with 3G plus WiFi and WiFi disappears, it ought to converge in a few RTTs.

Data-Intensive Computing

Summarized by Wolfgang Richter (wolf@cs.cmu.edu)

Ciel: A Universal Execution Engine for Distributed Data-Flow Computing

Derek G. Murray, Malte Schwarzkopf, Christopher Smowton, Steven Smith, Anil Madhavapeddy, and Steven Hand, University of Cambridge Computer Laboratory

The goal of CIEL, a distributed execution engine, is to make distributed programming easier. CIEL's advantage over MapReduce and Dryad is that it can express unbounded iterative algorithms within the framework rather than breaking them up into multiple jobs, making it Turing-complete. CIEL is more efficient at representing such algorithms than MapReduce and Dryad—representing algorithms as a single job cuts down on per-job overhead, and the scheduler can use information from previous iterations to improve the schedule in later iterations.

CIEL creates Turing-complete expressiveness by providing a universal execution model which allows dynamic changes to the task graph. CIEL expresses a strict superset of the possible graphs and algorithms possible in MapReduce and Dryad. CIEL maintains tables of finished results ("complete" parts) and results that are unfinished ("future" parts). Tasks are able to spawn subtasks and delegate outputs. This is similar to how modern processor architecture is designed: pipelining and aliasing as much as possible.

CIEL still provides a single master model with multiple worker nodes; however, it has built-in support to have multiple hot-swappable masters running at the same time. CIEL also provides the "Skywriting" scripting language for rapidly producing CIEL distributed programs; the standard library includes an implementation of MapReduce. Skywriting is an interpreted, dynamically typed C-like language with runtime state stored as CIEL objects (in the CIEL tables). All of CIEL, including Skywriting, is written in 9,000 lines of Python code and is available online: http://www.cl.cam.ac.uk/netos/ciel/. There is also a *;login:* article about CIEL in the April 2011 issue.

Anirudh Badam (Princeton) opened the discussion by asking about deadlocks. Derek said that they used `select` rather than deadlock control. Someone else asked Derek to comment on optimizations that their model supports. Derek said that they took a different approach. Hadoop cannot support directed graphs. Their streaming works at low bandwidth but cannot support high flows. Something they did was take advantage of local data. They also support speculative execution. Theo Benson (Wisconsin—Madison) asked if their language makes it easier to deal with outliers, and Derek responded that their long-term strategy is to try speculative execution and shoot down the slower versions. Currently, they don't have any straggler detection, as they had no stragglers. Dave Maltz (Microsoft Research) asked if extra context switching resulted in having to move more data, and Derek said that they hadn't run into this in their experiments. In a k-means problem, that cost would be relatively minor.

A Semantic Framework for Data Analysis in Networked Systems

Arun Viswanathan, University of Southern California Information Sciences Institute; Alefiya Hussain, University of Southern California Information Sciences Institute and Sparta Inc.; Jelena Mirkovic, University of Southern California Information Sciences Institute; Stephen Schwab, Sparta Inc.; John Wroclawski, University of Southern California Information Sciences Institute

Current approaches to diagnosing networking problems in large systems require large amounts of domain-specific knowledge. For example, experts analyze packet dumps, race conditions in distributed software, etc.—there are too many random patterns for simple regular expressions to highlight issues. The semantic approach advocates creating a knowledge base consisting of abstract models of understanding the system extracted from experts which allow queries from users for problem diagnosis. This is a logic-based approach.

Arun Viswanathan described the key differences from prior logic-based approaches as the composibility of abstractions for expressiveness and expressive relationships for networks. They define "behaviors" as a sequence or group of one or more related facts which can relate to other behaviors. Behaviors capture the semantics of a system and are thus closer to users' level of understanding of the system. The key to capturing interesting behaviors lies in the variety of relationships that can be expressed. There are four broad categories of relationships relevant to networked systems that the modeling language can express: (1) temporal—causal etc., (2) concurrent—overlaps, (3) logical—combinations, and (4) dependency—between data attributes. Thus, encoding a model consists of (1) capturing simple behaviors, (2) relating simple behaviors to capture complexity, and (3) defining a behavioral model from which answers—facts satisfying the model—are obtained. More information on the semantic analysis framework can be found at http://thirdeye.isi.deterlab.net.

Dave Maltz pointed out that they had taken a top-down approach, and wondered if there was a bottom-up way of bridging prior approaches and theirs. Arun answered that they wanted to make sense of lots of data, and by encoding the knowledge they allow data mining to work. They can extract models that are much more meaningful instead of packets and values. Jason Li (Intelligent Automation, Inc.) asked what Arun meant by composibility here; does the model come from a human expert? Arun replied that composibility is completely manual for now, and that models do come from human experts.

Paxos Replicated State Machines as the Basis of a High-Performance Data Store

William J. Bolosky, Microsoft Research; Dexter Bradshaw, Randolph B. Haagens, Norbert P. Kusters, and Peng Li, Microsoft

Bill Bolosky explained that they wanted to build a high-performance data store from commodity parts. The surprising finding was that compromising on consistency or any of the guarantees of Paxos was not needed for efficiency. The building block of this distributed system is Paxos replicated state machines. Replicated state machines copy state and inputs across a distributed system to maintain consistency at each node. The consistency guarantee they are able to provide is that a reader sees the *latest* commit from *any* client.

There are three key observations the authors made while designing and implementing their data store. The first key observation was that they need not apply operations in order/one-at-a-time to their replicated state machines. This is similar to how an out-of-order processor works. The second key observation was that using a writeback cache prevents slow synchronous writes to disk—again, similar to innovations in processor design. The third key observation was that batching significantly increases throughput in their system, along with not sending writes to reading disks and vice-versa.

They implemented a distributed file system represented as the Gaios virtual disk in a Windows client. NTFS is layered on top of this virtual disk. Thus, access is completely transparent to the end user.

Steven Hand (Cambridge) asked if they did measurements in the presence of failure, and Bill responded that they killed a process on one of the nodes and got a glitch while the leader fails-over. This results in a bit of slowdown, but scaling for writes is pretty flat. Colin Scott (U. of Washington) asked about Paxos group management, and Bill said they referenced prior work. Someone asked if there was an open source version and Bill said no. Another person asked about logging operations in memory, and Bill said that you can write logs to disk (slow), write to flash, or just have a system that forgets when the power goes off. Failing in Paxos means forgetting your state. But if you are willing to count a crashed system as a Paxos failure, it would probably recover in 400 μs, and could be faster if the network was better tuned. Dave Maltz asked if you could have very large Paxos groups, and Bill said you could, although they would eat bandwidth.

Security and Privacy

Summarized by Hendrik vom Lehn (vomlehn@cs.rwth-aachen.de)

Bootstrapping Accountability in the Internet We Have

Ang Li, Xin Liu, and Xiaowei Yang, Duke University

Xiaowei Yang started by giving examples of how the Internet is vulnerable to traffic hijacking over IP prefixes or denial of service (DoS) attacks. These attacks are often disruptive and can be costly. Yang wondered whether it is possible to secure the Internet with low-cost and adoptable changes. It is important that such changes be gradually deployable and benefit early adopters. There are already several proposals to secure routing and mitigate DoS attacks. These approaches, however, miss a lightweight bootstrapping mechanism that securely binds an IP prefix to its owner's keys.

IP prefixes and AS numbers, the currently used identifiers, can both be spoofed. Yang wants to counteract this problem with a system called IP made accountable (IPA). The approach used in IPA is to derive the AS number of a network from the hash value of the corresponding public key and use DNSSEC and BGP as a public key infrastructure to bind an IP prefix to its owner's keys. By utilizing DNSSEC and BGP as public key infrastructure, no additional new infrastructure is needed. Reverse DNSSEC records are used as IP prefix ownership certificates. Additional advantages of this approach are that DNS and IP addresses have the same root of trust and that DNS is a very scalable system and can be used to support certification revocations and key rollovers.

In order to speed up the certificate validation of incoming routing announcements, Yang proposes including the corresponding certificates in an optional BGP field. As this adds overhead, IPA uses a caching mechanism such that a BGP router only sends the certificates it has not sent to a peer before in its BGP messages. The resulting system can be used for both origin authentication and path authentication. For functionality such as certificate revocation and key management, Yang mentioned several solutions and suggested that the audience refer to the paper for more details.

The goals of IPA were to make it a lightweight, secure, and adoptable system. Out of the evaluation results included in their paper, Yang showed that IPA introduces only moderate BGP overhead and can be easily deployed.

Rik Farrow raised the point that one of the difficulties with approaches like S-BGP is that signature checking causes too much overhead for the routers. Yang responded that this was indeed a problem for older routers, but that modern routers should be able to handle the additional load. Dave Oran of Cisco asked whether it would not be easier if RPKI databases were copied and checked in a back-end server once a BGP update arrives. Yang answered that this potentially introduces a dependency loop between routing and certificate distribution: before a router can reach the back-end server to obtain necessary certificates to secure routing, the route to the server must be secured first. In contrast, they try to bootstrap their own certificate distribution infrastructure without relying on other infrastructures.

Privad: Practical Privacy in Online Advertising

Saikat Guha, Microsoft Research India; Bin Cheng and Paul Francis, MPI-SWS

Saikat Guha began by discussing the drawbacks of current advertisement systems for the Web. Instead of trying to improve the quality of advertisements, emphasis is placed on producing large quantities of ads, and this slows systems. To get to know more about users, advertisement brokers set cookies on the users' computers, which allows them to obtain the user's browsing history of the Web sites that contain their advertisements. Thus, targeted ads have two sides: they make money, but they invade privacy as well.

In response, the authors built a system, Privad, that tackles the privacy problem, but nevertheless supports targeted advertisements. In order to be successful, the new system should support as much of today's model as possible, should just be private enough, should not trade off privacy for low-quality ads, and should be scalable. The underlying idea of the system they have built is that the best place for privacy is one's own computer. The system therefore makes use of a client software that runs in a sandboxed environment on the user's computer. It profiles the user, serves advertisements, and sends reports.

To preserve the user's privacy, the actual advertisements are downloaded to the user's computer in larger amounts than required. Furthermore, reports are not sent directly to a broker. Instead, a third party (called a dealer) serves as an anonymization proxy. Since the reports are encrypted with the broker's key, the dealer does not know the specific advertisement the client clicked on. The broker, on the other hand, does not know from which user the reports are. Through a report ID which the dealer inserts into each report, the broker can notify the dealer to block a user in case of clickfraud.

Someone asked about the security of information on the user's computer. Guha explained that they do not attempt to solve this problem, but that the same problem exists for all kinds of other things (e.g., cached credit card information) as well. Aleksandar Kuzmanovic (Northwestern) asked about the compatibility with current models. Guha said their system supports cost per click, cost per impression, and cost per action. What are the incentives to install this kind of software if one does not want advertising? Guha said that this kind of software is definitely not for people who use Adblock, but that there are actually more people who install useless toolbars than who install Adblock. Kuzmanovic pointed out that people don't like advertising, and Guha replied that 12% of people use some form of ad blocking, while 21% will install anything. Wolf Richter of CMU wondered why he should feel safe installing massive Microsoft spyware on his computers. Guha replied that this is a blackbox broker that collects and holds privacy information, and it would be provided by someone else, like your AV provider. Guha suggested that people likely trusted their AV provider.

Bazaar: Strengthening User Reputations in Online Marketplaces

Ansley Post, MPI-SWS and Rice University; Vijit Shah and Alan Mislove, Northeastern University

Fraud is a big problem for many Web sites that connect buyers and sellers. These Web sites are based on identities and reputations that show up in a feedback profile. A problem, however, is that accounts are free and thus allow fraud through newly created identities. Each of the otherwise very successful Web sites has its own solution to prevent such types of fraud. These solutions range from account creation being made difficult, to in-person transactions and insurance services. All of these solutions, however, also come with drawbacks such as limited applicability or additional costs.

Alan Mislove presented a new approach, Bazaar, to tackle this problem. Bazaar works in conjunction with existing marketplace systems and is based on the insight that successful transactions represent a shared risk and thereby provide a bound on possible fraud. The core of Bazaar is a risk network in which identities are nodes and links represent the amount of successful transactions. The max-flow between buyer and seller is used as a metric to evaluate the risk of new transactions.

Mislove explained in more detail how the links are altered and why the max-flow is robust against known kinds of fraud. He then discussed how they dealt with certain challenges such as the delay of feedback, the inclusion of new users, and a scalable max-flow implementation.

To evaluate the system, they crawled the feedback on 8 million transactions on ebay.co.uk and used Bazaar to calculate a risk network based on the acquired data. Mislove explained that in this experiment it took only 1 to 6 seconds to check for fraud and that they obtained a false positive rate of 1% to 5% for the transaction history of only 90 days. He then concluded the talk by saying that, for this data, the use of Bazaar could have prevented fraud for an amount totaling $269,000.

What if the links between buyer and seller are not sufficient for a transaction? That kind of problem did not show up in the eBay graph, but one possible solution would be the use of an escrow service. Would this kind of system be an incentive to use compromised accounts to perform fraudulent transactions? That was possible, but Bazaar puts a bound on the amount of money that can be used.

Energy and Storage

Summarized by Brent Stephens (brents@rice.edu)

Dewdrop: An Energy-Aware Runtime for Computational RFID

Michael Buettner, University of Washington; Benjamin Greenstein, Intel Labs Seattle; David Wetherall, University of Washington and Intel Labs Seattle

Michael Buettner thinks that activity recognition for elder care is important. The goal is to track what and how objects are used to determine activities. One existing solution is to use cameras, which has privacy drawbacks. Another solution is to use "Mote"-based sensor networks, which detect object use from accelerometers, but battery life, size, and cost limit the deployment of these sensor networks. Buettner's proposal is to use computational RFID, where an RFID reader sends power and commands to CRFID tags. Battery-free CRFID tags use radio signals to compute, sense, and communicate. Dewdrop is a runtime for CRFIDs. This enables CRFID tags to use scarce energy to run programs, which have varied and non-deterministic energy needs and whose input power can vary by two orders of magnitude.

Dewdrop is implemented on the Intel WISP. WISP has a 4m range, a 10uF capacitor, and a 3D accelerometer. WISP has been used for such applications as sleep monitoring, neural monitoring, and cold-chain undersea neutrino detection, but all of these applications evaluate within less than one meter from the reader, where energy is plentiful. The challenges of running on CRFID tags are that they have a minuscule energy store, differing energy needs, and inherent inefficiencies, and they harvest energy even while executing programs. The WISPs take hundreds of milliseconds to charge, and only tens of milliseconds to discharge. Executing too early causes the tag to hit a black-out threshold where all state is lost. Because energy on the tags is stored in capacitors, charging the capacitor is non-linear; the more energy stored, the more time it takes to store additional energy. Dewdrop needs to store enough energy to compute, but so much as to waste time.

Dewdrop adaptively finds the wake-up voltage that maximizes the execution rate for the program and the RF environment, under the constraint that the runtime must be simple because cycles are tight on the tag and there are no floating-point units, etc. Dewdrop uses the heuristic that total waste is minimized when the wasted time from failures and over-charging is equal. In practice, this works well. To save energy, the implementation of Dewdrop uses an exponentially adapted low power wake-up to periodically poll the capacitor voltage, and Dewdrop uses a low power voltage sampling technique that reduces the energy from sampling the voltage by a factor of 4. Dewdrop matches the performance of the existing hardware mechanism for light tasks, and it doubles the range for heavy tasks. Dewdrop finds a wake-up voltage within 0.1V of optimal and achieves greater than 90% of the maximum rate for all distances. Technology trends will increase the range and performance of CRFIDs. The WISP platform and tools are available to the community in the form of open source hardware and software.

How does Dewdrop handle harvesting dynamics when the tags are mobile? It depends on how fast you are moving. If you aren't moving that fast, you'll be close to the operating point, but Dewdrop is intended for static or slow-moving objects. Hari Balakrishnan from MIT asked if it makes sense to partition tasks that use lots of computation across many tags. Buettner replied that most of the energy cost on the current WISP is communication, which is done in software, and you can compute for a very long time for the cost of a single communication. He said that they are looking into this problem with WISP 5.0. Why, for a given task, isn't the rate of charging and energy consumption estimated online? It's really hard to estimate the rate at which you get energy because it varies quickly, and if you are sufficiently close to the reader, you don't need to store any energy. You also can't profile offline because running around other WISPs changes the charging profile.

SSDAlloc: Hybrid SSD/RAM Memory Management Made Easy

Anirudh Badam and Vivek S. Pai, Princeton University

Anirudh Badam explained that memory in network systems is used both as a cache to reduce disk pressure with tools like memcache and as an index for things like proxy caches, WAN accelerators, and in-line data-deduplicators to reduce disk pressure. However, memory density is not scaling well. The cost of DRAM only scales linearly up to 64 gigabytes, and disk capacity is scaling, but disk speed is still around 200 seeks per disk per second. One solution to this problem is high speed disk arrays, but these are expensive and use more rack space. Their proposal is to use flash to overcome DRAM limits. Flash devices are fast for random reads, achieving one million IOPS per drive, but writes are slow and destroy the device. Because flash has low latency, it is closer to main memory than it is to disks.

Current transparent tiering relies on the OS page daemon to transparently move pages to swap. However, even a flash-aware pager only delivers 30% of the SSD's raw performance. This also has the drawback that both writes and frees are writes, which has a negative impact on performance on SSDs. Non-transparent tiering is possible, but this requires intrusive modifications to the application. The goals of SSDAlloc are to work with unmodified applications, to use DRAM as an object cache, and to use the SSD wisely by having the object store be log-structured.

SSDAlloc is able to act as an object cache rather than a page cache by storing one object per page (OPP) in virtual memory. Physical memory is split into a page buffer so as to not be wasteful. A small set of the pages contain a single object, and the rest of the memory is a compact object cache where multiple objects are packed into a single page. Pages are materialized on demand from the RAM object cache and the SSD. The SSD is used as a log structured object store. For comparison against SSDAlloc-OPP, SSDAlloc also has the option to implement a page cache, called SSDAlloc-MP. SSD maintenance is accomplished through object tables, which are similar to page tables. A garbage collector and log-writer copies and compacts objects in LRU order, using the object table to determine liveness. Objects that are written elsewhere and OPP objects that have been freed are treated as garbage.

SSDAlloc is implemented in 11,000 lines of C++ using the mprotect, mmap, and madvise system calls. The overhead of the SSDAlloc library is around 0.833 microseconds. For comparison, the latency of NAND is 30–50 microseconds. SSDAlloc is able to reach one million IOPS. Experiments were performed to evaluate SSDAlloc. SSDAlloc-OPP, SSDAlloc-MP, and SSD as swap were compared. SSDAlloc is able to write up to 32 times less data to the SSD. The performance of SSDAlloc-OPP is best at small object sizes, and SSDAlloc-OPP achieves a 2–4x improvement over SSDAlloc-MP. SSDAlloc is able to deliver 90% of the raw SSD random read performance.

In the Q&A, Bill Bolosky claimed that sweating about the log structured object store is not necessary, because the FTL on the SSD happens underneath you. Bolosky also asked about the performance of SSDAlloc for applications that actually fit into memory. Badam replied that you don't have to translate all of your malloc calls into SSDAlloc; you only should translate memory-intensive calls. Indices are stored with malloc; objects are stored with SSDAlloc. Aaron Gember from University of Wisconsin—Madison asked if performance is killed if the application has very small objects and the object table is larger than the size of memory. Badam replied that the object store is not kept in DRAM, and the object tables are implemented the same way as page tables. They have made optimizations to fit the object table into DRAM.

Debugging and Correctness

Summarized by Kristin Stephens (ksteph@cs.berkeley.edu)

Model Checking a Networked System Without the Network

Rachid Guerraoui and Maysam Yabandeh, EPFL

Maysam Yabandeh explained that there are two steps to dealing with bugs in a distributed system: testing and debugging. The system he presented focused on testing. Testing is done by exploring states, performing some transitions, and verifying user-specified invariants on the explored states—in other words, model checking (MC). The classical approach to model checking is to keep track of global states, which is a combination of system state and network state. Keeping things as only global state, however, means that global state changes following a change into any of the involved local states as well as the network state. This exacerbates the exponential explosion problem in number of possible states.

To alleviate the exponential explosion Maysam presented his idea of Local Model Checking (LMC). The global state is broken up into local states for each node with a shared network state. However, testing all possible combinations of local states and with all the different network messages could lead to invalid system states. Therefore, a soundness verification technique is used before reporting something as a bug. When a bug is found it is checked for soundness before being reported. Soundness verification looks at the partial order of predecessor states and transitions and sees if a total order can be found. If a total order can be found it could happen in a real run of the system. The evaluation of the system presented was testing a Paxos implementation with three nodes and one proposal. They were able to both rediscover previously known bugs and to find a new bug.

Sylvia Ratnasamy from Intel Labs Berkeley asked if Maysam had a sense of the general applicability of LMC. He responded that Paxos is known to be very complicated. The results from testing are a sign that it'll give good performance on other systems. However, the results do not necessarily mean that LMC's impact on other protocols would be as profound as its impact on Paxos.

Fate and Destini: A Framework for Cloud Recovery Testing

Haryadi S. Gunawi, University of California, Berkeley; Thanh Do, University of Wisconsin, Madison; Pallavi Joshi, Peter Alvaro, and Joseph M. Hellerstein, University of California, Berkeley; Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison; Koushik Sen, University of California, Berkeley; Dhruba Borthakur, Facebook

We have entered the cloud era, where the use of thousands of commodity machines means that rare hardware failures become frequent. Or, as the speaker, Haryadi Gunawi, put it, our forecast is "cloudy with a chance of failure." With failures becoming common, failure recovery becomes important. However, this is hard to get right, because testing is not advanced enough and recovery is often underspecified. Haryadi said we need two advances: a way to exercise complex multiple, diverse failures and a way to write recovery specifications. He presented FATE (Failure Testing Service) and DESTINI (Declarative Testing Specifications) for each advance, respectively.

FATE exercises multiple diverse failures. By doing so, FATE faces the challenge of a combinatorial explosion of multiple failures; with a brute-force approach, tens of thousands of multiple failures require over 80 hours to exercise. To quickly explore failures and find bugs, Haryadi presented two pruning strategies that improve testing time by an order of magnitude. DESTINI facilitates recovery specifications, which verify that the system under test is correct under failures. To make the specifications "developer friendly" (i.e., clear and concise), they experimented with Datalog, a declarative relational logic language. DESTINI only interposes I/O calls from which expectations of correct behavior and the actual behaviors can be deduced easily. FATE and DESTINI are written in around 6000 LOC in Java. For evaluation, they tested it on HDFS and found 22 new bugs, eight of which can only be found if multiple failures are injected. They also reproduced 51 known bugs.

Haryadi was asked about the news headlines he mentioned at the beginning of his talk, which showed various datacenters and companies losing data or having downtime. Did he have a sense of how many were from multiple concurrent failures? Haryadi responded that it was hard to tell from just the news. However, a lot of papers have explained that more bugs are found when injecting multiple failures.

SliceTime: A Platform for Scalable and Accurate Network Emulation

Elias Weingärtner, Florian Schmidt, Hendrik vom Lehn, Tobias Heer, and Klaus Wehrle, RWTH Aachen University

Elias Weingärtner started his talk discussing the pros and cons of network testbeds, network simulations, and network emulations. A network emulator, in particular, takes a real-world client and a discrete event-based network simulator. However, each has a different timing concept. Network simulation uses a series of discrete events, while real-world clients depend on continuous wall-clock time. The common solution is to bring the two times together by waiting between events, but many simulators are not real-time compatible, which causes time drift.

To prevent time drift, Elias presented the SliceTime system, which takes the approach of slowing down the client to match the simulator's speed. Clients are placed inside virtual machines and a barrier algorithm is used to limit time drift. SliceTime is implemented with a virtual machine for Linux and Windows, the ns-3 simulator, and a Xen hypervisor. The synchronizer implements the barrier synchronization algorithm and a modified sEDF scheduler is used to execute Xen domains for a time-slice duration. The event scheduler checks to see if the next event in the queue resides in the current time slice—if yes, it lets the event go, and if not, it blocks.

SliceTime was evaluated at different slice sizes, network configurations, and traffic setups. They found that it is resource-efficient, with low overhead for even 1 ms time slices. It is open source at http://www.comsys.rwth-aachen.de/projects/slicetime.

Jason Li from Intelligent Automation, Inc., asked if SliceTime was using an actual operating system or just an application connected to ns-3 through a socket. Elias replied that SliceTime does not make assumptions about what is put in the VM. Jason then asked whether the WiFi add-on was adapted for the ns-3 WiFi model or put in the VM. Elias said they implemented a device driver that gave the VM WiFi access into the simulation. Did every real client have a shadow node in the simulation and confirmation that the client never transmitted over any real interface? Yes. How does SliceTime compare to the optnet shadow model and hardware loop? It is similar, but the idea of network emulation was introduced 10 years ago. The key idea of SliceTime is that it slows down the simulation. Aaron Gember from the University of Wisconsin—Madison wondered what to keep in mind when choosing the size of a time slice. Elias said that the RTT is important and that the slice should be smaller than the network's RTT. How was time handled in the system—the simulator time is quantized, but real-world time is not, so does SliceTime attempt to quantize the client's time? Elias explained that the client time is not quantized and therefore does not execute in an entirely deterministic fashion. This also means that SliceTime cannot do absolute replicability.

Mobile Wireless

Summarized by Jeff Terrace (jterrace@cs.princeton.edu)

Accurate, Low-Energy Trajectory Mapping for Mobile Devices

Arvind Thiagarajan, Lenin Ravindranath, Hari Balakrishnan, Samuel Madden, and Lewis Girod, MIT Computer Science and Artificial Intelligence Laboratory

Arvind Thiagarajan presented a new method for providing accurate trajectory maps for mobile devices. Existing applications use a method that frequently polls the mobile unit's GPS device. This works well, but it quickly drains the battery of the unit, because GPS devices require a relatively large amount of power. There are also existing methods that use cell tower triangulation to attain point localization, but the low accuracy doesn't work well for trajectory maps.

The novel method proposed here, called CTrack, uses the low-power cell signal, combined with a unit's low-power accelerometer and compass to map these sensor readings to a trajectory. The method takes the recorded fingerprints from a device and maps them to grid locations using a dynamic programming algorithm. The results of this method produces trajectories with 75% precision at a small fraction of the power required for a GPS solution.

Responding to questions, Arvind stated that they did not measure the effect of road density; this method also does not work well for unmapped trajectories (such as running in a field). Mapping the city of Boston required 125 hours of training data. When asked if the HMM could use accuracy level of the fingerprints as input, Arvind said that when they tried this, it actually reduced the flexibility of the HMM, producing worse results.

Improving Wireless Network Performance Using Sensor Hints

Lenin Ravindranath, Calvin Newport, Hari Balakrishnan, and Samuel Madden, MIT Computer Science and Artificial Intelligence Laboratory

Lenin Ravindranath presented a new wireless protocol for mobile devices such as smartphones and tablets. Existing wireless protocols work well when a device is mostly in a fixed position, but when a device is in motion, packet loss is bursty, resulting in poor throughput.

A novel algorithm for rate adaptation, specifically designed for devices in motion, called RapidSample, gets much better throughput than traditional algorithms when a device is in motion. The problem with RapidSample, however, is that it performs poorly when a device is not in motion.

The key insight of this work is that they leverage the sensors provided in many mobile devices (e.g., GPS, accelerometer, compass, or gyroscope) to give hints to the wireless protocol stack as to whether the device is in motion. An evaluation of mixed-mobility traces shows that the hint-aware algorithm outperforms all other methods, giving ideal throughput when a device is either static or mobile.

Jason Li asked about how useful RapidSample is when the movements are small, and about the dangers of using the tool. Lenin replied that the tool works with sensor hints, but at a rather coarse scale. Using GPS does require a lot of energy, and designing a wireless device to take advantage of these sensor hints is difficult. Someone asked about supplying mapping information, for example, in large offices, and Lenin replied that they wanted to work without maps using commodity appliances.

Poster Session

First set of posters summarized by Kristin Stephens (ksteph@cs.berkeley.edu)

Assuring Network Service with Bandwidth and Integrity Based Fairness

Fariba Khan (fkhan2@illinois.edu) and Carl A. Gunter, University of Illinois Urbana-Champaign

Authentication is a desirable property on the Internet. However, ISPs need direct incentive to install it on their networks. Fariba's poster presented the idea of using priority-based queuing to provide this incentive. Authenticated packets would be given higher priority over unauthenticated packets. Included in the idea is allowing fine-grained authentication based on how many addresses the authenticated mask covered, with the more addresses a mask covered, the lower the priority.

Predicting the Safety of Web Logins with Piigeon

Xiao Sophia Wang (wangxiao@cs.washington.edu) and David Choffnes, University of Washington; Patrick Gage Kelley, Carnegie Mellon University; Ben Greenstein, Intel Labs Seattle; David Wetherall, University of Washington

Many Web sites send their users' login information in the clear, without telling their users. Sophia's poster presented a project that had three goals: predict if the login is sent in the clear, inform the user before this happens, and understand how logins are protected in the wild. To do this, Web pages were analyzed using various heuristics. A Firefox plugin has been developed and can be found at http://piigeon.org.

Wide-Area Datacenter Selection and Traffic Engineering for Online Service Providers

Srinivas Narayana (narayana@cs.princeton.edu), Joe Wenjie Jiang, and Jennifer Rexford, Princeton University

This poster presented the observation that mapping nodes and datacenters do not exchange information with each other when it comes to load balancing and performance. This then begs the question, is it useful to share information? The poster explained that sharing path performance, routing information, and mapping information between mapping nodes and datacenters can help improve both mapping and routing decisions. Datacenters can use the shared information to choose which paths to use when they are multihomed. And the mapping nodes can use the shared information to choose which datacenter to send requests to. Guided by an optimization framework, the poster also describes an architecture in which these decisions are performed in a distributed fashion with intermittent information sharing.

Don't Settle for Eventual: Stronger Consistency for Wide-Area Storage

Wyatt Lloyd (wlloyd@cs.princeton.edu) and Michael J. Freedman, Princeton University; Michael Kaminsky, Intel Labs; David G. Andersen, Carnegie Mellon University

The CAP theorem states that a distributed data store can only have two of three properties: strong consistency (linearizability), availability, and partition tolerance, so most settle for eventual consistency. This poster presented a new consistency model for distributed systems that still permits availability and partition tolerance. By combining causal consistency—related ops appear in the correct order—and per-key sequential consistency, they achieve consistency that is stronger than eventual. Wyatt Lloyd presented a system that realizes this new consistency model. It includes a modified key-value store and a client library with calls that track causality and a multiget operation that provides a consistent view of multiple keys.

dBug: Systematic Testing of Distributed and Multi-Threaded Systems

Jiri Simsa (jsimsa@cs.cmu.edu), Garth Gibson, and Randy Bryant, Carnegie Mellon University

This poster presents the design, implementation, and evaluation of dBug—a tool for systematic testing of concurrent programs. dBug repeatedly executes a test of a program while controlling the order in which concurrent events, such as intra- and inter-process synchronization and communications, execute. By doing so, repeated execution of a test can systematically explore different behaviors of the program and outcomes of the test. The implementation uses run-time interposition as a mechanism for transparently integrating the tool with existing programs. The evaluation used the tool to identify a number of concurrency errors such as deadlocks and incorrect usage of shared library API in unmodified distributed and multi-threaded programs.

vFlood: Opportunistic Flooding to Improve TCP Transmit Performance in Virtualized Clouds

Sahan Gamage (sgamage@purdue.edu), Ardalan Kangarlou, Ramana Rao Kompella, and Dongyan Xu, Purdue University

When multiple virtual machines (VMs) share a given CPU, VM scheduling latencies can be on the order of a few milliseconds, and may contribute to increasing the perceived round-trip times for TCP connections in sub-millisecond datacenter networks, causing significant degradation in throughput. vFlood is a lightweight solution which allows the VM to opportunistically flood packets to the driver domain and offloads the TCP congestion control to the driver domain, in order to mask the effects of virtualization. They found that this significantly improves the performance of small flows and is non-trivial for larger flows.

Suppressing Malicious Bot Traffic Using an Accurate Human Attester

Muhammad Jamshed, Younghwan Go (yhwan@ndsl.kaist.edu), and KyoungSoo Park, KAIST

A new human attestation method to suppress malicious bots, Not-a-Bot, ensures that a user is human in a message by associating an arbitrary input event proof occurring within a specific time window (usually 1 second) during its message construction. This model is vulnerable to forgery by smart bots that can exploit this time interval. This poster presents a model where human use is more securely bound to a legitimate message by attaching proof of all relevant input events generated during message construction. It uses a TPM and processor's late-launch technology that is available in all modern machines. The results show that the overhead for generating proofs is tolerable for human use and practical for all applications.

On Lookups in Content-based Routers

Ashok Anand (ashok@cs.wisc.edu), Nilay Vaish, and Aditya Akella, University of Wisconsin—Madison

Content-driven networks are great, but many of these systems do not include the details of their implementation. This poster focused on the problem of frequent updates in a router. To speed up the process of updating they used several techniques: fingerprint packets, reduced communication between threads, a shared partitioned table, a lock at the level of table partition, batch jobs, and a particular table partition access sequence to reduce contention. Their results show this greatly speeds up updates.

Towards Transactional Cloud Resource Orchestration

Changbin Liu (changbl@seas.upenn.edu), University of Pennsylvania; Yun Mao, Xu Chen, and Mary F. Fernandez, AT&T Labs—Research; Boon Thau Loo, University of Pennsylvania; Kobus Van der Merwe, AT&T Labs—Research

Infrastructure as a service, cloud computing, and datacenter resource management are hard. This poster presented a framework that provided the ability to make cloud resource orchestration like a transaction. This means it had the properties of atomicity, consistency, isolation, and durability. This poster included a demo in which they live-migrated a virtual machine from one datacenter to another over a wide-area network.

Second set of posters summarized by Aaron Gember (agember@cs.wisc.edu)

SNEAP: A Social Network-Enabled EAP Method: No More Open Hotspots

Aldo Cassola, Tao Jin, Harsh Kumar, Guevara Noubir, and Kamal Sharma, Northeastern University

SNEAP leverages existing social networks to grant users more ubiquitous access to wireless connectivity. Unlike prior approaches which rely on keys provided by a special authentication service, SNEAP leverages a modified version of WPA-Enterprise to authenticate users and verify trust relationships. Individuals who wish to share their wireless access point (AP) install a special SNEAP OpenWRT image on their router and register the router with the SNEAP radius server. When friends are in the vicinity and wish to connect to the AP, they authenticate using their social network, e.g., Facebook, credentials. The radius server verifies the credentials and ensures that a trust relationship, e.g., friendship, exists between the AP owner and the authenticating user. A simple demonstration showed the router registration process and user authentication process using a wireless router and two laptops (one serving as the owner's home computer and one as a client) with both tasks relying on a Facebook application and remote radius server.

Network Configuration Analysis

Theophilus Benson, University of Wisconsin—Madison; Hyojoon Kim, Georgia Institute of Technology; Aditya Akella, University of Wisconsin—Madison; Nick Feamster, Georgia Institute of Technology

Enterprise networks are difficult to configure, requiring thousands of lines of configuration commands distributed across thousands of devices to set up a wide array of protocols and standards. The objective of this work is to understand (1) how the network changes over time and (2) how operators interact with different devices, with the goal of designing better network management tools. The authors use a five-year history of configuration files from two enterprise networks to analyze the changes that occur. They observe that the most frequent change, which involves an interface and a route protocol, occurs to add a department. Other changes that occur frequently include adding a department with security controls, changing the control plane, and changing the addresses assigned to a department. However, the most prevalent configurations are not the most frequently changed configurations. Analysis also shows that the majority of configuration changes occur in routers, rather than in firewalls or switches. Lastly, most switch changes occur during the workday, while most firewall changes occur in the early evening.

BISMark: A Platform for Studying Home Networks

Walter de Donato, University of Napoli Federico II; Srikanth Sundaresan and Nick Feamster, Georgia Institute of Technology; Renata Teixeira, CNRS/UPMC Sorbonne Universités; Antonio Pescapé, University of Napoli Federico II

BISMark relies on instrumented home routers to measure the behavior of home networks and the last-hop broadband links that connect homes to the Internet. Two types of devices are being deployed in homes: a powerful NoxBox that runs Linux (16 of which have already been deployed) and a less-flexible but more stable Netgear router. With these devices, the authors can study the impact of ISP policies, the impact of certain home network configurations, e.g., WiFi, and the usage profiles of home network users. Active measurements include TCP/UDP throughput, last-mile latency, upstream/downstream jitter, packet loss, and DNS delay and availability. Passive measurements include per-application throughput (planned for the future), packet header captures, WiFi client and AP behavior, and DHCP requests. Measurements have analyzed an ISP policy termed PowerBoost—receiving a high burst throughput for a few seconds—explaining how it is implemented, how variable it is, and its impact on user perception of throughput. Analysis has also uncovered the presence of significant buffering in modems, which can cause up to multiple seconds of latency when the uplink is saturated.

Seattle: The Internet as a Testbed

Jeff Rasley, Monzur Muhammad, Alex Hanson, Sebastian Morgan, Alan Loh, and Justin Cappos, University of Washington

Seattle is a peer-to-peer version of PlanetLab, providing an environment for Internet-wide networking experiments. Seattle relies on real user nodes to provide a realistic testing environment. While PlanetLab provides experimenters with Linux virtual machines connected to Internet2, Seattle provides experimenters with a Python environment on machines with home broadband connectivity. Seattle has diurnal end-user availability, some mobile nodes and some limited access due to firewalls and NATs, and runs on servers, mobile phones, and laptops. The testbed has been used in the classroom for projects on link state routing and distributed hash tables based on Chord. Researchers have used Seattle for YouTube CDN mapping, tracking mobility and diurnal patterns, testing peer-to-peer encrypted file storage, and many other projects. Seattle is available for use now; users who donate N machines get access to 10ˆN machines. See http://seattle.cs.washington.edu for more information.

An Empirical Study on the Person-to-Public Distribution of Tor Bridges

Xiao Wang, Jinqiao Shi, Binxing Fang, Qingfeng Tan, and Li Guo, Institute of Computing Technology, CAS, China

Tor bridges, a technology designed to allow unrestricted Internet access in the presence of censors, are typically distributed in an authority-to-public or person-to-person fashion. However, users may sometimes post Tor bridge IP addresses online, eliminating some of the secrecy. In this work, the authors survey the status of potential Tor bridges located via Web searches for the phrases "Here are your bridge relays" and "bridge IP:Port". Addresses were found on a small fraction of Web sites. Of the 579 potential bridges the searches reveal, 64 IPs were still serving as bridges, 95 were Tor relays, and 420 were unidentifiable. Unfortunately, such public bridges are a considerable portion of the available bridges, posting a threat to Tor's censorship-resistance.

Structured Comparative Analysis of Systems Logs Using Distalyzer

Karthik Nagaraj, Charles Killian, and Jennifer Neville, Purdue University

Distributed system logs contain lots of information that is useful to developers, but the size of these logs makes manual analysis infeasible. Distalyzer is a distributed system performance analyzer that automatically looks for differences between two sets of logs. Logs are gathered from two different systems run in the same environment, or from the same system run with different parameters. Systems provide two types of logs, state logs and event logs, each with a known structure. Distalyzer compares the occurrences of events or the values of states to identify differences between the two system runs. The discovered differences are ranked and presented to the developer to help identify performance leaks.

Work in Progress: Uncovering the Privacy Implications of Web Usage

Hendrik vom Lehn, Jó Ágila Bitsch Link, and Klaus Wehrle, RWTH Aachen University, Germany

Internet users visit a wide array of Web sites, providing personal information to many of these sites. For example, users may provide their name and email address to gain access to a site. Knowing what information a user has disclosed during their browsing sessions, enables a user to identify potential information leaks and make better decisions about their Web usage. Especially important is the detection of hidden information leakage, i.e., private information it is not obvious a user is disclosing. Gathering this information is facilitated by a browser plugin which receives a copy of all HTTP traffic. The traffic is analyzed, using various modules, to identify personal data a user has disclosed; the data is stored locally along with metadata from the HTTP exchange. Users can view the extracted information in a summarized form to better understand the level of privacy in their Web browsing, and future tools may automatically warn users of potential privacy concerns.

Armonia: A Highly Available Distributed Transactional Memory Store

Mehul A. Shah, Nathan Binkert, Stavros Harizopoulos, Wojciech Golab, and Indrajit Roy, HP Labs

Armonia improves on prior distributed transactional memory stores by using Paxos-based replication instead of a primary-backup design. The system has four goals: (1) scalability to 100s of terabytes of memory, (2) microseconds latency, (3) transactional consistency, and (4) five nines of availability. Armonia integrates transaction execution and commit protocols with Paxos to avoid extra network round trips and achieve low latencies. Using Paxos provides better availability because data is replicated more than once, compared to only two copies in a primary-backup design, and because Paxos does not need accurate failure detection.

InContext: Simple Parallelism for Distributed Applications

Sunghwan Yoo, Hyojeong Lee, Charles Killian, and Milind Kulkarni, Purdue University

InContext is an event-driven framework suitable for systems seeking parallel execution capabilities. With the InContext model, events can be global (enabling them to both read and write global service state), anon (enabling them to only read global system state), or none (providing no access to global service state). During execution, only one event at a time can be in the global state. Events desiring to enter the global or anon states must wait for existing events to commit or enter the anon state. Optionally, a commit may be deferred to keep a logical ordering of events. The authors modified Mace to support this parallel event model. During runtime, applications make an implicit upgrade to move from none to anon, and applications add an explicit downgrade call to move from global to anon. A model checker and simulator are provided as additional tools.

A Service Access Layer, at Your Service

David Shue, Matvey Arye, Prem Gopalan, and Erik Nordström, Princeton University; Steven Y. Ko, SUNY, Buffalo; Michael J. Freedman and Jennifer Rexford, Princeton University

The Internet was designed for host-to-host communication, but recent trends are toward a service-centric Internet. In the service-centric architecture proposed by the authors, a service access layer is added between the transport and network layers. The service access layer enables applications to use topology-independent service names instead of topology-dependent addresses. The transport layer is purely responsible for data delivery; the service access layer deals with resolving a service, initiating and terminating connections to a particular instance, and maintaining affinity to that instance across network address changes such as during VM migration or client mobility events. When an application desires to establish a TCP connection to a specific service, the service layer intercepts the initial SYN and forwards the packet to an authoritative service router that recursively resolves the packet to an instance of the service, which replies directly to the source with a SYN/ACK. After the connection is established, all further communication occurs directly between a user and the selected service instance. In the demo, an Android client broadcast a request for a specific service, and the phone was served by the server (running on each laptop) whose SYN-ACK was received first.

Third set of posters

Block-based Bitrate Control for Wireless Networks

Xiaozheng Tie (xztie@cs.umass.edu), Anand Seetharam (anand@cs.umass.edu), Arun Venkataramani (arun@cs.umass.edu), Deepak Ganesan (dganesan@cs.umass.edu), and Dennis L. Goeckel (goeckel@ecs.umass.edu)

The poster presented BlockRate, a wireless bitrate control algorithm designed for blocks, or large contiguous units of transmitted data, as opposed to small packets. Recent trends in research as well as in practice (e.g., 802.11n) suggest significant overhead amortization benefits of blocks. Yet state-of-the-art bitrate algorithms are optimized for adaptation on a per-packet basis, so they can either have the amortization benefits of blocks or high responsiveness to underlying channel conditions of packets, but not both. To bridge this disparity, BlockRate employs multiple bitrates within a block that are predictive of future channel conditions. In each feedback round, BlockRate uses a history-based scheme to predict the SNR for packets within the next block. In slow-changing scenarios, such as indoor mobility, BlockRate uses a simple linear regression model to predict the SNR trend over the next block. In fast-changing scenarios, such as vehicular mobility, BlockRate uses a path loss model to capture more significant SNR variations. They have implemented a prototype of BlockRate in a commodity 802.11 driver and evaluated it via deployment on an indoor mesh testbed as well as an outdoor vehicular testbed. The evaluation shows that BlockRate achieves up to 1.4x and 2.8x improvement in goodput under indoor and outdoor mobility, respectively.

LocalFlow: Simple, Local Flow Scheduling in Datacenters

Siddhartha Sen (sssix@princeton.edu), Sunghwan Ihm, Kay Ousterhout, and Michael J. Freedman, Princeton University

LocalFlow is a completely local approach to flow routing in datacenter networks. It addresses the problem of large flows, which can significantly degrade network utilization and starve other flows if routed through oversubscribed links. LocalFlow is an efficient bin-packing algorithm that runs locally on each network switch. It uses two key ideas. First, it proactively splits and rate-limits flows to ensure that they are small enough before collisions occur. This guarantees provably optimal, max-min fair routing in standard fat-tree networks. Second, it uses a splitting technique that leverages wildcard rules in upcoming programmable commodity switches to group contiguous packets into "flowlets" to minimize end-host reordering. Previous flow-routing algorithms rely either on centralized schedulers or on end-host control of multiple paths. The first approach lacks parallelism and thus scalability, while the second approach cannot predict the paths of flows and thus only works if the flows-to-paths ratio is high. They presented the design and theoretical analysis of LocalFlow, as well as preliminary simulation results that demonstrated the practicality of splitting. For example, based on packet traces from a university datacenter switch, LocalFlow splits less than 4.3% of total flows on average, using approximate splitting to within 5% on a 1024-host fat-tree network.

A Memory-Efficient, High-Performance Key-Value Store

Hyeontaek Lim (hl@cs.cmu.edu), Bin Fan, and David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

This work develops a memory-efficient, high-performance key-value store. It focuses on indexing techniques in fast key-value stores as the indexing data structure is one of the main sources of the memory consumption in key-value store systems (e.g., 4 bytes/item), while DRAM is becoming a scarcer resource, as flash/disk's capacity per dollar is growing much faster than DRAM's. This work proposes three basic key-value store designs based on new indexing data structures (partial-key cuckoo hash tables and entropy-coded tries) and combines these basic stores to build a full key-value system; by inserting new data to a write-optimized basic store and gradually moving the data to a very memory-efficient basic store, this system achieves approximately 0.7 bytes/item, while a data retrieval requires only 1.01 flash reads, which allows nearly full utilization of flash drive random read performance.

KARMA: Trade Your Idle Resources Now for Elastic Scale Out Later

Shriram Rajagopalan (rshriram@cs.ubc.ca), Dhirendra Singh Kholia (dkholia@cs.ubc.ca), Mohammad Shamma (mshamma@cs.ubc.ca), and Andrew Warfield (andy@cs.ubc.ca)

Virtual machine (VM) utilization in the cloud seldom exceeds 10–12%, as these were, in an earlier life, underutilized physical servers. But these VMs are charged for resources like CPU and memory even though they remain idle during low load periods. This work proposes a system called KARMA that will leverage techniques used by grid systems such as Condor, BOINC, etc., to pool idle resources of VMs belonging to a community of users in the cloud. KARMA will use container virtualization techniques such as OpenVZ to create low overhead application containers (sandboxes). During peak loads, instead of scaling in cloud by launching additional short-lived VMs (and paying for them), users could scale their applications by launching application containers (for free) that draw upon the resources in the KARMA pool. Container virtualization offers both performance and isolation from the host VM's perspective. To ensure information privacy from the guest application's perspective, they propose using "malwarized applications" that leverage malware creation techniques such as packing and code obfuscation to deter the host VM from tampering with the guest application. Performing resource sharing from within virtual machines (VMs) has the potential to reduce user costs at the expense of global resource optimization. Their intention is to provide a short-term benefit for some users, that (antagonistically) motivates a longer term optimization of how resources are accounted and charged.

DARD: Distributed Adaptive Routing for Datacenter Networks

Xin Wu (xinwu@cs.duke.edu) and Xiaowei Yang, Duke University

From http://www.cs.duke.edu/events/?id=00000001331

Datacenter networks typically have many paths connecting each host pair to achieve high bisection bandwidth for arbitrary communication patterns. Fully utilizing the bisection bandwidth may require flows between the same source destination pair to take different paths to avoid hot spots. However, the existing routing protocols have little support for load-sensitive adaptive routing. This work proposes DARD, a Distributed Adaptive Routing architecture for Datacenter networks. DARD allows each end host to adjust traffic from overloaded paths to underloaded ones without central coordination. They use an OpenFlow implementation and simulations to show that DARD can effectively use the network's bisection bandwidth. It out-performs previous solutions based on random flow-level scheduling, and performs similarly to previous work that assigns flows to paths using a centralized scheduler but without its scaling limitation. They use competitive game theory to show that DARD's flow scheduling algorithm makes progress in every step and converges to a Nash equilibrium in finite steps. The evaluation results suggest its gap to the optimal solution is likely to be small in practice.

WebCloud: Enabling More Direct Content Exchange Between Web Clients

Fangfei Zhou (youyou@ccs.neu.edu), Liang Zhang (liang@ccs.neu.edu), and Eric J. Franco, Northeastern University; Richard Revis, Jandrell, Pearson & Revis Ltd.; Alan Mislove (amislove@ccs.neu.edu) and Ravi Sundaram, Northeastern University

From http://www.northeastern.edu/expo/view_abstracts/abstract.php?sid=1814

We are at the beginning of a shift in how content is created and exchanged over the Internet: today, individual users, powered by devices like digital cameras and services like online social networks, are creating content that represents a significant fraction of Internet traffic. As a result, compared to content shared over the Internet just a few years ago, content today increasingly is generated and exchanged at the edge of the network. Unfortunately, the existing techniques and infrastructure that are still used to serve this content, such as centralized content distribution networks, are ill-suited for the new patterns of content creation and exchange, resulting in a mismatch of infrastructure and workload.

In this work, they take a step towards addressing this situation by introducing WebCloud, a content distribution system that enables more direct content sharing between users in existing online social networks. WebCloud works by adding a small amount of JavaScript to a social network's Web pages, locally storing content that each user views. When another user browses the content, the JavaScript fetches it from one of the user's online friends instead of directly from the social networking site. The result is a more direct exchange of content between users; essentially, WebCloud leverages the storage and bandwidth resources of social networking users to help serve content. Because WebCloud is built using techniques already present in many Web browsers, it can be applied today to many online social networking sites. They demonstrated the practicality of WebCloud with simulations and a prototype deployment.

Seeking Efficient Data-Intensive Computing

Elie Krevat (ekrevat@andrew.cmu.edu) and Tomer Shiran, Carnegie Mellon University; Eric A. Anderson, Joseph Tucek, and Jay J. Wylie, HP Labs; Gregory R. Ganger, Carnegie Mellon University

From http://www.cs.cmu.edu/~ekrevat/

New programming frameworks for scale-out parallel analysis, such as MapReduce and Hadoop, have become a cornerstone for exploiting large datasets. However, there has been little analysis of how these systems perform relative to the capabilities of the hardware on which they run. They have developed a simple model of I/O resource consumption and applied it to a MapReduce workload to produce an ideal lower bound on its runtime, exposing the inefficiency of popular scale-out systems. Using a simplified dataflow processing tool called Parallel DataSeries (PDS), they demonstrated that the model's ideal can be approached within 20%. Current research explores why any DISC system built atop standard OS and networking services faces a gap between ideal and actual performance. They have found that disk stragglers and network slowdown effects are the prime culprits for lost efficiency in PDS. They are also building up PDS into a more feature-rich system (e.g., to support fault tolerance), to understand all areas where efficiency is lost at scale.

Datacenters Learning to Share

Summarized by Andrew Ferguson (adf@cs.brown.edu)

Mesos: A Platform for Fine-Grained Resource Sharing in the Datacenter

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

Recent attention to the problem of performing large-scale computation on commodity clusters has yielded numerous computational frameworks, such as Hadoop, Dryad, and CIEL. Organizations that wish to use several frameworks, or even several versions of the same framework, in isolation must currently use multiple clusters, which can require extensive data duplication and yield idle resources.

Matei Zaharia presented Mesos, a layer for sharing common resources such as CPU cores, memory, and data blocks over which diverse computational frameworks can run. Besides enabling multiple frameworks to fairly share the same cluster, Mesos makes it easier to build and deploy new, specialized frameworks for more unique computations. To illustrate this point, Zaharia worked with his group to develop Spark, a lightweight system for machine learning which achieves significantly higher performance than Hadoop.

The core of Mesos is a small microkernel-like engine which makes resource offers to the frameworks running above it. These resource offers describe slots of CPU and memory that are available on particular cluster nodes. The frameworks accept or reject the offers and employ their own scheduling logic on accepted offers. Mesos uses dominant resource fairness (see summary below) to choose which resource offers to make. Because of this design, applications running on Mesos perform best when they have many fine-grained tasks to fill the offers. However, this is not a requirement; Zaharia presented examples of Mesos clusters simultaneously supporting Hadoop, MPI, Torque, and Spark applications.

Mesos consists of about 20,000 lines of C++ and is available as an open-source Apache Incubator project. The Mesos masters have only soft-state, and they provide high availability failover using Apache ZooKeeper. It is currently in use at Twitter, Conviva, and by researchers at UCSF.

George Porter asked how Mesos shares the network bandwidth between competing applications. Zaharia said that-Mesos does not do anything in particular but that the next presentation addresses this issue. Srikanth Kandula asked if the predictability of job completion times suffered. Zaharia replied that completion times depend upon the intra-framework scheduling policy, and jobs may finish faster. Arkady Kanevsky (VMware) asked about the mechanism used to isolate frameworks running on the same nodes, and Zaharia said that Mesos uses Solaris zones and Linux containers.

Sharing the Datacenter Network

Alan Shieh, Microsoft Research and Cornell University; Srikanth Kandula, Microsoft Research; Albert Greenberg and Changhoon Kim, Windows Azure; Bikas Saha, Microsoft Bing

Today's datacenters occasionally suffer from poor network performance due to application interference. A single application may monopolize a shared resource with many TCP flows, use a more aggressive variant of TCP, or simply use UDP; malicious users can launch denial of service attacks against other virtual machines or entire racks.

Alan Shieh presented Seawall, a client-side solution to sharing the datacenter network by decoupling network allocation from applications' traffic profiles. Seawall sits in the hypervisor of each datacenter node and establishes a single tunnel between each source and destination VM. It determines per-link rate limits and converts them to per-tunnel rate limits.

Seawall also makes use of periodic congestion feedback (e.g., percentage of lost packets) and ECN marks to adapt the rate of traffic permitted through each tunnel. Shieh describes this as "link-oriented congestion control," and it can employ standard congestion control loops such as AIMD, CUBIC, and DCTCP. A heuristic is used to convert path-congestion feedback into link-level congestion feedback, because a congested link will result in path congestion on many tunnels. Path feedback is combined in proportion to the amount of traffic on that path.

Shieh presented two evaluations of the Seawall system. In the first experiment, an application attempted to use a UDP flood to deny service over a particular link. In the second, an application attempted to gain more than its fair share of the network by using many TCP flows. In both cases, the Seawall system appropriately isolated and limited the malicious traffic. Finally, Shieh compared Seawall with related works such as SecondNet, Gatekeeper, and CloudPolice.

Steven Hand (Cambridge) asked how this could be scaled to an Internet-wide topology, and Shieh answered that it would be difficult because the necessary topology information is not readily available. Ye Wang asked how this interacts with the virtual machine's existing network stacks. Shieh indicated that they modified the TCP stacks in the VMs to respect the congestion control parameters passed up from the Seawall system.

Dominant Resource Fairness: Fair Allocation of Multiple Resource Types

Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica, University of California, Berkeley

Many scheduling approaches, such as weighted fair-queueing, round-robin, and Hadoop and Quincy's task schedulers employ (weighted) max-min fairness to fairly allocate resources, for several reasons. The first reason is that it provides a share guarantee: each of *n* users will get at least 1/*n* of the scheduled resource. Furthermore, if one user requests less of the shared resource, the newly available resource is split evenly by the other users. Secondly, max-min fairness is strategy-proof—users are not incentivized to lie, and would not improve their performance by requesting more resources than needed. Finally, max-min fairness provides the flexibility to implement a variety of scheduling policies, such as proportional sharing, fixed or dynamic priorities, and reservations.

Ali Ghodsi presented dominant resource fairness (DRF), a strategy for extending max-min fairness to the domain of many users making heterogeneous demands on multiple resources. The development of this strategy is important for today's datacenters, in which not all tasks require the same ratio of CPU to memory. In the DRF model, all allocated resources are divisible, and users record their task requirements using a demand vector, such as <1 CPU, 4 GB of RAM>. The DRF algorithm then applies max-min fairness to the user's dominant resource—the resource in the cluster which is most tightly constrained.

Ghodsi compared DRF with two alternative policies: asset fairness, which equalizes each user's sum of resource shares, and Competitive Equilibrium from Equal Incomes (CEEI). CEEI gives each of *n* users 1/*n* of each resource, which can then be freely traded in a market. The speaker showed that, unlike DRF, asset fairness does not guarantee that each user will get at least 1/*n* of each resource, and that CEEI is not strategy-proof. Finally, Ghodsi presented the results of a simulation of Facebook's Hadoop workload and showed that DRF-based scheduling could outperform the existing Fair Scheduler.

Derek Murray (Cambridge) wanted to know if the performance bottleneck was perhaps disk access and not memory or CPU. Ghodsi responded that they did not have disk access statistics, but confirmed that they had observed clusters hitting their CPU and memory limits. Michael Freedman asked if it was possible to make Amazon EC2 exchange strategy-proof; the speaker did not think it possible.

Wireless and More

Summarized by Hendrik vom Lehn (vomlehn@cs.rwth-aachen.de)

PIE in the Sky: Online Passive Interference Estimation for Enterprise WLANs

Vivek Shrivastava, Shravan Rayanchu, and Suman Banerjee, University of Wisconsin—Madison; Konstantina Papagiannaki, Intel Labs, Pittsburgh

Vivek Shrivastava started with an introduction to the use cases of interference estimation. In enterprise WLANs with a set of access points, a central management entity can be used to dynamically control the transmission power and channel assignment. The link interference ratio (LIR) is a helpful metric that is often derived from bandwidth tests. Such bandwidth tests, however, require downtime and are not scalable.

Shrivastava presented a new system for passive interference estimation (PIE) that is able to estimate the interference in a passive way in real time. Access points are equipped with a traffic sniffer and a clock synchronization mechanism. Reports from these traffic sniffers are sent to a wireless LAN controller, which then calculates the LIR based on the calculated station isolation loss rate and the interference loss rate.

They evaluated how fast PIE converges depending on the report periodicity and the occurring traffic patterns. Their measurements showed that 100 ms is enough in case of saturated traffic. An evaluation with real wireless LAN traces showed that in case of lighter traffic approximately 700 ms is required. Shrivastava then presented results of an application of PIE to data scheduling. The results showed that PIE outperforms existing approaches, especially in mobile scenarios. Shrivastava finished by telling the audience that measurements in production systems showed that hidden terminals and rate anomalies are indeed a problem and describing how PIE fits into related work and some limitations of PIE.

Dan Halperson (U. Washington) asked about data rate selection. Shrivastava answered that the caused interference depends on the chosen data rate of clients, but that they focus on interference estimation. Based on this, it would be possible to perform a more intelligent data rate selection. Shrivastava was asked if the numbers given earlier are for hidden terminal problems between access points. He explained that the numbers are for everything which the access points sees and thus also include transmissions from clients.

SpecNet: Spectrum Sensing Sans Frontières

Anand Padmanabha Iyer, Krishna Chintalapudi, Vishnu Navda, Ramachandran Ramjee, and Venkata N. Padmanabhan, Microsoft Research India; Chandra R. Murthy, Indian Institute of Science

Anand Iyer said that the wireless spectrum is currently underutilized and that new rules of the FCC make it possible to opportunistically access licensed spectrum. Current studies on the utilization of spectrum are, however, all static and cover only a few places around the world. In order to get better measurements, they decided to build a PlanetLab-like system for spectrum analyzers.

After presenting the goals and implementation challenges, Iyer presented the SpecNet architecture. A master server manages all requests and forwards them to slave servers which in turn are connected to the actual spectrum analyzers. Users can access the system using XML-RPC. The API provides both low-level access and more abstract commands through a uniform interface. After the request has been performed, the master server stores all results in a database.

Before explaining key challenges, Iyer gave a brief overview of how spectrum analyzers work. In order to decrease the required scan time, SpecNet supports several mechanisms which allow distribution of the scanning among several spectrum analyzers. Iyer presented two example applications of SpecNet: performing simple scans for a given area, and detecting and localizing violators, which requires coordination between the sensing devices. He presented a simple scan with a live demonstration. After that, he explained that expensive devices, the attenuation of buildings, and security concerns are current limitations of the system. Iyer said that they are looking for people with spectrum analyzers who want to participate in SpecNet: http://bit.ly/SpecNet.

How many sites are running the experiments? Currently, 10; they hope that more people will sign up.

Towards Street-Level Client-Independent IP Geolocation

Yong Wang, UESTC and Northwestern University; Daniel Burgener, Marcel Flores, and Aleksandar Kuzmanovic, Northwestern University; Cheng Huang, Microsoft Research

Aleksandar Kuzmanovic explained that GPS and WiFi are good localization methods for the end user, but not for a server that wants to know where the user is, which, for example, would be required for street-level online advertising. One way to determine the location of an Internet host is to use active vantage points that measure the delay and thereby approximate the host's location. More advanced approaches that also take the network topology and demographics information into account reach an accuracy of about 35 km (22 miles), which is insufficient for the desired applications. Kuzmanovic said that the system they developed is much better in this regard. Their system is based on two insights: many Web sites provide the geographical location of the host that is serving them and relative network delays from different landmarks can be used instead of absolute delays. Their system uses multiple steps to select the landmarks which are used to determine a host's location. In the last step, the hosts location is approximated using the landmark with the minimum delay to the targets. This mechanism has the advantage that it avoids the so-called last-mile inflation.

They evaluated their system using three datasets as ground truth: one from PlanetLab, a collected residential dataset, and the locations that search engine visitors searched for. Kuzmanovic explained that for these datasets they could decrease the median error (compared to existing approaches) from 35 km to about 1.5 km. Important factors that influence the resulting quality are the density of landmarks, the population density, and the type of access network through which the host is connected.

How easily could this approach be transferred to other parts of the world? They currently have data for 200,000 landmarks in the US, but one can do the same for other regions as well. Is this going to be an open system? They will make it a commercial system.

LEET '11: 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats

Boston, MA
March 29, 2011

Invited Talk

Summarized by He Liu (h8liu@cs.ucsd.edu)

Tor and Circumvention: Lessons Learned

Nick Mathewson, The Tor Project

Nick Mathewson introduced the Tor project. Tor is free open source software that enables Internet online anonymity, used by approximately 250,000 users every day. Anonymity means an attacker cannot learn who is communicating with whom in a communication network, and Tor aims to be a usable, deployable software that maximizes online anonymity. Anonymity serves different interests for different users. Anonymity is privacy for private citizens, network security for businesses, traffic-analysis resistance for governments, and reachability for human rights activists. To be anonymous, it is important to have company and hide in an anonymity network. The Tor project benefits good people more, because bad people are already doing well. Tor uses a multiple-hop relay network design.

Nick talked about how Tor evolves to have more blocking-resistance, i.e., is harder to be blocked by firewalls. Firewalls can block Tor if Tor's traffic has a unique fingerprint. To defend against these attacks, Tor's TLS handshake mimics the ones between Firefox and Apache. Attackers can also block Tor by preventing users from discovering relays by blocking relay directories or all relay nodes. To defend against this, Tor uses "bridge" relays. Bridge relays are relay nodes that are not listed in any directory, and Tor partially distributes addresses of bridges to users. Bridge relays help users to bootstrap and connect to other relays in the network.

Nick also showed several figures of user statistics of Chinese, Tunisian, Egyptian, Libyan, and Saudi users. These figures show that use of Tor has a strong correlation with big political events. Some of them also showed how Tor's blocking-resistance upgrades are reflected on user counts.

Finally, Nick talked about several challenges for the Tor Project, including how to defend against application-level attacks, how to guarantee Tor software integrity on software distribution, and how to educate people about using online anonymity software for better Internet security.

Dan Geer (InQTel) asked about transition from IPv4 to IPv6. Nick said Tor is now IPv4 only, but IPv6 is under consideration. IPv6 is not universally deployed over the Internet, and hence it is hard to use it to build a world-wide anonymity network. Stefan Savage asked how to debug a blocked node. Nick answered that mostly they manually cooperate with users in different countries to debug. Is Tor mostly used in countries with censorship? Nick said no, that the two biggest user bases are in the US and Germany. Dan Geer asked about defending against traffic correlation attacks. Nick said it is still unacceptably expensive and would require sacrificing too much usability, but they are happy to use it if related research produces more feasible results.

Attacking Large-Scale, Real-World Systems

Summarized by Gianluca Stringhini (gianluca@cs.ucsb.edu)

Exposing the Lack of Privacy in File Hosting Services

Nick Nikiforakis, DistriNet, Katholieke Universiteit Leuven; Marco Balduzzi, Institute Eurecom; Steven Van Acker and Wouter Joosen, DistriNet, Katholieke Universiteit Leuven; Davide Balzarotti, Institute Eurecom

File hosting services are becoming a popular way for users to share files on the Internet. Anytime a user uploads a file to a file hosting service (FHS), the service assigns a URI to it that univocally identifies it and allows other people to download it. Unlike traditional peer-to-peer, they allow people to share a file in a "private" way, since only those who know the URI can access the file.

This work studies to what extent FHSes guarantee privacy, and whether it is possible for an attacker to guess the URI that identifies a file and, thus, illegally download it. Nick said he studied how 100 FHSes generate their URIs, finding out that they typically generate them in a sequential way. In particular, 20 of them do not add any non-guessable information at all, making it trivial for an attacker to enumerate all the files uploaded to the service and download them.

Those services that do not use sequential numbers might appear to have better security, but Nick showed that many of them use short identifiers that are easy to brute-force. After getting the URI that identifies a file, an attacker can find out whether the file has been uploaded as private. To do this, it is enough to search for the file on a search engine. If no results are returned, there is a good chance that the file is private. Nick said that 54% of the files they crawled were private.

Nick said they then created some honey files that were "calling home" when opened, to see whether people crawled for them and downloaded them. These files were downloaded 275 times, by more than 80 unique IP addresses. This is the proof that there is a problem of people looking for private files on FHSes. To check whether the intentions of these people are malicious, Nick created a fake credit card trading Web site and put information about it, and the credentials to log into it, in some of the honey files. In total, they logged 93 logins in the fake Web site, by 43 different IP addresses. This proves that attackers use the information found in illegally retrieved data.

Someone asked if they were blacklisted during their trolling, and Nick replied, Only when they were greedy. They slowed down the rate and were fine. Chris Kruegel asked if any of the top five FHSes were using sequential identifiers, and Nick said that when they reported this issue to one of the top five, they explained they would slow down the ability to search, but not fix sequential numbering in URIs. Someone wondered what was displayed to the users who logged into the fake site. Nick answered that the only thing that was displayed to them was a "come back later" message.

One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users

Stevens Le Blond, Pere Manils, Abdelberi Chaabane, Mohamed Ali Kaafar, Claude Castelluccia, Arnaud Legout, and Walid Dabbous, I.N.R.I.A., France

Stevens briefly introduced how Tor and the whole onion routing concept works, and then started talking about two attacks they developed against users using BitTorrent trackers through Tor. These attacks nicely complement the keynote by Nick Mathewson, who acknowledged that the Tor team knows about a few privacy issues that involve Tor and peer-to-peer applications. The attacks developed by Stevens and his colleagues require the attacker to control one or more exit nodes in the Tor network. This type of node is really valuable, since it allows the attacker to see the outgoing traffic in the clear, although without any notion about the IP that generated it. The first attack hijacks the list of BitTorrent peers returned by a tracker to include a host controlled by an attacker. Since 70% of the people using BitTorrent over Tor do that only for accessing the tracker, they will connect to the malicious peer directly, disclosing their real IP address. The authors were able to track 10,000 IP addresses using this technique.

The second attack exploits BitTorrent distributed hash tables, which are a way for users to hide which files they downloaded. DHTs work over UDP, which Tor does not support. Therefore, when a user tries to use them through Tor, he will fail, and connect directly to them, publishing his real IP address in the DHT. At this point, by looking at the BitTorrent subscription identifier stored in BitTorrent subscription to the central tracker, it is possible to deanonymize the user who passed through the rogue Tor exit node. Stevens also showed how, after having assigned a real IP to a BitTorrent connection passing through a Tor circuit, it is trivial to identify all the other connections by that host, since they will all pass through the same circuit at a certain time. This attack also allowed them to deanonymize not only BitTorrent traffic but other kinds of traffic such as HTTP. Stevens next showed the breakdown of the searched words on BitTorrent over the exit nodes controlled by them. The vast majority of the query strings were adult content–related.

Chris Kruegel asked how you could do anonymous file sharing over Tor. Stevens replied that this is a fundamental problem with Tor, and the only safe way to do file sharing would be within your own ISP's networks. Someone asked whether using Tor over BitTorrent would be incredibly slow. Stevens replied that the vast majority of the tracked users use Tor only for the tracker data, which is what is usually checked for copyright infringement, but do not use it for actually downloading files.

Studying Cyber-Criminals and Their Tools

Summarized by Rik Farrow (rik@usenix.org)

The Nuts and Bolts of a Forum Spam Automator

Youngsang Shin, Minaxi Gupta, and Steven Myers, Indiana University, Bloomington

Youngsang Shin explained how they had worked with a demo version of XRumer to see how it works. They also compared its features to other forum spam automators in their paper. XRumer can register with forums and automatically deals with the obstacles designed to prevent this. It can solve some classes of CAPTCHAs with built-in algorithms. XRumer can also set up Gmail accounts to be used for receiving an activation email, then interpret the email and send the proper response to activate a forum account. XRumer includes a macro language for customizing spam postings, and the commercial version includes a search tool for finding the appropriate forums to spam.

Youngsang pointed out that XRumer does behave in ways that make it easy to notice. Although it will use different User-Agent tags, XRumer always sends the same HTTP header (and uses HTTP/1.0), and always sends a cookie even if none is required. XRumer also sends a Host header with a hostname but without the port number usually found in IE headers. XRumer also adds a Proxy-Connection header, something that is not a standard and is rarely seen. XRumer is designed to work with proxies to hide the sender's IP address, and proxies typically change the Accept-Encoding header, enabling detection of proxies.

Someone asked if the posted links point to legitimate sites or to spam sites. Youngsang said that they could point to either. One link went to an Amazon profile that included the link being farmed. The same person asked if this was baked into the software; Youngsang said it was up to the forum spammer. Chris Kruegel asked if most links got to spam pages or were they used for search engine optimization? Youngsang said that they didn't get a good classifications of links.

The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns

Brett Stone-Gross, University of California, Santa Barbara and LastLine, Inc.; Thorsten Holz, Ruhr-University Bochum and LastLine, Inc.; Gianluca Stringhini, University of California, Santa Barbara; Giovanni Vigna, University of California, Santa Barbara, and LastLine, Inc.

Brett Stone-Gross explained that through their continuing efforts to investigate botnets, they located over 20 command-and-control (C&C) servers for the Cutwail botnet. They had previously established relationships with various hosting providers, and after shutting down these servers, they gained access to a subset of them. These included 13 servers, three development systems (with sources), 24 databases containing billions of email addresses, spam templates, and a 40-page user manual. They also uncovered statistics on the servers that covered 50–67% of the Cutwail botnet. During 26 days in 2010, Cutwail successfully delivered (SMTP servers accepted) 87.7 billion spam emails. During the entire period covered by the captured logs, 516,852,678,718 messages were accepted for delivery out of a total of 1,708,054,952,020 attempts.

Brett described the bot life cycle as a primary infection, via some form of spam, of either Pushdo or other malware loaders. The loaders contact the C&C servers and install the bot software. Over a period of days, the bot may be removed by anti-malware tools, or become less effective at delivering spam after being blacklisted. Cutwail's operators must continually collect new bots. Within an average of 18 hours, 90% of bots have been added to blackhole lists for spamming activity. Cutwail does not assign unique IDs to bots, but one day's logs showed that there were 121,336 bots online.

The operators rent out the parts of the botnet via a Web interface that helps the purchaser design spam templates, rent email lists, rent groups of bots, and check whether SpamAssassin will accept the email as real. The costs of email lists vary based on type of address and location. Groups of bots also vary in price, depending on whether they are "clean loads" (only infected with Cutwail), country of location, and whether they have been blacklisted yet. User account information for connecting to mail servers can also be rented.

The Cutwail C&C servers also maintained an archive of Spamdot.biz, a spammer's forum. As forum participants kick out grifters and new members must be vetted by trusted members, the authors believe that the information in the forum was accurate. For example, a spam campaign's cost ranged from $100 to $500 per million emails sent. New malware installations were sold for $300–$800 for 10,000. The authors estimate that the Cutwail operators have netted between $1.7 and $4.2 million since June 2009, depending on the level of bulk discounts given to big customers.

Nathaniel Husted wondered about the costs of new bots; Brett explained that bots need constant replacing. Stefan Savage asked what the cost premium was for an exclusive load. It costs five times as much for clean loads, but how could the operators tell the load was clean? Do the operators collect a share of the proceeds of spam campaigns? They have "affiliate programs," and the operators could receive as much as 40% of the profits on these campaigns. How could the operators be sure they were getting their share? "In the underground, everything relies on trust."

On the Effects of Registrar-level Intervention

He (Lonnie) Liu and Kirill Levchenko, University of California, San Diego; Márk Félegyházi, Budapest University of Technology and Economics and International Computer Science Institute; Christian Kreibich, University of California, Berkeley, and International Computer Science Institute; Gregor Maier, International Computer Science Institute; Geoffrey M. Voelker and Stefan Savage, University of California, San Diego

Lonnie described the point of their research as an effort to understand the effectiveness of registrar-level intervention on spam. Their method was to examine WHOIS information for newly registered domains, which they collected by comparing TLD root zones and using a commercial service.

They started with CNNIC, the registrar for China. In 2007–2008, registering a domain name with CNNIC was one yuan (about $.15). In December 2009, CNNIC changed the price to 69 yuan (about $10) and initiated a policy that required real documentation (photocopy of business and Chinese registrant licenses), with a one-week warning period. Immediately, spam domain registration with CNNIC dropped, and shifted to Russian domains five weeks later.

In their second investigation they looked at eNom, a registrar infamous for being difficult to work with, and LegitScript, an organization that identifies legitimate US Internet pharmacies. LegitScript was able to get eNom to place thousands of domains into *clientHold*, a state where the domain is removed from root zone files and cannot be transferred to another registrar.

In both cases, Lonnie claimed there was a noticeable effect: an increase in the costs of spamming because domains now cost more and through the loss of domains via clientHolds. Someone pointed out that over time the price of domains will converge. He wondered if policies that support takedowns (clientHolds) would also spread. Lonnie said that it was difficult to get people to agree globally on anything. Dan Geer pointed out that the number of top-level domains (TLDs) was growing without bounds. Rik Farrow wondered if this would have much effect if the number of registrars stays the same. Stefan Savage suggested that with an increase in the number of TLDs, and thus domains, the cost of new domains would be reduced. Tyler Moore asked if they saw any further displacements after the eNom action. Lonnie said that happened near the end of their research, so they didn't see anything.

Invited Talk

Summarized by Brett Stone-Gross (bstone@cs.ucsb.edu) and Rik Farrow (rik@usenix.org)

Complex Web of Search Engine Spam

Sasi Parthasarathy, Bing

Sasi Parthasarathy described the difficulties that Microsoft faces in combating search engine spam, which poses a significant threat to the quality of search results returned by Bing. Sasi presented two primary types of search engine spam: page-level and link-level. Page-level spam relies on a number of techniques to influence the search engine rank such as keyword stuffing, parked domains, hidden/invisible text, machine generated content, and social media spam. The primary objective of these attacks is to deceive a search engine into believing that the content of the page is relevant to popular search queries, when in fact the content is low quality or potentially malicious. Link-level spam exploits the fact that Web sites are ranked higher by search engines if other sites contain links to them. As a result, attackers create sites (known as link farms) that link to their own sites and contain no useful content. In addition, link exchange communities have formed to mutually exchange links in order to boost their search rank.

Sasi also presented several newer types of threats such as hijacked sites, scareware, and low-quality user-generated content, and upcoming threats such as "like" farms. Bing currently uses both manual and algorithmic approaches to combat these search engine spam threats.

Stevens Le Blond asked about cloaking mechanisms. Sasi said that advanced cloaking has to do with recognizing the range of your IP address. Rik Farrow asked about the type of search engine spam used by a well-known retailer during the 2010 Christmas season. Sasi said that the retailer had paid a company to purchase links, moving the retailer into the number one position for many searches. Stefan Savage wondered how long someone needs to have their pages lifted in the ratings to make it worthwhile, and Sasi said while it was a great point, they don't look at those numbers. Tyler Moore asked about link farms. Sasi responded that link farms can span hundreds of sites, giving instant authority to lots of links. Moore then asked if link farms can contain legitimate links, and Sasi said that they can, and that makes them much harder to block. Dan Geer asked how long it took before a new site gets noticed. Sasi said that you won't be instantly seen, as they are looking for authority on the Web and that takes time. They also are looking for value for the user.

Nick Mathewson asked if they see any anti-optimization, attempts to get sites banned by using bad links (like links for Viagra pointing to a site). Sasi said they can't tell this from other behavior and don't punish sites for it. Chris Kruegel wondered about a site that gets lots of authority because of negative comments with links directed to a site. Does Bing use semantic analysis? Sasi said they do not police the Web, but do honor DMCA requests. Stefan asked if they follow noindex in a robots.txt file with illegal sites, and Sasi said they honor these tags. Stefan followed up by asking if they have a specific malware crawler, and Sasi said they didn't.

Threat Measurement and Modeling

Summarized by He Liu (h8liu@cs.ucsd.edu)

Characterizing Internet Worm Infection Structure

Qian Wang, Florida International University; Zesheng Chen and Chao Chen, Indiana University—Purdue University Fort Wayne

Qian Wang and his advisers attempted to characterize the underlying topology of a infection tree formed by worm infection. In particular, they are interested in modeling the children count of infected nodes, and the generation that a node belongs to. In distinction to previous work, they focused on micro-level characteristics, trying to understand individual nodes. They used a sequential growth model, applying probabilistic modeling methods. The modeling results showed that most nodes do not have a large number of children, and half of the infected nodes have no children. In addition, a typical worm tree does not have a large average path length, while the distribution of the generation approximately follows a Poisson distribution with a bell-like shape probability distribution function (PDF). They verified their model with discrete time simulations.

Their work implies that if a defender has access to some number of nodes, say A, in a P2P connected botnet, they will also have access to 3A nodes: the node itself, the parent, and one child on average. A defender can reach more bots if he or she targets nodes with the largest number of children. However, future botnets can respond with a self-stopping mechanism as a countermeasure.

In the future, they will try to use fractals as a tool to model and analyze the structure.

Nathaniel Husted raised a question about the minimum number of nodes to start analyzing using fractals, and another person also wondered if fractals are appropriate for spam tree analyzing, because an Internet spam tree may lack uniform self-similarities. Giovanni Vigna said Internet worm modeling was a topic many years ago, questioning if there is a renaissance of worm modeling. Qian said they worked on the micro level, and the work has value to Conficker C-like botnets. Giovanni asked how to find the nodes with the maximum number of children. Qian answered that they estimated a worm infection sequence, and early infected hosts might have a large number of children. Giovanni further asked about the best tradeoff for self-stopping. Qian said it might not have a huge effect on worm expansion speed.

Why Mobile-to-Mobile Malware Won't Cause a Storm

Nathaniel Husted and Steven Myers, Indiana University

Nathanial Husted explained that mobile-to-mobile malware used to spread using protocols like Bluetooth on feature phones using Symbian OS. Shifting to smartphones today, WiFi becomes available. Unlike Bluetooth, WiFi is always visible, uses transparent management traffic, and has a greater range and higher speed. To understand how smartphone malware will propagate, the presenter and his colleagues modeled and simulated human mobile users in an area in Chicago, using the SEIR (Susceptible-Exposed-Infected-Recovered) model and UdelModels. They tried both parallel and serial infection styles, and used different exposure times, susceptibility, and broadcast radius. The results showed that exposure time does not have a major effect below 120 seconds, and that susceptibility plays a much bigger role than broadcast radius. The simulation implied that current US cities still do not have the density for epidemics, and epidemics require high susceptibility rates. They concluded that mobile-to-mobile epidemics are the least of our worries.

Christopher Kruegel pointed out that they varied many parameters, but not the population density. What happens if the user density increases? Nathaniel answered that it is an interesting question and worth looking at in the future; realistic city user data with high density would be helpful. Stefan Savage asked why other vulnerabilities were not considered. Nathaniel answered that this is the vector of their focus, while other vulnerabilities may have characteristics similar to traditional PC-based malware.

Inflight Modifications of Content: Who Are the Culprits?

Chao Zhang, Polytechnic Institute of NYU; Cheng Huang, Microsoft Research; Keith W. Ross, Polytechnic Institute of NYU; David A. Maltz and Jin Li, Microsoft Research

Chao Zhang presented work in which they found that nearly 2% of clients in the US are affected by inflight modifications, and 44 local DNS servers (LDNS) in nine ISPs redirect clients to malicious servers. They collected user data from 15 million unique clients, of which 4,437 were proxies. They determine if a page is modified by comparing two pages fetched through two servers; different contents may indicate that the proxy is malicious. They discovered four different types of modifications: modifying the search result links, modifying advertisement links, inserting JavaScript, and redirecting requests. In total, they found 349 rogue servers.

Afterwards, Chao talked about the root cause of these modified pages. Some of the cases resulted from compromised local domain name servers. In fact, 48 out of 108 LDNS, grouped by /24 IP prefix, were compromised, and they were also operated by a small number of ISPs. Further study showed that these ISPs indiscriminately redirect all clients to malicious servers. Since only the DNS level is affected, public DNS improves service availability.

Christopher Kruegel asked whether the presenter thought that these ISPs are knowingly involved. Chao said that they have not contacted any ISPs to confirm that. Christian Kreibich said that he had seen ISPs that only redirect search engine Web queries, and asked whether they have similar or related observations. Tyler Moore asked if it is common to see redirections to ad-related domains. Chao said it is common, and some of the malicious servers do not redirect every time. Sasi Parthasarathy asked if there are any particular patterns of Web requests that are affected, and Chao responded that this has not been examined yet. Giovanni Vigna asked how they collected user information with user privacy as a concern. Chao answered that it is collected through MSN Toolbar from users who agreed to share data with MS to improve performance.

New Threats and Challenges

Summarized by Rik Farrow (rik@usenix.org)

Application-Level Reconnaissance: Timing Channel Attacks Against Antivirus Software

Mohammed I. Al-Saleh and Jedidiah R. Crandall, University of New Mexico

Mohammed Al-Saleh described how the authors fingerprinted AV by using timing attacks. For example, if you unzip Code Red, it takes about 100 seconds before Symantec AV recognizes it and cleans up. In their research, they used ClamAV. ClamAV filters on file writes by generating a hash and searching for this hash in the signature database. By using different files, they can use the length of time before detection to determine the update level of the AV in use.

They used ActiveX to open and close a file and to sample CPU time to determine how busy the CPU was. With this they could determine with high accuracy if a particular signature was included in an AV database. Mohammed claimed this will work against algorithmic and heuristic defenses as well.

Ben April (Trend Micro) wondered about the applications for their research. Mohammed said that they could fingerprint both the type of AV and the specific signature-update level. Ben then wondered if they had considered cloud-based AV, and Mohammed said they hadn't considered it. Engin Kirda wondered if other AV scanners would have the same issues, and Mohammed said they focused on ClamAV as they had the source for it, but that they thought other AV scanners would have the same issues. Ben April asked about countermeasures, and Mohammed said they should do "the usual": add in delays. With a scan taking 100 seconds already, a little more delay would hardly be noticed.

Reconstructing Hash Reversal based Proof of Work Schemes

Jeff Green, Joshua Juen, Omid Fatemieh, Ravinder Shankesi, Dong Jin, and Carl A. Gunter, University of Illinois at Urbana-Champaign

Jeff Green explained that some servers use proof-of-work protocols as resource management. These protocols require that the clients are willing to perform some amount of work. There are two types of protocols: task-based, and task plus time token–based. The typical task requires the client to perform hash reversal puzzles that are easy for the server and difficult for the client. The server provides bounds and the client must brute-force an answer.

In their research, they used an older GPU that worked with CUDA, an API for running programs on GPUs. Jeff said that even their several-year-old graphics processor was 600 times faster than last year's high-end CPU at solving these puzzles. Their suggested solution requires that servers track clients and vary the amount of time required by the puzzle based on client responses.

If they are tracking clients, Chris Kruegel wondered, wouldn't it be simpler to rate-limit clients? It would, but only if you wanted to predetermine how much service everyone gets. Chris asked if this took anything special, and Jeff answered that it only required a CUDA-enabled graphics card and an application to use it. Dan Geer remarked that work-based proofs are now a non-starter, and Jeff agreed.

Andbot: Towards Advanced Mobile Botnets

Cui Xiang, Fang Binxing, Yin Lihua, Liu Xiaoyi, and Zang Tianning, Institute of Computing Technology, Chinese Academy of Sciences

Guong Liung of Boston University presented this paper, as the authors could not attend because of visa problems. He identified himself as a friend of a friend of the authors. He attempted to explain the paper, which he suggested was better if just read.

The authors expect that smartphones will soon be used in botnets. But smartphones have specific issues, such as limits on power, generating network traffic which may reveal the bot, and the absence of a public IP address. The authors suggest several strategies to make Andbot low cost, resilient, and stealthy. For example, they use signals detected in microblogging sites (e.g., Twitter) to find JPEG images that have botnet commands embedded in them.

Sven Dietrich (Stevens Institute) wondered how this could work, as mobile providers tend to compress images and this would likely scramble any commands. He also pointed out that there are already command-and-control botnets that use Twitter. Goung again suggested reading the paper.

Hot-ICE '11: Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services

Boston, MA
March 29, 2011

Cloud Resource Management

Summarized by Theophilus Benson (tbenson@cs.wisc.edu)

Dynamic Resource Allocation for Spot Markets in Clouds

Qi Zhang, Eren Gürses, and Raouf Boutaba, University of Waterloo; Jin Xiao, Pohang University of Science and Technology

Current cloud computing offers on-demand computing at fixed prices; however, these fixed pricing models lead to underutilization of the cloud. Some cloud providers have begun to adopt spot-pricing models: a spot market for each class of virtual machines. The number of physical hosts allocated to each market is fixed. Qi Zhang claims that static allocation of physical resources to each market restricts the flexibility of each market and reduces the maximum achievable revenue. To this end, Qi proposed a technique to dynamically adjust the supply of physical resources to reach market, with the sole goal of maximizing the amount of revenue received by the cloud provider.

There are two challenges to tackle in developing this technique. First, accurate prediction of the demand for each market. This is accomplished by using an auto-regressive (AR) model to predict future demand based on prior demand. Second, developing an algorithm to utilize the demand to allocate the resources to each market. The framework, proposed by Qi, consists of three components: a market analyzer, a capacity planner, and a virtual machine scheduler. The market analyzer uses the AR model to predict demand. The capacity planner uses the observation that the demand curve is a non-increasing function of spot price to determine the spot prices for the different markets based on the predicted demand. Given the spot price, the capacity planner optimizes for the product of price and quantity. The authors show that this concave optimization problem can be reduced to the multiple knapsack problem. Finally, the scheduler launches a virtual machine and adjusts the supply of resources to each market based on the output of the capacity planner.

To evaluate the framework, Qi conducted experiments on CloudSim by modifying the scheduler to incorporate his framework. In comparing static allocation of physical resources to dynamic allocation the authors showed that their framework improves the revenue of the cloud provider.

How does granularity of decision-making affect total price? Currently, the granularity of decision-making isn't evaluated, but it will be in the future. How does the constraint of fixing the size of the pools based on predicted demand affect the solution? The current system is designed to improve prices via increased flexibility. Future work will examine this.

On the Benefit of Virtualization: Strategies for Flexible Server Allocation

Dushyant Arora, Anja Feldmann, Gregor Schaffrath, and Stefan Schmid, Deutsche Telekom Laboratories/TU Berlin, Germany

Dushyant focused on the migration of services between different locations to improve users' QoS while reducing the economic cost associated with migrating the servers of the service. This work is prompted by the fact that users are mobile and over time their request patterns change. For example, countries will have different access patterns due to different time zones, and users' daily commute will cause them to access resources from various locations. The goal of this work is to provide worst-case guarantees without knowledge of future demand. To provide these guarantees, the authors chose to employ an online algorithm.

Their algorithm divides time up into epochs and at the end of each epoch it uses the request patterns seen to approximate where future request patterns will originate from. With this prediction, a server is picked and moved towards the center of gravity of its request patterns.

In evaluating the effectiveness of this algorithm, Dushyant compared his algorithm with an optimal offline algorithm that makes choices using actual future demands and with a static online algorithm. The algorithms are compared under two different patterns: time-of-day requests originating from different countries, and commuter requests originating from different locations. The authors show that their algorithm is beneficial in times of moderate dynamics, as it is able to exploit the request patterns.

Is each epoch static or migration-time static? Epochs are defined based on the latency between clients and servers. Do you assume a constant amount of time to get the monitoring information? Yes, there is a centralized infrastructure that gathers all the information. Currently, constant time is assumed. Certain organizations include policy such as middlebox interposition—how do you account for such policies? Currently, they do not account for such policies, as these policies complicate the algorithm; however, they address other policies. Furthermore, they don't assume intimate knowledge of provider topology.

Cost-Aware Live Migration of Services in the Cloud

David Breitgand, IBM, Haifa Research Lab; Gilad Kutiel and Danny Raz, Technion, Israel

Live migration consists of two phases: the pre-copy phase, when initial memory pages are copied to the new location while the virtual machine is running, and the copy phase, when the virtual machine is stopped and the un-copied or dirty pages are copied over. David claims that the pre-copy phase is equally important as the copy phase, as the processing power and bandwidth used during the pre-copy phase impacts the response time of the applications and could lead to SLA violations. David assumes that live migration is performed in band—the same network that is used to service requests is used for migration. In choosing the amount of bandwidth to reserve for migration, the operator faces a tradeoff, since reserving small bandwidth leads to longer migration, while reserving large bandwidth will reduce migration time but degrade the service.

The cost of migration is, then, the cost of the pre-copy phase and the cost of the copy phase, both of which are a function of the bandwidth reserved for migration. The authors propose an optimization algorithm for dynamically deciding the amount of bandwidth to reserve for the pre-copy phase, based on the number of pages to copy and the amount of time elapsed.

To evaluate their optimization, the authors use a trace-driven simulation and compared this algorithm to the default algorithm in Xen. Their experiments show that the framework is able to better adjust itself according to the available bandwidth and cost function than Xen's default algorithm.

To what extent is this solution affected by the assumption that bandwidth is a bottleneck because the same link is used for both migration and service requests? The solution applies in situations where bandwidth isn't a bottleneck. Do they assume time to migrate is not dependent on workload, as workload affects time of migration because workload may largely affect memory? The model contains provisions for determining the probability of modifying the memory pages and is able to simulate workloads with varying impact on the memory. Do they measure downtime from the server, when the server is turned off and moved, or from the client, when the user perceives disruption in service? From the client's perspective.

Server Operational Cost Optimization for Cloud Computing Service Providers over a Time Horizon

Haiyang Qian and Deep Medhi, University of Missouri—Kansas City

Haiyang claimed that servers account for a large portion of the operational costs of maintaining a datacenter. Furthermore, he claimed that the operational cost for servers can be broken down into energy consumed by the server and the cost of replacing hard disks. The authors' goal is to determine the optimal frequency and number of servers to use to satisfy demand for the various services while minimizing the cost of running the datacenter. To do this, the authors formulate an optimization problem that minimizes the power consumption, the cost of turning on and off servers, while ensuring the constraint that each server can only run at one frequency and that the datacenter must satisfy all the users' demands.

Haiyang showed that the optimization problem is quadratic in nature and very time-consuming, and thus he developed optimizations to make the problem linear. In evaluating his formulation, he compared the cost of maintaining a datacenter under different server allocation strategies. Haiyang observed that adjusting CPU frequency provided significant cost improvements over running the servers at max CPU frequency. Haiyang also noted that the granularity of time over which calculations are made impacts the cost; larger time granularity costs less.

Do they assume the load can be moved to other locations in the datacenter, and do different servers have access to the same resources? Yes, they do assume migration; however, they didn't look at other locations. Do people actually change the frequency of the CPU? Yes, certain operating systems come with sufficient functionality to change the CPU frequency. What is the impact of CPU throttling on the performance of the application, and how does throttling affect the ability of an application to meet its SLA? They assume all service level agreements are satisfied.

Service Management

Summarized by Theophilus Benson (tbenson@cs.wisc.edu)

Automated Incident Management for a Platform-as-a-Service Cloud

Soumitra (Ronnie) Sarkar, Ruchi Mahindru, Rafah A. Hosn, Norbert Vogl, and HariGovind V. Ramasamy, IBM T.J. Watson Research Center

Cloud providers offer tenants on-demand access to resources at low cost. To maintain this low cost while remaining competitive, providers are forced to reduce operational expenses by limiting real-time problem determination and eliminating SLA guarantees. The authors propose a framework to further reduce the amount of problem determination by using simplified automated corrective actions. The framework performs integrated monitoring for virtualization systems across all layers: networks, physical, hypervisor, and OS. The framework responds to issues by taking the automated action for some incidents and ticket creation for other items. In a small number of scenarios both actions are taken; this is to inform system admins of actions via ticket creation.

The framework is a centralized system which aggregates information from multiple sources, including agents installed on the physical hosts. In designing this system, the authors had the following design requirements: scalability, persistence, and fault tolerance. To achieve persistence, data is stored to persistent cloud storage. For scalability, each server is modeled using a simple finite state machine (FSM). Using the current state of a server in the FSM, the framework is able to determine the actions to perform. By limiting the amount of state required to make progress, the framework is able to easily recover fault.

Automated Incident Management System (AIMS) was deployed in a cloud, and during the deployment a few interesting additions were made to the framework: (1) to ensure that bugs in the system do not take down the entire datacenter, a circuit breaker was added to limit the number of elements to take action on; (2) to ensure that the system remains quiet during a planned outage, they introduced the ability of a system freeze for some fraction of time; (3) the authors included several filtering rules to reduce false positives; (4) they included self-healing properties to allow the system to write temporary logs that can be used by the framework at a future date to support future actions; and (5) due to the asynchronous nature and lag between diagnosis and corrective action, work-flow validation was added to ensure that corrective action was still required before performing it.

Does the framework understand the relationship between different resources used by a cloud user to create a service? If so, is there a service for this level of information? No, the system is currently geared towards provider-side management. Is there a state machine for each virtual machine? Yes, there is one for each virtual machine. However, the system only needs to track the current state. If the current state of a server is tracked by the framework and can be determined by querying the server, which is the authoritative source of information? The server is the authoritative source of information, and before actions are performed on the server the system reevaluates the state of the server.

QoSaaS: Quality of Service as a Service

Ye Wang, Yale University; Cheng Huang, Jin Li, and Philip A. Chou, Microsoft Research; Y. Richard Yang, Yale University

VoIP is becoming an integral part of the enterprise network ecosystem. In this talk, Ye Wang provided a system for aggregating measurements from different locations. The system uses these measurements to estimate the performance of different applications and to determine problems introduced by individual devices. Ye proposes that VoIP applications query the system to determine how to adapt to issues in the network and that operators can use the system to diagnose QoS degradation.

This work is prompted by problems encountered by users of the Microsoft Lync VoIP system, used by 7 million users. Debugging performance-related problems is difficult, as it requires information about how each system component affects the end-to-end quality, information that is currently missing.

To debug QoS issues in VoIP calls, the system collects data from different clients and network entities and uses an inference engine at a central location to determine problems. The system models the following entities: the network, the Microsoft media servers, and the VoIP clients. The system models communication between two VoIP clients as a set of media lines encapsulating the entities involved in the VoIP class. Inference is performed by running a maximum likelihood algorithm on entities in the media lines. When performing inference, information from prior calls is used; however, unlike other inference systems, the system takes into account session durations and weighs the information used in the inference according to length of the session. In running the inference engine on the data collected from the deployed Microsoft Lync system, Ye observed that most problems are networking problems and most entities do not drop packets.

Is this data already being reported by Microsoft? If so, what is this system adding? It is currently being collected by Microsoft agents, and the system adds aggregation and inference algorithms.

KnowOps: Towards an Embedded Knowledge Base for Network Management and Operations

Xu Chen and Yun Mao, AT&T Labs—Research; Z. Morley Mao, University of Michigan—Ann Arbor; Jacobus Van der Merwe, AT&T Labs—Research

Network management comprises all activities required to keep the network healthy while delivering a certain SLA. These activities consist of configuration management, fault management, and planned maintenance, among others. The network management systems required for each activity are created and used by very different teams. The domain knowledge used in creating these systems and the documentation for using these systems are all encoded manually in human-readable documents.

Xu argues that a machine-readable knowledge base is required to express the information from the different domains and ultimately provide easy access to this information. COOLAID provides a framework for capturing this knowledge as rules that can be run against a database containing information on the network systems. The rules and abstractions made by COOLAID are machine readable. Working backwards from COOLAID, with the rules stating how different information interacts, the system can extract event correlation and determine which items to monitor and how frequently to monitor them. Furthermore, these rules can prove useful in determining the holistic impact of automated procedures and can be used to determine how and when to schedule tasks.

Can they automate all management tasks? It may not be profitable to automate all tasks, and some tasks may require some manual interaction due to diversity. Furthermore, guard rules may be required to ensure that the amount of evil is limited. Although the system requires experts to determine rules currently, can they create rules from the data itself? While it may be possible to mine the rules from the data, such an approach assumes that all data is correct and that data is comprehensive; however, there may be certain rules that will be absent from the data. Can the vendors, ISPs, and operators work together to determine how to standardize and express information? Yes, they can, and KnowOps is a step towards such standardization.

Network Management

Summarized by David Shue (dshue@cs.princeton.edu)

Using Hierarchical Change Mining to Manage Network Security Policy Evolution

Gabriel A. Weaver, Nick Foti, Sergey Bratus, Dan Rockmore, and Sean W. Smith, Dartmouth College

Security goals are often expressed in policies that ultimately affect system behavior. If you don't change the policy, the system becomes increasingly vulnerable to exploits. If you do change policy, implementation and change tracking complexity can explode. To combat the potentially unbounded complexity induced by evolving polices, Gabriel Weaver presented a mechanism to detect and determine the changes in hierarchically structured documents to track evolution and change, and ultimately help policy makers make better decisions.

Existing techniques such as changelogs (e.g., Adobe redlining, Word track changes) are insufficient for pinpointing the scope and extent of changes. By parsing the document into a hierarchy, as per a language grammar, Gabe's method can easily distinguish between large and small changes. At the core of the technique are two distance metrics used to compare document trees: Tree Edit Distance (differences in subtree structure) and Word Edit Distance (differences within leaf text blocks). In comparing the output of the technique to a manually documented changelog, Gabriel found that 9 out of 178 reported changes were never made in the revised document, demonstrating the utility of the approach for edit verification.

The general technique can be applied to any well-structured document type, including router configuration (e.g., Cisco IOS), to help analyze and track the gap between security policy and implementation evolution. Current techniques based on RANCID do not leverage the structure of IOS and spot textual differences that are semantically equivalent. Change mining, on the other hand, analyzes the logical structure, filtering out order-invariant changes. Moreover, by constructing a fully diffed parse tree, the technique enables multi-level change querying based on tree prefixes, i.e., /root/interface for all interface-related changes. A user can drill down further to explore structural differences at finer granularity: interface block -> line card type -> specific interface.

Anees Shaikh asked whether the technique could rank changes based on impact or importance. Gabe said that is future work. Kobus Van der Merwe asked what the user actually queries in the system. The system parses the documents and the resultant paths are then loaded into a DB for querying. Ye Wang asked whether change mining is limited to just security policy. The technique is applicable to any well-structured text that exhibits hierarchical structure. Morley Mao asked whether the technique can detect invariance. Yes, by modeling the parse tree, which can filter out irrelevant textual changes.

Towards Automated Identification of Security Zone Classification in Enterprise Networks

HariGovind V. Ramasamy, IBM Research; Cheng-Lin Tsao, Georgia Tech; Birgit Pfitzmann and Nikolai Joukov, IBM Research; James W. Murray, IBM

Enterprise networks are partitioned into multiple zones of criticality where devices of similar security requirement are placed into the same zone (e.g., Internet, intranet, DMZ with firewalls between, etc.). Although the number of classifications (colors) may be small, there are often a large number of zones per color created to support diverse organizational and application isolation needs. Unfortunately, enterprise network administrators must manually track the zones in ad hoc documentation, which can be tedious and error-prone. To combat this problem Hari introduced BlueGates, a system that examines application-specific, flow-level access control to facilitate automatic security zone classification.

BlueGates relies on a defined security policy matrix that prescribes the allowed traffic between security zones—e.g., only port 80, ssh, all traffic, authentication required, etc.—as the canonical metric for classification. By gathering the actual flows allowed in the network using existing monitoring tools and comparing them against flows permitted by policy, the system can then infer zone colors. With the policy matrix, the set of hosts to analyze, and a small set of known classifications as input, the system generates a full delineation of network zones, their colors, and the interconnections between zones. Indeterminate hosts are assigned all colors. If all colors are infeasible for a host, then the system flags a potential conflict in policy and configuration. Future work on BlueGates includes loosening the assumption that the network configuration fully complies with security policy and evaluating the system on a large-scale infrastructure.

Kobus Van der Merwe noted that the system assumes that the security policy exists and can be properly represented as input. Hari said that this is true, someone must glean the information and perform data entry, but polices change on an infrequent basis compared to network configuration. Kobus followed up by asking how they know that the model is correct. Verification of the model and output with document authors and operators is an iterative process. What if configuration does not comply with the specification? While the solution may be imperfect given the assumptions, any useful though potentially flawed tool is helpful. Would reachability analysis between networks arrive at the same conclusions but in a simpler fashion? Simple flow-level reachability ignores per-application (protocol port) ACL and security requirements. Theo Benson asked whether switch configurations could provide similar information without the need to intrusively probe the hosts. Accessing switches and firewalls often raises more alarms than probing the hosts themselves.

Simplifying Manageability, Scalability and Host Mobility in Large-Scale Enterprise Networks using VEIL-click

Sourabh Jain and Zhi-Li Zhang, University of Minnesota—Twin Cities

With the increasing degree of mobility (mobile devices, VM migration) comes a new host of networking challenges: scalability, mobility, availability, manageability, etc. Existing layer 2 (L2) techniques support plug and play but at the expense of flat routing and the reliance on broadcast protocols. Layer 3 (L3) routing is scalable but not amenable to mobility and still requires control-plane flooding. Moreover, simple address changes require significant modification to network state (firewalls, DHCP servers, etc.). To surmount these challenges, Sourabh presented VEIL-click, an L2 (Ethernet) network architecture with built-in support for scalable mobility and fast-failure rerouting.

In VEIL, IP addresses become persistent unique identifiers that remain immutable across network changes. To support topology-aware location-based routing, VEIL assigns each NIC a Virtual ID (L2 MAC) that encodes the network location. Routing on the VID is transparent to end hosts and compatible with existing Ethernet switches. On startup, VEIL-enabled switches in the network communicate with a centralized controller that assigns topology-dependent VIDs to each switch. Each VEIL switch then acts as an access gateway, assigning VIDs to each host (32-bit switch VID and 16-bit hostID) and mapping persistent IPs to VIDs by intercepting ARP requests.

Together, the VEIL switches form a DHT to distribute the IP-to-VID registry and forward traffic based on VID. When an ARP request arrives at a VEIL switch, it is resolved via unicast to the appropriate DHT root. A translation rule at the ingress VEIL switch maps the source MAC to VID for return traffic, and a corresponding rule at the egress VEIL switch maps the VID to actual MAC for final delivery to the destination NIC. To support mobility, when a host moves to a different point in the network the access VEIL switch assigns a new VID and updates the IP-VID mapping at the corresponding DHT node. An update instructs the VEIL switch responsible for the old VID to forward packets to the new VID mapping. Sourabh has built a VEIL prototype based on the Click modular router and showed a graph demonstrating a 1–2 second TCP flow disruption for mobility between wireless networks.

Morley Mao asked what the impact would be for real applications (VoIP) and not just at the flow level. Sourabh said that this is future work. Ye Wang asked about scalability for a large number of clients and switches. One could create a hierarchy of VEIL networks based on L3 routing. What is the performance impact of modifying Ethernet headers? The overhead is less than TRILL. Chang Kim asked why not rewrite IP addresses. L2 name resolution already exists (ARP) and the different sizes of IPv4/v6 would also cause issues for VID-based routing. What is the increased risk of address spoofing due to the extra indirection? Mitigating the attack would require a means of verifying the owner of the IP address, such as a PKI-based certificate mechanism.

Mini-panel

Morley Mao asked if there is anything at odds between security and performance, or are there any improvements vendors can make to facilitate management?

Hari Ramasamy said it would be useful to have vendor-provided tools for constructing an end-to-end view of the network. Someone in the audience remarked that the ConfigChecker research project does just this, providing a global end-to-end perspective of the network. Hari also noted that the teams looking at performance and security are different and often have different goals that may conflict.

Gabriel Weaver posited that finding out the issues that vendors have can inform and direct new research work.

Someone asked a VEIL-specific question: Can there be multiple VIDs for an IP address? Sourabh Jain answered that there is one IP per NIC and only one VID per IP.

Someone wondered what happens for multi-homed devices and the scalability to determine liveness. This resulted in a discussion that can be summarized as "liveness mechanism doesn't scale (n^2)."

Morley Mao had a question to the ISPs: What are the insights into problems from homegrown solutions—that is, what are the underlying principles you've gleaned? Kobus Van der Merwe answered that you need to think about the effect on services, not just the lower layers of the network stack.

Morley Mao asked about the challenges in terms of validation with real data. Hari Ramasamy replied that firewall configs are not taken from the production firewall but rely on a snapshot instead, which could lead to stale analysis. These are inherent limitations of the internal processes. Vendor tools could help by anonymizing configs. Someone else wondered if trouble tickets could be used as a source of information. Hari replied that they could be a useful source of info.

Gabriel Weaver asked about analyzing security changelogs. Hari said that this requires a loop between researcher and practitioner to obtain data and verify results and to determine the usefulness of the results.

Someone asked Gabriel Weaver whether you can have a subtree exist and assign a metric to state the level of interest/non-interest. Gabriel replied that you can determine what to evaluate/ignore. The technique can parse human-readable and machine-readable specifications: RFCs, configuration, etc.

Datacenter Networking

Summarized by David Shue (dshue@cs.princeton.edu)

Enabling Flow-level Latency Measurements across Routers in Datacenters

Parmjeet Singh, Myungjin Lee, Sagar Kumar, and Ramana Rao Kompella, Purdue University

Many of the online services populating datacenters today provide latency-sensitive customer-facing applications.

Troubleshooting latency anomalies at the end host requires simple local measurements, but flow-level latency measurements in the network fabric have proven difficult. Previous work on Reference Latency Interpolation (RLI) can perform fine-grained measurements but requires that every router be RLI-enabled. In this work, Myungjin Lee proposed a lighter-weight approach to router-level flow-latency measurement: RLI across Routers, or RLIR, which only requires every other router to support RLI by trading off latency localization granularity (from link to path segment) for cost and ease of deployment.

In the RLI architecture, router pairs send reference probe packets with embedded timestamps between an ingress interface on one router and an egress interface on the other to determine link latency. By exploiting delay locality, where closely spaced packets experience similar queueing delay, the routers can linearly interpolate between reference packet latencies to estimate nearby regular packet latencies. In a partial deployment scenario, ECMP may cause packets to take different routes between routers, violating the delay locality principle for the measured paths. Cross-traffic along the path also complicates the reference packet sampling rate estimation.

To address these issues, RLIR senders associate with all intermediate RLIR routers along its flow paths, and RLIR receivers examine the source IP prefix to distinguish reference from regular packets for proper flow latency correlation in the upstream direction (top-of-rack to core router). Additionally, an RLIR receiver may also need to reverse compute the ECMP splitting or rely on packet marking to determine the full flow path in the downstream direction (core to top-of-rack). For sampling rate estimation, RLIR assumes worst case utilization along the path segment and rate-limits reference packets accordingly. Evaluated in simulation using an OC192 link trace, RLIR was able to achieve less than 10% relative error in mean estimates at 93% utilization for 70% of the flows and similar relative errors in standard deviation estimates for 90% of the flows. For bursty traffic, RLIR was able to capture the differences in mean latencies to aid in identifying and isolating adversarial cross-traffic.

Jeff Mogul wondered how far you can push deployment sparseness. Myungjin said it is a natural tradeoff between cost and localization granularity. Chang Kim asked why the technique requires hardware assistance. Myungjin replied that the reference packet injection with router timestamps and linear interpolation estimates at line speed require fast-path hardware support.

OpenFlow-Based Server Load Balancing Gone Wild

Richard Wang, Dana Butnariu, and Jennifer Rexford, Princeton University

Effective load balancing for datacenter-based replicated services should handle aggregate request volume, scale out across available replicas, and provide a programmable interface, while remaining cost-effective. Existing software load-balancers are flexible, but are throughput limited. Dedicated hardware load balancers lose programmability and can be complex and expensive. Enter OpenFlow, which combines a simple distributed hardware data plane with a flexible centralized software control platform. In this talk, Richard Wang introduced a scalable and flexible load-balancing solution based on OpenFlow wildcard rules.

Although OpenFlow supports a large number of exact match rules, rule-per-flow load-balancing is still bottlenecked by the microflow rule table size and controller throughput, since the controller must act on the first packet of every flow request. On the other hand, although smaller in number, OpenFlow wildcard rules can match on src IP prefixes to bulk distribute incoming flows across server replicas without any additional controller interaction. Assuming a uniform distribution of requests across the client (source) IP address space, wildcard rules can be used to split flows across replicas according to a (power of two) weighted distribution. Additional rule savings can be achieved by combining rules with a common source IP prefix to achieve a particular weighting for a given replica.

Since wildcard rules direct flows to replicas in aggregate, the controller must intercede whenever the wildcard rule set changes, to preserve flow affinity. When a new wildcard rule overlaps with an existing rule, the controller enters a transition period before installing the new rule. During this interval, the old rule directs packets to the controller, which then installs microflow rules to direct existing flow packets (non-SYN) to their original replica, while new flows (SYN) are directed to the new replica. After the transition period expires, the controller installs the new wildcard rule. Any existing microflow rules will expire once the flows terminate, due to the OpenFlow rule idle timeout. Future work includes supporting a broader range of client IP distributions and scaling the transition period reliance on microflow rule computation with multiple controllers.

Someone asked how the duration of the transition period is determined and how the controller handles long-lived flows. Richard answered that the default is 60 seconds, which is a tradeoff between flow affinity and system responsiveness. What is the scale difference between wildcard and exact match rules tables? Richard said 36k exact match vs. 100 wildcard (TCAM-based). Why can't the wildcard rules match on the least significant IP bits for better entropy? Currently, OpenFlow only supports prefix wildcards. Would the OpenFlow 1.1 spec which supports hash-based rule any-cast obviate the need for wildcard matching? The ensuing discussion concluded that it may be a while before hardware vendors support the 1.1 spec, and consistent hashing based on the typical flow 5-tuple could be used to distribute flows across the replica set.

Kobus Van der Merwe wondered how much flexibility is truly necessary, given that the set of functionality provided seemed standard. Richard replied that controller programmability enables feature evolution, though a more detailed feature set and performance comparison to F5 and Zeus load-balancers that are available as VMs for free evaluation may be useful.

Online Measurement of Large Traffic Aggregates on Commodity Switches

Lavanya Jose, Minlan Yu, and Jennifer Rexford, Princeton University

Monitoring and identifying large traffic aggregates improves network visibility and fosters a deeper understanding of the latent traffic structure: that is, heavy users, popular Web sets, anomalous DDoS traffic patterns, etc. In this work, Lavanya Jose presented an OpenFlow-based approach for online aggregate traffic measurement, with a particular focus on identifying hierarchical heavy-hitter traffic, using a cheaper and simpler commodity platform compared to previous approaches.

Coarse-grained aggregate views may overlook individual culprits, while fine-grained monitoring could miss the heavy subnet forest for the mid-weight IP trees. Hierarchical heavy hitters (HHH) use a multi-resolution approach to identify both individual and aggregate traffic sets of interest. HHH forms a monitoring trie (prefix tree) based on subnet prefixes. In this prefix tree, any leaf node that consumes more than a fraction T of its link capacity is marked as a heavy hitter. To filter redundant information, parent nodes are only considered as heavy hitters if the total link utilization of non-HHH children still exceeds T. Prior work on HHH (Autofocus), which analyzed offline packet traces, was accurate but slow. Building custom hardware for online monitoring would be prohibitively expensive. Instead, by leveraging OpenFlow's programmable data plane monitoring facilities with wildcard rules provides a more feasible monitoring solution.

OpenFlow wildcard rules make a good match with HHH, despite the limited pool of TCAM-powered rules. Since OpenFlow rules respect a strict priority ordering, high and low resolution rules can form an overlapping monitoring hierarchy based on set-differences. The controller uses at most 2/T rules to monitor and detect heavy hitters by iteratively adjusting wildcard rules over each monitoring interval M. If a prefix monitoring rule detects that a node exceeds T, the controller adds additional rules to drill down and monitor the children. Otherwise, if the utilization drops below T, the node monitoring rules are dropped. Any leftover rules are used to monitor HHH parent nodes that are close to the T threshold. In a simulated evaluation on a packet trace from CAIDA (400K packets/s) with a monitoring interval spanning 1–60 seconds, the framework was able to identify 88–94% of the T = 10% HHH with only 20 rules.

Someone asked about the complexity of the algorithm. Lavanya replied that the technique is linear in the number of rules used, which corresponds to the number of HHHs (2/T), not the size of the data. How does the algorithm bootstrap the monitoring? It starts with the largest (root) prefix first. What is the timescale of convergence? About 10–20 intervals.

Topology Switching for Datacenter Networks

Kevin C. Webb, Alex C. Snoeren, and Kenneth Yocum, University of California, San Diego

Large-scale datacenter networks house multiple tenants with a diverse range of applications. While most current techniques address network contention by adding capacity, applications may have differing and conflicting needs: MapReduce wants high bandwidth and MPI desires low latency. A simple MPI benchmark run with a background Hadoop job caused latency to increase 20% in the network. In this talk, Kevin Webb presented a system that constructs a virtual network topology according to application-specified properties: bandwidth, redundancy, latency, and isolation using the OpenFlow platform.

Applications submit routing tasks which describe the set of communicating end hosts, logical topology, and an allocation algorithm for the desired properties. The allocator specifies an objective function to evaluate prospective allocations, an allocation algorithm, and an annotation mechanism to mark and filter resources. Using these pieces, the allocator translates the application demands into allocations and routes in the network given the physical network view exported by a server-based topology. Once allocations are determined, the topology server annotates the network graph to mark the consumed resources and installs the routes into the network.

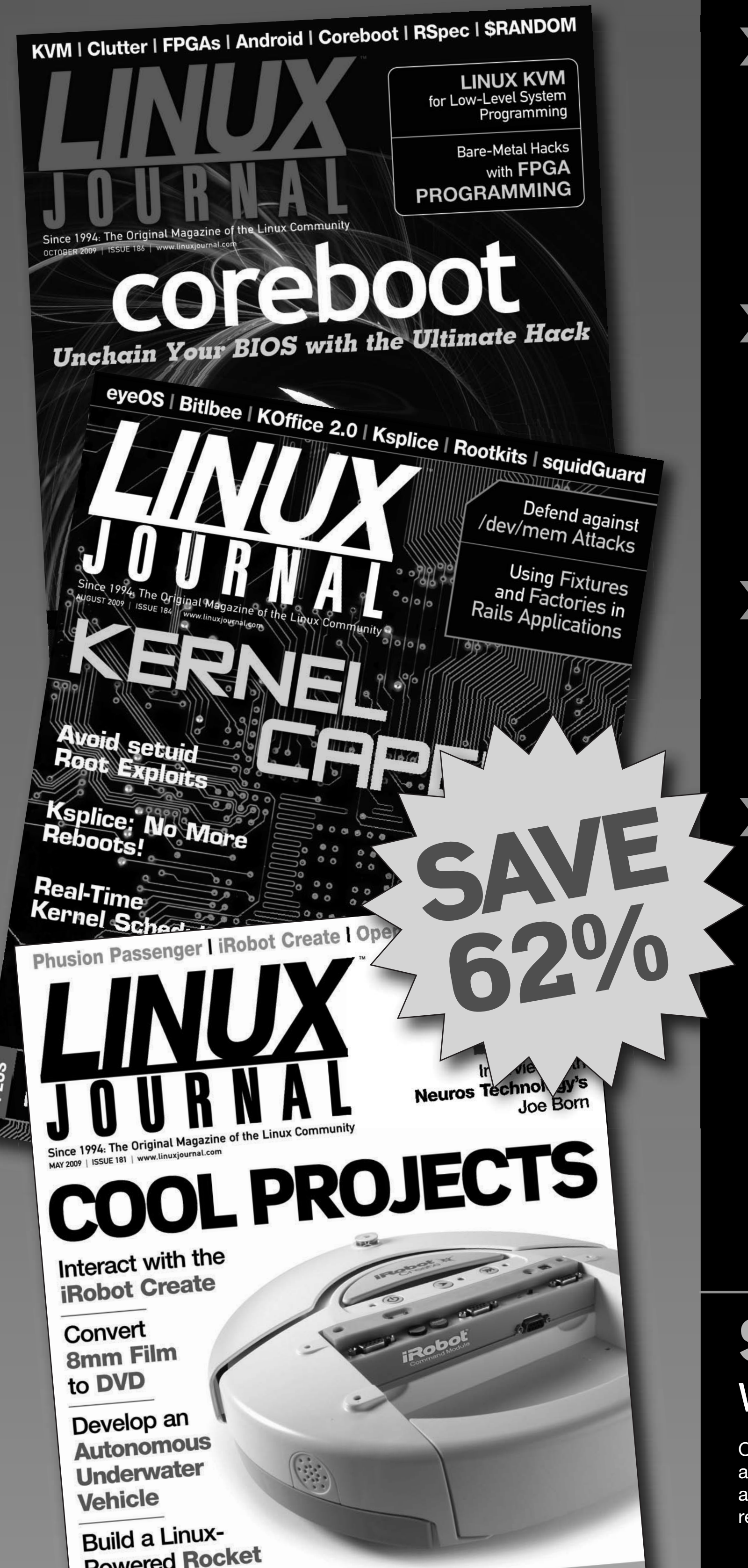
Kevin described three allocators implemented in the system: bandwidth driven, which finds the least loaded paths by building a max spanning tree over the remaining network capacity; resiliency, which searches the path space to find N disjoint paths between every host pair; and k-isolation, which builds a spanning tree with links used by, at most, k other tasks. Simulations comparing the system to a full ECMP solution for resiliency and bandwidth and VLANs isolation show that the system can achieve resiliency and full isolation, where the ECMP/VLAN approach overachieved on resiliency but provided a smaller degree of isolation. Future work will focus on building the architecture and quantifying the degree of inter-application interference.

Kobus Van der Merwe inquired about the timescale of decision-making. Kevin said that the system responds to on-demand requests and makes decisions as quickly as it can. Chang Kim asked whether capacities were part of the application specification. Rate-limiting based on capacities would be helpful but is not yet included. Kobus asked why the system assumed a fixed set of servers and a malleable network when often the reality is the opposite for virtualized environments. The system should probably handle both VM placement and network resource allocation. Another questioner remarked that the allocation decisions may be good for isolation but not utilization, and multipath may work better. Kevin agreed that the bandwidth allocation is not optimal and will look into multipath in future work.

Mini panel

Someone asked if there is a standards body for defining relationships between multiple VMs, including their network requirements. Kobus Van der Merwe wondered if OpenFlow happens to be the best hammer available for implementation or is it the right switch abstraction? Richard Wang replied that OpenFlow simplifies control and implementation of custom switch behavior and allows substantial customization. Both Amazon and Google are starting to use it in real networks. Someone asked if the panelists had any thoughts on the Open Networking Foundation. An unidentified person responded that its primary goal is to harden and extend the OpenFlow spec for customers and vendors.

If You Use Linux, You Should Be Reading LINUX JOURNAL™



» In-depth information providing a full 360-degree look at featured topics relating to Linux

» Tools, tips and tricks you will use today as well as relevant information for the future

» Advice and inspiration for getting the most out of your Linux system

» Instructional how-tos will save you time and money

Get *Linux Journal* delivered to your door monthly for 1 year for only $29.50! Plus, you will receive a free gift with your subscription.

SUBSCRIBE NOW AT:

WWW.LINUXJOURNAL.COM/SUBSCRIBE

Offer valid in US only. Newsstand price per issue is $5.99 USD; Canada/Mexico annual price is $39.50 USD; International annual price is $69.50. Free gift valued at $5.99. Prepaid in US funds. First issue will arrive in 4-6 weeks. Sign up for, renew, or manage your subscription on-line, www.linuxjournal.com/subscribe.

Save the Date!

LISA '11

25TH LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

December 4–9, 2011, Boston, MA

SPONSORED BY usenix IN COOPERATION WITH LOPSA and SNIA

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:

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

Find out more at **www.usenix.org/lisa11/lg**

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

Discounts Available!

20th USENIX Security Symposium

August 8–12, 2011 • San Francisco, CA

Join us for a 5-day tutorial and refereed technical program for researchers, system programmers, and others interested in the latest advances in the security of computer systems and networks.

The Program Begins with 4 Days of Training Taught by Industry Leaders, Including:

- Richard Bejtlich on TCP/IP Weapons School 3.0 (2 Day Class)
- Rik Farrow on SELinux—Security Enhanced Linux
- Jim DelGrosso on an Overview of Threat Modeling
- SANS on Hacker Detection for Systems Administrators (2 Day Class)

And Continues with a 3-Day Technical Program Including:

Keynote Address
Charles Stross, Hugo award-winning author, on "Network Security in the Medium Term: 2061–2561 AD"

Paper Presentations
35 refereed papers that present new research in a variety of subject areas, including securing smart phones, understanding the underground economy, and privacy- and freedom-enhancing technologies

Plus:

- Invited Talks
- Panel Discussions
- Rump Session
- Poster Session
- Birds-of-a-Feather Sessions (BoFs)

Register by July 18 and save!

www.usenix.org/sec11/lg

Don't miss the co-located workshops. Details are available at www.usenix.org/sec11/workshops.

Stay Connected... www.usenix.org/facebook/sec11 http://twitter.com/usenix #SEC11