conference reports

THANKS TO THE SUMMARIZERS

Asad Awan
Maria R. Ebling
Rik Farrow
Andreas Haeberlen
Guy Lichtman
Nikitas Liogkas
Nikolaos Michalakis
Alan Mislove
Tapan Parikh
Swapnil Patil
Ansley Post
Niraj Tolia
Alex Varshavsky

SUMMARIES

NSDI '06: 3RD SYMPOSIUM ON NETWORKED SYSTEMS DESIGN & IMPLEMENTATION 77–90

FREEBSD DEVELOPER SUMMIT AND BSDCAN 90–95

HOTMOBILE 2006: 7TH IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS 95–103

NSDI '06: 3rd Symposium on Networked Systems Design & Implementation

San Jose, CA
May 8–10, 2006

KEYNOTE ADDRESS

PARALLEL COMPUTING AND SEARCH: A PERSONAL PERSPECTIVE

Udi Manber, VP of Engineering, Google Inc.

Summarized by Nikitas Liogkas

Udi Manber started this keynote address by describing his past efforts in computer science. When he started, thirty years ago, people thought that the speed of computation would stop being an issue soon. Then, twenty years ago, he was involved in writing parallel programs that should be robust against unexpected failures. Recently, he worked for the Yahoo! Web portal. He is now the VP of Engineering for Google, a company working on search, a new research area, based on the premise that computing power, main memory, and disk space are very cheap. Traditionally, the "search" feature was considered a commodity, not interesting from a research point of view. This is now changing. Google has made it its mission to "organize the world's information and make it universally accessible and useful." Within Google, the corporate culture is optimized for innovation: Employees can devote 20% of their time to work on their own projects. In fact, many successful products have come out as a result of these efforts through Google Labs. Google has an enormous user base; the traffic on Google servers never stops: They were receiving 1,000 queries/s at 2 a.m. PST on December 25!

The speaker went on to note that humans have not become smarter in the past ten years, but computers have: There are now considerable infrastructure, tools, and algorithms to help us with searching. He described the lifetime of a Google query and outlined potential failures, both in hardware and in software, and how to mitigate their impact by utilizing replication and redundancy. At Google, servers experience an average of 1.9 machine failures per job, so failure is a fact of life. Fortunately, fault-tolerant software makes cheap hardware practical. He then described some of the tools currently used to achieve the desired effects at Google. The key insight behind many of these tools is that many large data sets can have a simple structure; these include the Web page repository, and query logs, as well as the health records of machines. He also briefly mentioned MapReduce, a tool that extracts relevant information for each record of input and then aggregates the results.

He said that a personal goal of his is to make parallel programming easy to use, put search on top of it, and take it to the next level. Nevertheless, search is very hard, for numerous reasons. First, people do not really know how to search efficiently. In addition, content authors do not know how to make their content findable, while schools do not teach proper searching techniques, and universities hardly do research in this area. To make search easier and optimize the search results for users, Google utilizes certain tricks, such as providing the best answer at the top, and using wildcards. There is also a spelling-correction feature, which uses the Web as a contextual lexicon to guess misspelled search queries.

The speaker also shared with us a story about a restaurant search result, where the entire Web page consists of a single image, and how that makes it difficult to provide good results to users. He then went on to describe how Google tries to address all these shortcomings by running usability tests with real users and also by analyzing the logs. He gave an example of such a study by showing anonymous traces of search actions and attempted to analyze user behavior by exploring a specific "full moon" query. Lastly, he prompted all the researchers in the room to build tools for searching and analysis of queries, noting that having fast CPUs and ample storage has the potential for changing the field in a deep way.

During the Q&A period, Brian Noble from the University of Michigan inquired about a potential curriculum of a class on searching, but he did not get a concrete answer. A researcher from the University of Utah noted that Google could even change the culture of a country and warned about the related responsibilities. Michael Freedman from NYU brought up the possibility of adversarial search; another researcher asked about Google's cached pages and whether password-protected pages are cached. The speaker replied that adversarial search is indeed a concern and that cached pages are submitted directly to Google on a voluntary basis; there is no automated tool that Google uses to guess passwords for pages. Emin Gün Sirer from Cornell brought up the idea of searching in a personal context, based on what you have searched for in the past, and the speaker mentioned that this is already offered as a service by Google. Emil Sit from MIT inquired about potential research directions for search, and classes of problems, but the speaker replied that he cannot disclose what Google is currently looking into. A researcher from HP Labs asked whether the problems to solve are different for different search engines, in particular for Yahoo! versus Google. According to the speaker, Yahoo!, being a portal, may be looking at the same problems from a different perspective, but there are definitely many commonalities. Petros Maniatis from Intel Research inquired whether employees have the freedom to define the direction of the projects they work on, or whether questionable or semi-illegal projects are filtered out early. Interestingly, the speaker noted that Google's mantra ("do no evil") is actually followed by management and technical employees alike and that Google already filters out credit card number results, thereby protecting user privacy. Lastly, Peter Druschel from the Max Planck Institute for Software Systems wondered about the role of Google as a media organization providing a variety of opinions and the great responsibility this entails. The speaker replied that the results for a query are not produced manually, but rather through an algorithm, and thus manual intervention is not performed on the results. As such, whatever Google provides is presumably free of bias.

WIDE-AREA NETWORK SERVICES I

Summarized by Niraj Tolia

■ ***Experience with an Object Reputation System for Peer-to-Peer Filesharing***

Awarded Best Paper

Kevin Walsh and Emin Gün Sirer, Cornell University

This talk, given by Kevin Walsh, described the design of Credence, a decentralized peer-to-peer reputation system and the evaluation results from a long-term deployment of the system. The system allows peers to make statements regarding the authenticity of files. While deciding on whether to fetch an object, a client would make a network query to gather previous votes on the object. The votes would be weighted, with a higher weight being given to peers that have voted similarly to the client in the past and a negative weight being given to votes from peers that have had contrary voting patterns in the past. Like-minded peers can be found either through direct correlation or via transitive trust relationships. An implementation of Credence within the Limewire client for Gnutella networks validated its goals as it identified all known large decoys and a number of smaller decoy attacks on the system. More information can be found at http://www.cs.cornell.edu/people/egs/credence/.

During the Q&A session, Phil Gibbons from Intel Research asked about Sybil attacks and whether a single client could cheat the system by providing almost entirely good answers but a few negative ones for the files the client was interested in. Kevin replied that this was hard to do as malicious votes would need to generate a large number of positive votes and therefore would have to do honest work. Further, as an effective attack would require multiple clients, attackers interested in different files would cancel each other out. Jonathan Duerig from the University of Utah asked a similar question about honest users turning into malicious ones and voting down authentic content. However, in this case, even though users might not initially download the content, they will ultimately move down the list, download it, and positively vote on it. This would further eliminate the now-malicious peers from the web of trust. Robert Ricci from the University of Utah asked how many people were required to vote on decoys before Credence became effective.

- *Corona: A High Performance Publish-Subscribe System for the World Wide Web*

Venugopalan Ramasubramanian, Ryan Peterson, and Emin Gün Sirer, Cornell University

This talk, given by Venugopalan Ramasubramanian (Rama), described Corona, a framework for detecting updates to online data while delivering high performance, availability, and scalability. This work is motivated by a large number of online sources that are dynamic and frequently updated. However, polling these sources by a large number of clients is not scalable, delivers poor update performance, and can be responsible for high server load. This work expressed the problem as a mathematical optimization problem that allowed for fast update detection and informed resolution of tradeoffs between bandwidth and latency. The system is layered on top of a structured overlay to decide on node allocation problems for given workloads. For more details, see http://www.cs.cornell.edu/people/egs/beehive/corona/.

During the Q&A session, Hari Balakrishnan from MIT asked whether the techniques used in Corona (a pull system) would also be applicable to push systems. The answer was that some of it would be, since resource allocation with a large number of nodes can also be a crucial problem for push systems. Steve Hand from the University of Cambridge asked about the feasibility of a structured overlay with Corona's optimization being built in from the start. Rama replied that they haven't looked into this but it would be an interesting case to consider.

- *Scale and Performance in the CoBlitz Large-File Distribution Service*

KyoungSoo Park and Vivek S. Pai, Princeton University

This talk, given by KyoungSoo Park, was motivated by the fact that most Web content distribution networks (CDNs) are optimized for small files whereas larger file transfer (100 MB to >2GB) is becoming more prevalent today. Using current CDNs for large files can be wasteful, because each large file evicts a much larger number of smaller files from the CDN's cache, increases memory pressure, and might waste resources as a result of overprovisioning. This work presented the CoBlitz system, which allows CDNs to handle large files by splitting larger objects into chunks and distributing these chunks over different proxies. CoBlitz has been shown to be 55–80% faster than BitTorrent. The talk also discussed how the system addressed the challenges of scalability, robustness, peering set differences, and origin-server load. The evaluation of the system contained controlled experiments as well as results from a real-world deployment over the past two years. More information can be found at http://codeen.cs.princeton.edu/coblitz/, and any public URL can be accessed using the system by surfing over to http://coblitz.codeen.org:3125/.

During the Q&A session, Robert Ricci from the University of Utah asked whether the performance comparison to Shark and Bullet were based on new experiments. However, it turned out that the results were actually directly taken from previously published results, because source or binaries for the systems were unavailable. Jawwad Shamsi from Wayne State University wondered if the experimental setup used different network links for BitTorrent and CoBlitz systems. However, it turns out that the same machines and links were used, to provide a fair comparison. Nick Feamster from Georgia Tech asked whether CoBlitz was faster than BitTorrent because it did not have to deal with free-riding (such as tit-for-tat) or policies such as local-rarest-first. It was argued that this was not the case and that performance was similar because BitTorrent also has seeds to help in such situations, whereas CoBlitz does not. Michale Sirivianos from the University of California, Irvine, mentioned that BitTorrent seemed to be much more scalable for an extremely large number of nodes than CoBlitz. However, KyoungSoo mentioned that CoBlitz too can deliver this scalability and that real usage has shown that CoBlitz can deliver higher mean download bandwidth than BitTorrent in a number of cases. Further, the goal of the CoBlitz project is not to replace BitTorrent but instead to improve Web CDNs.

REPLICATION AND AVAILABILITY

Summarized by Alan Mislove

- *Efficient Replica Maintenance for Distributed Storage Systems*

Byung-Gon Chun, University of California, Berkeley; Frank Dabek, MIT Computer Science and Artificial Intelligence Laboratory; Andreas Haeberlen, Rice University/MPI-SWS; Emil Sit, MIT Computer Science and Artificial Intelligence Laboratory; Hakim Weatherspoon, University of California, Berkeley; M. Frans Kaashoek, MIT Computer Science and Artificial Intelligence Laboratory; John Kubiatowicz, University of California, Berkeley; Robert Morris, MIT Computer Science and Artificial Intelligence Laboratory

Emil Sit started by presenting the motivation for this work, which is to study how to provide efficient replica maintenance in a distributed storage system. Most systems in this area work by creating an initial number of replicas, then monitoring the system for replica failures and responding by creating new replicas. Thus, the high-level problems are determining how many initial replicas to create and deciding which replica failures must be responded to.

To demonstrate why such an analysis is necessary, Sit presented a straw-man naive replication algorithm, showing that the bandwidth cost of replica maintenance is very high (over 250% of optimal). This is largely because most solutions cap the number of existing replicas and focus only on availability, rather than durability. To fix these flaws, the authors present the Carbonite maintenance algorithm, which attempts to provide data durability as a practical goal. From Carbonite, Sit distilled three lessons for replica maintenance:

First, Sit noted that extra replicas (above an artificial cap) avoid the need for future replicas. Thus, the authors recommended removing the artificial limit on the number of replicas alive at any one time. They showed that removing this limit results in an acceptable number of replicas and does not cause the number of replicas to grow without bound.

Second, Sit noted that to provide high availability and durability, the system must be engineered to provide fast repair times. However, since the repair time is often limited by the uplink bandwidth from existing replicas, increasing the number of replicas (in the previous point) increases the rate at which repair can be performed. Additionally, Sit recommended placing replicas on random hosts, to ensure that a variety of access links can be used for repair.

Third, the authors also attempted to handle "bursts" of permanent failures. Using a Markov-chain-based model, they derived the number of replicas required for systems with certain failure properties (and showed that three replicas are sufficient for PlanetLab).

In the Q&A session, James Bacons from the University of Michigan asked how the analysis holds up under more dynamic churn. Sit responded by describing how the system will adapt over time and explaining that the availability fraction does not have to be estimated. Ryan Peterson from Cornell asked about increasing availability by picking a node to create a new replica rather than by choosing one randomly.

■ *PRACTI Replication*

Nalini Belaramani, Mike Dahlin, Lei Gao, Amol Nayate, Arun Venkataramani, Praveen Yalagandula, and Jiandan Zheng, University of Texas at Austin

Jiandan Zheng began this talk by noting that a number of replication systems exist, with each one focusing on one or two of the properties of (i) practicality through partial replication, (ii) arbitrary consistency semantics for applications, and (iii) topology independence by allowing data exchange between any two nodes. Since no replication system currently provides all three of these properties, the authors present the PRACTI architecture for data replication, which allows greater flexibility in choosing tradeoffs and potentially opens up new areas of the design space.

Zheng demonstrated why providing all three PRACTI properties is hard by showing a set of files, including A and B, which are replicated on two desktop computers and one handheld one. Since the handheld computer is space-limited, it can only replicate B and not A. Now, if writes come in on the first desktop for A (resulting in A') and then B (resulting in B'), and the handheld machine is updated, it will only receive the update B' for B, since it is not replicating A. However, if the handheld is used to update the second desktop machine, that machine will have A and B', which does not reflect causal consistency.

To solve these problems and present a unified architecture, the authors presented the following ideas. First, PRACTI uses peer-to-peer log exchanges to synchronize metadata between nodes, similar to Bayou, with the metadata logs always sent first and the data logs fetched on demand. Second, for efficiency, PRACTI allows imprecise invalidations via object group summaries. This provides topology independence (since it is peer-to-peer), arbitrary consistency (since entries are ordered), and practicality (since only metadata is sent, and it can be aggregated).

For their evaluation, the authors compareD the PRACTI system to other replication systems such as Coda and showed that the requirement of a server connection makes other systems much slower.

In the Q&A session, Emin Gün Sirer from Cornell asked whether causal message logging with carefully tracked updates would result in a system that is close to PRACTI.

■ *Exploiting Availability Prediction in Distributed Systems*

James W. Mickens and Brian D. Noble, University of Michigan

James Mickens began the talk by pointing out that the traditional paradigm for handling churn in distributed systems is to use overly pessimistic assumptions. This work, in contrast, wants to understand, predict, and exploit churn. As a preview, by exploiting availability, the authors can reduce data copying in DHTs and are able to avoid unreliable hosts in delay-tolerant networks.

To try and predict when a machine will become unavailable, the authors examineD multiple classes of availability predictors, including predictors based on Markov chains and linear predictors. To evaluate and compare these predictors, the authors tested the predictors on two data sets, one from Microsoft Research and another from the PlanetLab testbed.

They found that PlanetLab is highly predictable in the short term but unpredictable in the long term, since failure events in PlanetLab are unpredictable and infrequent. However, they found the Microsoft Research trace to be much more predictable, largely because many of the machines were almost always on and others exhibited weekly and daily availability patterns.

To examine whether application-level availability differs from machine availability, the authors used the predictors on a trace from the OverNet DHT. They found OverNet availability to be significantly less predictable than the machine availability from PlanetLab and MSR. The authors showed that this is due to an increased level of entropy in the OverNet trace, which fundamentally limits the predictability.

Lastly, to show how their analysis can be used in real-world scenarios, they pointed out that the Chord DHT places replicas on the first k of N successors to increase availability. They showed that by focusing data on highly available nodes within the next N successors, the amount of extra storage required goes down, and the availability goes up. Second, they showed how machine predictability can be used to examine how viruses propagate.

In the Q&A session, topics of discussion included the accuracy of the availability data with respect to ICMP pings and firewalls, and how to use correlated history between machines. Additionally, Mickens was asked whether the predictors could be inverted if the accuracy dropped below 50%.

TOOLS

Summarized by Nikolaos Michalakis

■ *To Infinity and Beyond: Time-Warped Network Emulation*

Diwaker Gupta, Kenneth Yocum, Marvin McNett, Alex C. Snoeren, Amin Vahdat, and Geoffrey M. Voelker, University of California, San Diego

Diwaker Gupta explained the idea of using time dilation to perform tests beyond the limits of the hardware. The motivation for this work was to predict, using currently available hardware, the performance characteristics of hardware that does not yet exist. The key to being able to do this is to recognize time as yet another resource of the system that can be manipulated. Time dilation means that the operating system perceives time as only a small factor of the real time, so with a dilation factor of 10, 10 seconds of real time is perceived as 1 second of system time. Time dilation scales all system components uniformly, but the speaker focused on network dilation for this talk.

To implement time dilation the authors used Xen, and they modified the Xen hypervisor to scale time with a few modifications to the guest OS. To validate that time dilation does not change system behavior, the authors ensured that the same bandwidth and latency characteristics were seen in both the real and the perceived configuration. More specifically, their results showed that time dilation does preserve packet behavior of a single TCP flow, and it is accurate for multiple flows under varying bandwidth. Their experiments tested BitTorrent on multiple nodes. The speaker mentioned that one fundamental limitation to using time dilation in experiments is that they would take longer by a factor of the dilation ratio to finish. Part of the authors' future work would be to implement time dilation with an unmodified OS.

The first question after the talk was focused on what hardware characteristics are not captured by time dilation. The answer to that was that time dilation will work well if hardware improves linearly. Jonathon Duerig from the University of Utah asked how time dilation performs with large systems. Diwaker replied that they haven't experimented with more than 50 machines. Kevin Walsh from Cornell asked whether they scaled all the resources uniformly. The answer was that only temporal resources were scaled. On the question of how the software of 2006 would affect measurements of the hardware of 2026, the answer was that not all changes can be predicted with time dilation.

■ *The Dark Oracle: Perspective-Aware Unused and Unreachable Address Discovery*

Evan Cooke, Michael Bailey, and Farnam Jahanian, University of Michigan; Richard Mortier, Microsoft Research Cambridge, UK

Evan Cooke started the talk by explaining how serious the problem of unused network address spaces is for security and that it is challenging to identify such address spaces, because they are highly distributed over the prefix space. The goal of the work on Dark Oracle was to automate the process of discovering dark addresses by participating directly with allocation, routing, and policy systems. The architecture is composed of two major components. The first is the address allocation data sources. There are three main sources of allocation data: external routing data, internal routing data, and host configuration data (e.g., dhcp). The second major architectural component is the address manager, which utilizes the address allocation data to provide a map of dark addresses.

To reduce the number of misclassified visible addresses as dark addresses, the Dark Oracle samples addresses more often, and to smooth fluctuations it adds a delay from the transition of a visible to dark address. Using this technique the authors found many more dark addresses. More specifically, the benefits of using a perspective-aware approach to dark address discovery are that one can construct both incoming and outgoing honeynets, and automating the process results in an order-of-magnitude more unused addresses being discovered. Finally, by using this approach pervasive honeynets can be built that are better at detecting without being detected.

■ *Pip: Detecting the Unexpected in Distributed Systems*

Patrick Reynolds, Duke University; Charles Killian, University of California, San Diego; Janet L. Wiener, Jeffrey C. Mogul, and Mehul A. Shah, Hewlett-Packard Laboratories, Palo Alto; Amin Vahdat, University of California, San Diego

Patrick Reynolds talked about Pip, an infrastructure for comparing actual behavior and expected behavior to expose structural errors and performance problems in distributed systems. The idea is that the programmer defines the expected behavior of the system and Pip compares actual system runs with the expected model. Pip can find both structural and performance bugs. It works by feeding it application traces, which it uses to reconstruct a behavior model. The expected behavior is input to the Pip checker and the checker returns unexpected events.

Expectations are specified by recognizers and aggregates. Recognizers validate or invalidate individual execution path instances by essentially mapping paths to sets. Recognizers are written in a domain-specific language. Aggregates assert properties of sets, essentially modeling aggregated expectations on sets of paths. Since manual work for writing expectations can be too much, Pip can generate expectations automatically from the application's behavior model. Also, Pip offers a tool for visualizing paths, execution timelines, and performance that helps discover graphs of unexpected behavior. Using Pip, the authors managed to find several bugs in FAB and SplitStream, and by reusing Pip they checked that they did not reappear after the fix.

Timothy Roscoe from Intel Research asked whether the authors could get some theoretical guarantees from using other parsers or model checkers. The speaker replied that using other parsers would not be good to express behavior. Emil Sit from MIT asked where could he run Pip feasibly. The answer was that in practice Pip can reasonably handle about 5 million events. Eric Eide from the University of Utah asked whether Pip's domain-specific language was suitable for expressing the details of the application's programming model. Patrick replied that their language is not bound to any specific programming model. They use messages and event handlers that span different models.

WIDE-AREA NETWORK SERVICES II

Summarized by Ansley Post

■ *OASIS: Anycast for Any Service*

Michael J. Freedman, New York University and Stanford University; Karthik Lakshminarayanan, University of California, Berkeley; David Mazières, Stanford University

Micheal Freedman presented an infrastructure for anycast that can support a wide variety of systems. In general, OASIS tries to find the best replica among many to provide a particular service. Currently, most users must manually select a replica of a service to use, based on their current geographic location. OASIS seeks to improve upon this by automatically finding the best replica for a user. Once an infrastructure for finding the best replica is in place, it can be used by many services and will not have to be reimplemented for each service.

To implement the OASIS service there were several possible solutions that trade off efficiency versus accuracy. One possible solution that is accurate but very costly is pinging the host making the request from all points where the service is provided and then choosing the one with the lowest latency. The authors instead chose to probe each IP prefix, using the insight that 99% of /24 addresses are in the same location. This is a good solution, which does not require probing each requesting host.

The OASIS system consists of two types of nodes: a large set for measurement and a small reliable core to provide the anycast service. A client contacts a resolver, which uses the measurements gathered to direct the request to the best replica given a certain anycast policy specified by the service. The reliable OASIS core uses gossip to disseminate information about failures and the services being provided.

OASIS has been deployed on PlanetLab for six months and is currently used by numerous services. OASIS can potentially provide a large performance improvement and can be easily integrated into an existing service. Other metrics for the best replica can be used, such as load. To configure a service for use with OASIS, the developer must provide a service description, a proxy (in this case PlanetLab), and a set of statistics, such as load on each replica.

■ ***OverCite: A Distributed, Cooperative CiteSeer***

Jeremy Stribling, MIT Computer Science and Artificial Intelligence Laboratory; Jinyang Li, New York University and MIT Computer Science and Artificial Intelligence Laboratory via University of California, Berkeley; Isaac G. Councill, Pennsylvania State University; M. Frans Kaashoek and Robert Morris, MIT Computer Science and Artificial Intelligence Laboratory

This talk was given by Jeremy Stribling, who showed a system, OverCite, that is a distributed and cooperative version of the popular CiteSeer service. Citeseer is a repository of computer science research papers and metadata. The service is currently run on two servers by the University of Pennsylvania. These two servers are often overloaded by the requests that it has to service, often resulting in the system being unavailable to people trying to use it. In the current architecture it is hard to add new resources.

The solution presented is to build a distributed version of CiteSeer, OverCite, which is cooperative and to which new resources can easily be contributed for running the system. The goal of this system is to leverage the parallelism of CiteSeer and for each site that is running the service to have a low resource and administrative burden. It works by distributing the responsibilities of the CiteSeer system to a large number of nodes. These responsibilities include storing research documents and the metadata about the documents, responding to search queries, and crawling the Web for new documents.

To accomplish this, a DHT is used to store all documents and metadata. Documents are partitioned into groups, and hosts in the system are assigned to these groups. Search queries are directed to the right group for service. Some time was spent selecting the right grouping factor. This factor is a tradeoff between latency and the level of parallelism possible. Crawling activity is coordinated using the DHT. OverCite is built upon DHash, Searchy, OKWS, and the OASIS system discussed in the previous talk. It is currently running on the MIT Ron testbed plus some private nodes. The authors hope to make the deployment publicly usable in the near future. Additionally they are planning an open API to the data to allow the creation of new applications on top of OverCite.

■ ***Colyseus: A Distributed Architecture for Online Multiplayer Games***

Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan, Carnegie Mellon University

Colyseus is a system for building massively multiplayer online games (MMOGs). MMOGs have grown exponentially in the past decade; they have scaled because the games often employ certain “tricks” such as partitioning the world onto different servers, keeping many replicas of the gaming universe, or maing slow-paced games. First-person shooters (FPSs) are fast-paced games and as such no MMOG with a large number of players currently exists. The goal of this work is to build a cooperative architecture for FPSs that is very scalable.

A game in Colyseus is broken into components that are immutable, such as the map of the game world, and pieces that are mutable, such as the players and weapons. For each frame of action the game state is updated by a function. If the objects are partitioned across servers this function can be run in parallel. The problems that must be solved to do this are object discovery and replica synchronization. To do this in Colyseus each object has primary and secondary replicas. These replicas are kept weakly consistent; inconsistency is tolerable as this is often present for a short time in decentralized games. Object discovery is accomplished by every object publishing its position. Queries are then made for all objects in a particular area. Several optimizations allow this to be done efficiently, and these are presented in the paper. An example of such an optimization is the leveraging of the physics of the game world to predict which objects can be prefetched.

The experimental evaluation of Colyseus shows both the bandwidth required for running an FPS game and the consistency that is achieved. The bandwidth overhead is 4–5 times higher than a server-based game but still feasible and scales well with number of nodes. The view is consistent to within 0–0.8 objects. These results indicate that building a game on top of Colyseus is indeed feasible.

END-SYSTEM DESIGN

Summarized by Guy Lichtman

■ *Na Kika: Secure Service Execution and Composition in an Open Edge-Side Computing Network*

Robert Grimm, Guy Lichtman, Nikolaos Michalakis, Amos Elliston, Adam Kravetz, Jonathan Miller, and Sajid Raza, New York University

Nikolaos Michalakis presented Na Kika, an architecture for scaling dynamic content with a focus on collaborative efforts. Dynamic content is popular and easy to build but hard to scale. Examples include popular mashups such as zillow.com. Current platforms do not address small independent content producers and present a clear tension between extensibility and securit, which Na Kika tries to reconcile.

The architecture uses DNS to direct clients to nearby proxies that are organized in a structured overlay. Scripts are published like static content and executed on the proxies. The programming model is based on scripted event handlers. OnRequest and onResponse handlers are used for producing requests and responses, respectively. Service modularity is achieved through the descriptive nature of HTTP messages. Handlers are selected based on HTTP message properties and execute a most specific match. Composition is achieved through putting event handler pairs, one after the other through a nextStage predicate. The same mechanisms used for event handler selection and composition are used to enforce security admission and emission control through two additional stages, separating the client and the server from the Na Kika pipeline. Hosted code is contained through script sandboxes, but this doesn't prevent scripts from overusing memory and cpu. Na Kika uses an approach of controlling consumption only under congestion. No hard quotas are used. Throttling is used to reject requests and, as a last resort, terminate active requests.

Evaluation included using the Wise-MD application, a Web-based education tool developed at the NYU medical school. It is spread across multiple universities across the world and is multimedia intensive, with both static and dynamic content. One developer ported Wise-MD to Na Kika in 2 days. Simulated traces with Na Kika cold cache and Na Kika warm cache were executed and the results were compared to those of a single server: Na Kika both cold and warm cache clearly outperform a single server in terms of failure rate and bandwidth seen by clients. For dynamic content as well, the response time is significantly lower. Additionally, Nikolaos provided four examples demonstrating Na Kika's extensibility and how building extensions is easy and scalable. Each extension took less than 100 lines of code to implement using Na Kika.

■ *Connection Conditioning: Architecture-Independent Support for Simple, Robust Servers*

KyoungSoo Park and Vivek S. Pai, Princeton University

Vivek Pai started by stating that, overall, server performance is great, owing to many OS contributions, Moore's law, and load balancers, but that poor server performance is often seen in CGI, db, and network. There has been a lot of research in building server software in the past 10 years, including async IO, events, and so on. But this hasn't mattered too much. Apache is still the most popular server (with a 75% market share). Users love multi-process servers, as they make it easier to add features and modules. Pai makes a case to go with the flow and bring research benefits to Apache. He points out some economics: a Walmart Linux machine costs $400, an HP DL320 is under $3000, but a 100 Mbps WAN runs $30,000/month. The main conclusion here is that hardware costs are dwarfed by network costs.

Connection Conditioning's new approach is to make servers simpler, more robust, and easier to program, defend, and share. They might be a little slower, but that's OK, since hardware is cheap. Pai suggests doing this by using an old idea, UNIX pipes, which are good for text processing but aren't used for servers. In Connection Conditioning (CC), instead of having clients talking to the big server, the server instantiates filters. The clients talk to the filters, which bundle the requests and pass them on between the filters to the servers. Filters are separate processes; thus you can write them however you like, using threads, processes, or events. Communication is done via UNIX domain sockets, thus allowing passing of the socket/request bundle. The server sees TCP sockets, and responses are sent directly via the client socket. The main idea behind the filter design is that the inbound path is lightweight and the outbound path is heavyweight; thus, filtering the requests and passing the response socket works well.

Experience shows that modifying Apache takes less than 50 lines of change to add support for CC filters, and flash takes around 30 lines. Additionally, writing a new server that is designed to take advantage of this library is easy. A sample server, CC-Server, took about 80 lines of code. CC-Server handles non-parallel processing and is a proof-of-concept server which isn't meant to replace Apache. Performance tests using a single file test and a file mix similar to the SpecWeb99 static content benchmark show that CC-Server performance is at least comparable to Apache's. Robustness tests applying incomplete connections show that Apache dies at around 150 connections per second, whereas CC-Apache stays steady around 2000 requests per second.

■ ***PCP: Efficient Endpoint Congestion Control***

Thomas Anderson, Andrew Collins, Arvind Krishnamurthy, and John Zahorjan, University of Washington

Anderson began by saying that TCP is known to be suboptimal. The basic thrust of PCP is to take a systems approach to the problem and optimize it for the common case. TCP starts slowly and increases bandwidth until loss is detected. But most transfers are small to moderate and thus stay at the wasted capacity stage. From a systems point of view, if we could control the network we could easily build functionality into the network to solve the problem. As an example, consider ATM rate control, where the network indicates the safe rate to transmit. The real problem is that it is not trivial to change the network. Anderson raises the following questions: Can we accomplish something that is as good without changing the network? What is the best that we can do?

PCP uses the assumption that endpoints can cooperate. PCP does well both with and without fair queuing in the middle of the network as compared to TCP, which performs worse with fair queuing systems. PCP's approach is to directly emulate the behavior the router would use if we had control of the underlying network. The basic idea is to query the network for what rate it can support. Its main goals include minimal transfer time, negligible packet loss, work conservation, stability under extreme load, and fairness. TCP achieves only the first three goals.

Performance evaluation measured PCP versus TCP on US RON nodes. PCP outperforms TCP, since if there is a loss TCP goes through a painful recovery process that takes time to recover from. Even for 4 PCPs it outperforms TCP. The tests show that PCP isn't causing TCP backoff. Simulation shows that PCP actually does better at fairness than TCP, as packet losses are not a good way to achieve fairness. Even in a fair-queue router PCP does very well compared to TCP.

MEASUREMENT AND ANALYSIS

Summarized by Guy Lichtman

■ ***Availability of Multi-Object Operations***

Awarded Best Paper

Haifeng Yu and Phillip B. Gibbons, Intel Research Pittsburgh; Suman Nath, Microsoft Research

Haifeng Yu started by saying that placing objects for multi-object operations is hard. There are various ways to make assignments. The paper's main contributions include:

1. Identity assignments as a critical design choice.

2. Operations classified as either strict or loose, using two strategies, RAND (random) and PTN (partition).

3. Proposed design to approximate PTN under dynamic settings.

4. MOAT implementation and evaluation.

Some existing systems use assignments that are close to PTN, such as Glacier. Glacier becomes PTN only if nodes are evenly distributed on the ring. Existing systems similar to CAN (GFS, FARSITE) use RAND assignment.

Comparing RAND and PTN shows that they give dramatically different behaviors. There is a crossing point very close to the 1.0 fraction of objects needed for the operation to succeed. All other assignments fall between RAND and PTN. At crossing point RAND is better than PTN. This allows classifying to types of operations: loose and strict. Theoretical results show that PTN works best among all assignments for strict, and likewise RAND for loose. Haifeng proposed Group DHT to solve the limitation of Glacier. He further presented MOAT, a wide-area object storage system. It supports a range of assignments.

PlanetLab and simulation were used for evaluation with a TPC-H benchmark workload. Glacier behaves similarly to Chord, but Group DHT outperforms Glacier. The difference between assignments can be three 9s. Object assignments have critical impact on the availability of multi-object operations. It is not about "concentration" and "spread." The results show that it is impossible to combine best assignments for both strict and loose.

■ *Subtleties in Tolerating Correlated Failures in Wide-area Storage Systems*

Suman Nath, Microsoft Research; Haifeng Yu and Phillip B. Gibbons, Intel Research Pittsburgh; Srinivasan Seshan, Carnegie Mellon University

Suman Nath stated that traditional replication techniques assume failure independence. In practice, failures are not independent. Correlated failures include DoS attacks, viruses, etc., and hurt availability dramatically. Techniques to fight against correlated failures include predication, modeling, and overprovisioning.

Suman presented how they found that existing failure pattern prediction techniques give negligible benefits. Additionally, simple failure models have dramatic inaccuracies, and overprovisioning doesn't help much, either. The study is based on three failure traces: Planetlab, Web server, and RON trace. The traces are long and thus capture rare correlated failures. The study looked at several ways of mitigating correlated failures, including failure pattern prediction using introspection (OceanStore), modeling correlated failures (used in Glacier), and using smaller fragments. Suman designed and implemented IrisStore, a distributed storage system similar to OceanStore. The system is based on the design principles learned in this study and show that during the SOSP 2005 deadline PlanetLab experienced numerous failures but still IrisStore maintained over 90% availability.

■ *Open Versus Closed: A Cautionary Tale*

Bianca Schroeder, Adam Wierman, and Mor Harchol-Balter, Carnegie Mellon University

Schroeder started out with a motivation example on scheduling static Web requests and a question: Is it possible to improve response time by better scheduling of requests to Web servers? To try to answer this question she proposed an implementation of SRPT (shortest remaining time) scheduling and used workload generators to test the idea. Two generators were used. One had a tuning knob for request rate; the second had a tuning knob for number of users and think time. These generators brought different results for plain Apache. Using the TCP-W benchmark with one generator, SRPT gave a huge improvement; with the other generator, a small improvement.

These differences arise because there is a fundamental difference between open and closed systems used for simulation and generation. Closed systems can be viewed as user driven. Each user resides in an endless loop of sending a request, then waiting during a think time period, then repeating the cycle. This differs from the open system model, where the user sets the request rate. In an open model there is no maximum number of simultaneous users. There are popular generators that use both models, but very little documentation and literature about these models exist. An analysis of both systems shows an order of magnitude difference in the response times. Open systems have a higher response time. Variability affects open systems much more than closed ones. This is because of the inherent dependence between arrivals and completions in a closed system. In some sense a closed system reduces burstiness.

A hybrid option, which is equivalent to open arrivals of session and then a closed stream of requests per session, would probably be more suitable. To analyze what real Web workloads model, we can look at the number of requests per visit. The more requests per visit, the closer it is to a close system. There is of course an area in between, for which the hybrid option is more suited.

ARCHITECTURES AND ABSTRACTIONS (AND EMAIL)

Summarized by Swapnil Patil

■ *An Architecture for Internet Data Transfer*

Niraj Tolia, Carnegie Mellon University; Michael Kaminsky, Intel Research Pittsburgh; David G. Andersen and Swapnil Patil, Carnegie Mellon University

In the first talk of the session, Tolia presented a new architecture, DOT, for Internet data transfers. Current applications have tightly coupled content negotiation and data transfer; DOT separates the two and introduces a service to handle all data transfers. DOT provides a modular, plug-inbased architecture that helps applications easily use new innovations in data transfer. DOT currently supports three plug-ins: for storage, multi-path transfers, and portable storage devices. DOT uses content-based techniques for chunking-based data transfers and application-independent naming semantics. As a part of the evaluation, DOT has been used for file transfer application, production mail server (Postfix), and multi-path transfers. DOT performs with a negligible overhead, and integration of DOT in the Postfix mail server required less than 200 lines of code. Additional information about the project is available at http://www.cs.cmu.edu/~dga/dot.

There were several questions. What kind of security does DOT support? DOT will use convergent encryption for individual chunks and their hashes. This is a work in progress. Does the sender have the flexibility to decide what chunking policy to use? Yes, as long as the sender and the receiver agree on the chunking policy. How does this compare with LBFS chunking? Like LBFS, DOT uses Rabin fingerprinting for chunking.

■ *OCALA: An Architecture for Supporting Legacy Applications over Overlays*

Dilip Joseph and Jayanth Kannan, University of California, Berkeley; Ayumu Kubota, KDDI Labs; Karthik Lakshminarayanan and Ion Stoica, University of California, Berkeley; Klaus Wehrle, RWTH Aachen University

This talk, by Dilip Joseph, was motivated by a need for legacy applications to support the new network architectures, services, and overlays. OCALA (Overlay Convergence Architecture for Legacy Applications) provides an overlay-agnostic approach for legacy applications and as a proxy for incremental deployment. OCALA adds a new Overlay Convergence (OC) layer below the transport layer in the network stack. This OC layer replaces the network layer and has two components, overlay-dependent and overlay-independent sublayers. This helps users on a regular IP network communicate through the overlay. In addition, OCALA can enable applications running on the same machine to use different overlays and stitch different overlays to help users of these overlays to communicate with each other. OCALA has a very extensible architecture, making it easy to implement a new overlay network. At the end, Dilip gave a quick demo of their techniques by accessing his home machine, behind a NAT, through the OCALA proxy. OCALA is available for download at http://ocala.cs.berkeley.edu/.

■ *Distributed Quota Enforcement for Spam Control*

Michael Walfish, J. D. Zamfirescu, Hari Balakrishnan, and David Karger, MIT CSAIL; Scott Shenker, University of California, Berkeley, and ICSI

In this talk, Michael Walfish presented the design and implementation of a quota-based spam control system. The basic goal is to control sent messages by making it more expensive to send bulk mail. An allocator gives each sender a certain quota of mail messages that it can send. Every message the sender sends has a stamp, which is verified by the receiver. On getting email, the receiver checks whether the stamp is valid. The receiver does this by asking the quota enforcer if this stamp has been seen before. If it is a new stamp, then the email is valid and the enforcer stores the record for this stamp. If this stamp was seen before, then the mail is discarded as spam. The main challenge is the design of the enforcer to store billions of stamps, tolerate Byzantine and crash faults, and provide a fast stamp lookup. The evaluation showed that the system can easily handle over 80 billion messages per day.

People asked a couple of questions at the end of the presentation. How do the micropayments work in distributed quota enforcement? The details about micropayments are in the paper. Can probabilistic marking be used, instead of providing such a large infrastructure? If you want to avoid infrastructure costs, then probabilistic marking (or any other technique) can be used.

■ *RE: Reliable Email*

Scott Garriss, Carnegie Mellon University; Michael Kaminsky, Intel Research Pittsburgh; Michael J. Freedman, New York University and Stanford University; Brad Karp, University College London; David Mazières, Stanford University; Haifeng Yu, Intel Research Pittsburgh and Carnegie Mellon University

The final talk of the session, by Michael Kaminsky, described another proposal to control spam. This work proposes a new whitelisting system that works on social networking by exploiting "friend-of-friend" relations among email correspondents. This technique automatically populates the whitelists by using attestations among different users. For example, suppose A and B know each other, B and C are friends, but A and C don't know each other. Using the fact that A knows B is not a spammer and B knows that C is not a spammer, then A may conclude that C is unlikely to be a spammer. These attestations are kept at the server and are checked when any mail is received from a sender. In addition, RE uses cryptographic private matching techniques to preserve the contacts and whitelists. Using email traces, the evaluation showed that RE can accept about 75% of received email and can prevent up to 88% of the false positives incurred by existing spam filtering.

In the questions that followed, someone asked why the results show that you lose 13% of email. This happens because the one-hop social network does not have enough whitelists. How many hops of social network can work? You can have any number of hops. This increases the whitelist size, which will in turn take more time for verification. What happens if a spammer gets access to the address-book whitelist? Currently, RE does not handle this threat model.

WIRELESS AND SENSOR NETWORKS

Summarized by Asad Awan

■ *PRESTO: Feedback-driven Data Management in Sensor Networks*

Ming Li, Deepak Ganesan, and Prashant Shenoy, University of Massachusetts, Amherst

Ming Li presented PRESTO (PREdictive STOrage), an energy-efficient data management architecture for query processing and data acquisition in sensor networks. PRESTO architecture capitalizes on cooperation between resource-rich proxies and resource-constrained sensor nodes by splitting system intelligence between these two network tiers. Li demonstrated this key design principle of PRESTO by discussing the pros and cons of the sensor-centric and proxy-centric architectures. Sensor-centric architectures achieve greater energy efficiency (through in-network processing) and accuracy while sacrificing query-response latencies and introducing complexity at sensor nodes. The proxy-centric approach pushes intelligence to the sensor network edge, thus reducing query latencies, while trading off energy efficiency or data accuracy.

PRESTO achieves low query latencies by replying to queries using a data cache of proxies on the network edge. Energy efficiency and data accuracy result from exploiting the predictable structure of sensor data. Finally, proxy-to-sensor feedback allows the system to adapt to data and query dynamics. The model-driven push component of PRESTO captures deviations in sensed data relative to the prediction model. Sensor nodes only transmit anomalies, thus reducing communication costs and providing an accuracy bound on data cached at proxies. However, if the cached data is not sufficient to respond to a query, PRESTO acquires data using a model-driven pull of sensor data. Data from push/ pull operations is refined using interpolation and reprediction to maintain high cache accuracy. The proxy observes acquired data for trend changes and provides feedback to adapt the data model used at sensors. Similarly, the system adapts to query dynamics using feedback in the form of tuning parameters to sensors.

PRESTO uses an Autoregressive Integrated Moving Average (ARIMA) to model the time series of observations, allowing extraction of both long- and short-term data trends. Further, the asymmetric processing requirements of ARIMA (with computationally intensive model training and low-processing cost data prediction) fit well with the computation split between resource-rich proxies and resource-constrained sensor nodes. Li concluded the talk with performance evaluation results, which showed the efficacy of PRESTO in terms of communication costs, accuracy, and query latency.

■ *Practical Data-Centric Storage*

Cheng Tien Ee, University of California, Berkeley; Sylvia Ratnasamy, Intel Research, Berkeley; Scott Shenker, University of California, Berkeley, and ICSI

Cheng Tien Ee presented PathDCS, an approach that enables data-centric storage (DCS) in sensor networks without requiring point-to-point routing. Ee explained that queries over sensor networks require locating data pertaining to observed events. In the DCS approach, data identified by its content is stored at a specific location. Any query for this content is routed to this location, reducing messaging overhead. DCS is an optimal solution (under certain conditions; cf. Ratnasamy's GHT paper). DCS requires a location reference system and a data-to-location mapping. Ee explained that the current DCS approaches rely on point-to-point routing, which has significant performance overheads. Existing approaches (e.g., GHT) elegantly use a geographical reference system. However, the need to account for (geographical) network boundary and coverage holes in such approaches has yielded complex solutions. PathDCS provides a simple and scalable solution. It builds on a widely used routing primitive, namely, tree routing.

Using an animation, Ee explained the fundamentals of the PathDCS algorithm. PathDCS defines storage location using beacon nodes acting as reference locations. Trees rooted at each beacon are built. The PathDCS algorithm selects a data storage node based only on the identifier of the data and the beacon node ids and does not depend on the location of the source of the data. Data routing simply uses the existing tree paths. Ee also described how beacons to be used in PathDCS can be elected. Ee explained that to handle failures and load balancing, one-hop neighbors of beacons take over. The key intuition here is to minimize the changes in path to the beacon and hence to the storage locations. Further, PathDCS relies on refreshing data to account for path (and storage location) changes. Simulation and system-based evaluation results were presented. The results showed that PathDCS achieves a good load balance, has high route completion and data lookup success, and is robust in failures.

Walsh asked how nonuniformity of data events would affect the load balance of the system. Ee replied that for such a scenario, balancing keys and local replication can be used. Michalakis commented that one-hop replication might not be a good idea, because failures may be geographically correlated. Ee replied that the paper focuses primarily on routing, not storage. The use of one-hop replication is thus to increase lookup success by considering the effects of path fluctuation. Other mechanisms to improve replication can be used and are complementary. Ganesan inquired how the system design would change if the network were heterogeneous. Ee replied that since beacons require more resources, powerful nodes can be used as beacons. They can also be used to cache data to reduce load for the rest of the network.

■ *Geographic Routing Without Planarization*

Ben Leong, Barbara Liskov, and Robert Morris, MIT CSAIL

Ben Leong presented the Greedy Distributed Spanning Tree Routing (GDSTR) algorithm. In geographic routing, packet destinations are specified with x-y coordinates. Packets are first forwarded greedily, that is, to the neighbor closest to the destination. However, greedy forwarding causes a packet to end up at a dead end, where all the neighbors of a node are further from the destination than it. Leong explained that the existing approach is to forward the packets around the voids in the network on the faces of a planarized version of the connectivity graph. It turns out that planarization is difficult to achieve in a distributed environment and was only recently solved by Kim et al. using CLDP. Leong commented that the complexity and overheads of this scheme motivated GDSTR, which does not require planarization.

GDSTR routes packets around voids by routing on an annotated spanning tree, called a hull tree. Convex hulls are used to aggregate information about the points that are reachable along different branches of the tree. By forwarding packets up the tree (i.e., decreasing levels of the tree) the packet can be forwarded to an appropriate subtree that contains the destination. A key decision in routing a packet around a void is to choose a forwarding direction (clockwise or anticlockwise). A good forwarding direction choice yields fewer hops to the destination. Leong explained that GDSTR uses two extermal-rooted trees to "approximate" a void, and the simple heuristic of choosing the tree with a root that is closer to the destination is often equivalent to choosing the optimal forwarding direction. GDSTR guarantees packet delivery in a connected graph. Simulation-based comparisons among GPSR, GOAFR, GPVFR, and CLDP showed that GDSTR outperforms the other approaches, in terms of hop stretch, by a large margin when the average node degree of the network topologies is low (i.e., when voids are large). Results for GDSTR with different numbers of hull trees around voids showed that using two trees achieves significant improvements in hop stretch performance over using only one tree. Having more than two trees yields marginal benefits. Leong commented that the computation and memory overhead of GDSTR are also low. Similarly, the bandwidth overhead is an order of magnitude lower for GDSTR than for CLDP, while routing performance of GDSTR is up to 20% better.

In response to a question from the audience about the effect of overlapping convex hulls on routing performance, Leong responded that routing performance will be worse if there are intersecting hulls, since the size of the search tree will be increased. However, this is only if the destination falls within the intersection of the hulls.

FILE AND STORAGE SYSTEMS

Summarized by Andreas Haeberlen

■ *Virtualization Aware File Systems: Getting Beyond the Limitations of Virtual Disks*

Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum, Stanford University

Ben Pfaff proposed a new storage solution for virtual machines (VMs) that combines the benefits of virtual disks and distributed file systems. Virtual disks provide useful features such as rollback and versioning, which allow users to specialize VMs for different tasks. However, these features are only available at disk granularity, which makes it difficult to roll back individual files, and virtual disks cannot be shared among multiple VMs. Distributed file systems offer both fine-grain access and sharing, but they lack other features such as versioning and specialization. Ben argued that we can get a better storage solution by combining the two; the result is a virtualization-aware file system.

In the second part of his talk, Pfaff presented a prototype system called Ventana. In Ventana, virtual disks can be recursively specialized, which results in a tree of versions. Multiple VMs can share a branch of this tree, so that changes by one VM become visible by all others; also, there is a mechanism that can be used to restrict access to certain branches. To interact with virtual machines Ventana uses the NFS protocol, which is understood by most operating systems. Ben concluded his talk by discussing several use cases, for example, how to use the branch hierarchy to apply a security patch efficiently in multiple virtual machines.

■ ***Olive: Distributed Point-in-Time Branching Storage for Real Systems***

Marcos K. Aguilera, Susan Spence, and Alistair Veitch, HP Labs

The talk was given by Marcos Aguilera, who argued that a widening gap between storage capacity and transfer rates makes it increasingly difficult to handle large volumes of data. For example, an administrator may want to archive a snapshot of a volume for further reference, or run a "what if" installation of a new software package without affecting the main copy. Aguilera presented Olive, a distributed and replicated storage system that addresses these problems by providing an efficient branching operation. By creating a new branch, the user obtains a second copy of a volume which can evolve independently from the first.

Aguilera pointed out that the main technical challenge in Olive was to provide strong consistency, and he described the mechanisms Olive uses to achieve this. Specifically, Olive provides linearizability, which implies that the state captured by a branch is one that could also have resulted from a crash. Aguilera also presented evaluation results from an implementation of Olive in the federated array of bricks; he showed that a new branch can be created in tens of milliseconds and that the per-branch metadata is small enough to allow dozens of branches.

■ ***Pastwatch: A Distributed Version Control System***

Alexander Yip, Benjie Chen, and Robert Morris, MIT CSAIL

In the last talk of the conference, Alexander Yip presented Pastwatch, a cvs-style version control system that supports disconnected operation. In Pastwatch, users do not have to be connected to a central server to commit changes; for example, a small group of developers can use it to collaborate while on an airplane and later merge their changes into the main repository when they are connected to the network again. Of course, this can result in write conflicts if multiple disconnected users modify the same file. Pastwatch handles this by lazily creating branches, which are visible to the users and can be merged later.

Each Pastwatch user maintains his or her own local copy of the repository, which is organized using a special data structure called a revtree. The revtree structure is such that two repositories can be synchronized simply by forming the set-union of the revtree nodes. This allows updates to spread in an ad-hoc manner and yet ensures eventual consistency. Pastwatch has been in production use for over a year, and an implementation is available for several major operating systems. More information is available at http://pdos.csail.mit.edu/pastwatch/.

In the Q&A session, Yip fielded several questions regarding how Pastwatch handles write conflicts. Brad Karp asked how users could find out about new branches; the answer was that Pastwatch displays an explicit warning during synchronization. Eric Eide asked what would happen if two users reconciled the same branches; the answer was that Pastwatch would create another branch, but that this had rarely been observed in practice.