;login:

THE MAGAZINE OF USENIX & SAGE
August 2003 • volume 28 • number 4

inside:

CONFERENCE REPORTS

HotOS-IX

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

HotOS-IX

May 18–21, 2003
Lihue, Hawaii

[*This is a somewhat abbreviated set of summaries of the events at this conference. A complete set of summaries is available at* http://www.usenix.org/events/hotos03/. *Ed.*]

INVITED TALK

Summarized by David Oppenheimer

Operating Systems: Shouldn't They Be Better?

Andrew Hume, AT&T Labs–Research

Andrew Hume gave the HotOS keynote talk, explaining that his perspective comes from having designed, implemented, and delivered large-data applications for more than 10 years. The problems he discussed in the talk were that operating systems have gone "from a help to a hindrance," that even users' lowered expectations for operating systems have not been met, and that as a result, applications have to be designed around OS quirks. Hume pointed out that this situation hasn't always been the case, citing WORM-based backup systems in research versions of UNIX and a cluster-based billing system that AT&T built using Plan 9 as examples of systems that were highly reliable, even under load.

The first problematic system Hume described was Gecko, a large-scale (250GB/day) billing system implemented in 1996 on Solaris 2.6. AT&T required 1GB/sec. of file-system throughput and predictable use of memory. Among the problems encountered were: Solaris crashed every few days for the first six months that the system was in production; Solaris limited file throughput to about 600MB/sec.; reading large files sequentially crashed the VM; and a "VM roller coaster" developed when a large chunk of memory was allocated (causing a repetitive page-out, page-in cycle of all the system's physical memory, rather than just paging out the amount of new memory needed).

The second problematic system Hume described was a replacement for Gecko that required six times the capacity of the original Gecko. This system was implemented on a cluster running Linux. The architecture was a "Swiss canton" model of loosely affiliated independent nodes with a single locus of control, data replication among nodes, and a single error path so that software could only halt by crashing (there was no explicit shutdown operation). Hume described eight problems the Gecko implementers experienced with Linux (versions 4.18 through 4.20), including Linux's forcing all I/O through a file-system buffer cache with highly unpredictable performance scaling (30MB/sec. to write to one file system at a time, 2MB/sec. to write to two at a time), general I/O flakiness (1–5% of the time corrupting data read into gzip), TCP/IP networking that was slow and that behaved poorly under overload, lack of a good file system, nodes that didn't survive two reboots, and slow operation of some I/O utilities such as df. In general, Hume said he has concluded that "Linux is good if you want to run Apache or compile the kernel. Every other application is suspect."

Hume proposed the following definition of OS reliability: "[The OS] does what you ask, or it fails within a modest bounded time." He noted that FreeBSD has comparable functionality to Linux, better performance, and higher reliability, and he speculated that this might stem from BSD's (and other "clean, lean, effective systems") having been built using "a small set of principles extensively used, and a sense of taste of what is good practice, clearly articulated by a small team of mature, experienced people." Hume took Linux to task for not demonstrating these characteristics, in particular for being too bloated in terms of features, and for having been developed by too large a team. Further, he singled out the Carrier Grade Linux effort for special condemnation for "addressing zero of the [types of] problems" he has had.

SESSION: THE EMPEROR'S CLOTHES

Summarized by Matt Welsh

High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two

Charles Blake and Rodrigo Rodrigues, MIT Laboratory for Computer Science

Charles Blake spoke on the overheads of "maintenance bandwidth" – network bandwidth consumed to maintain a given level of replication or redundancy – in a peer-to-peer storage system. The basic argument is that maintenance bandwidth across the WAN, not the aggregate local disk space, is the fundamental limit to scalability in these systems. Given the dynamics of nodes joining and leaving the system, Charles presented a conservative estimate of the maintenance bandwidth that scales with the WAN bandwidth and average lifetime of nodes in the system. Under a typical scenario (100 million cable modems with a certain bandwidth available for replication, one week average lifetime, and 100GB storage per node), only 500MB of space per node is usable, only 0.5% of the total.

To try to address these problems, Charles looked at alternatives such as admission control (only admitting "reliable" nodes) or incentivizing nodes to have long lifetimes. It turns out that a small core of reliable nodes (such as a few hundred universities with a single reliable machine dedicated to hosting data) yields as much maintenance bandwidth reduction as millions of home users with flaky connections. The talk concluded with a number of open issues in organizing WAN-based storage systems, such as whether it is appropriate to assume millions of flaky users and whether the requirement of aggregate data availability should be reconsidered.

ONE HOP LOOKUPS FOR PEER-TO-PEER OVERLAYS

Anjali Gupta, Barbara Liskov, Rodrigo Rodrigues, MIT Laboratory for Computer Science

Anjali Gupta presented a talk on the use of one-hop lookups in peer-to-peer systems, avoiding the high latency associated with the typical log (N) lookup paths required by most systems. The challenge is keeping up with membership change information on all hosts. For example, the UW Gnutella study in 2002 showed an average node session time of 2.9 hours, implying 20 membership changes per second in a system with 100,000 hosts. Anjali presented a hierarchical scheme, in which the address space (forming a ring) is subdivided into slices, each with a slice leader that is the successor to the midpoint in the slice. Slices are further subdivided into units.

The basic approach is for nodes to exchange frequent keep-alive messages with their predecessor and successor nodes. A change to a node's successor is an event that is propagated by piggybacking a recent event log onto keep-alive messages. A node change event is relayed to the slice leader, which periodically (every 30 seconds) notifies other slices of the updates. Internally to a slice, slice leaders periodically (every 5 seconds) notify unit leaders of node change information. Given some reasonable assumptions on the size of the system, all nodes can be updated within 45 seconds of a node leaving or joining the system, which permits a 99% "hit rate" for an address lookup. In this scheme, it is important to choose good slice leaders that are well-provisioned. Anjali concluded with a summary of ongoing implementation and experimentation work, noting that systems larger than a million nodes will require two-hop lookups.

AN ANALYSIS OF COMPARE-BY-HASH

Val Henson, Sun Microsystems

Val Henson presented one of the most controversial papers of the conference, admonishing those systems that rely upon comparison of data by comparing cryptographic hashes of the data. Many systems (such as rsync, Venti, Pastiche, LBFS, and OpenCM) use this technique, but it is not yet widely accepted by OS researchers, due to little characterization of the technique and many unanswered questions. The risk of collision using (say) a 160-bit SHA-1 hash is just 2^{-160}, which is far lower than a hardware failure or probability of an undetected TCP error. So why the controversy?

First, these techniques assume that data is random, but real data is not random and has a fair amount of commonalities (think about ELF headers and English text). Second, cryptographic hashes were designed for authentication and care about "meaningful" collisions, such as two contracts with the same text but different dollar amounts that happen to collide in the hash space. Third, hash algorithms are short-lived, and obsolescence is inevitable – systems need an upgrade strategy. Finally, collisions are deterministic – two blocks that collide always collide – rather than a transient error such as a hardware fault. Hash collision is therefore a silent error in those systems that rely on compare-by-hash techniques. Val claims that we should be striving for correctness in systems software, not introducing "known bugs." It is OK to rely on compare-by-hash when the address space is not shared by untrusted parties, and when the user knows and expects the possibility of incorrect behavior — citing rsync as an example. Note that "double hashing" is not an acceptable solution, as this results in just another hash function, albeit one with a lower collision probability.

Some alternatives to compare-by-hash were discussed, such as content-based addressing that checks for collisions, using compression, maintaining state to only send or store identical blocks once (as in LBFS), sending diffs instead of an entire block, or using universal IDs for common blocks.

WHY EVENTS ARE A BAD IDEA (FOR HIGH-CONCURRENCY SERVERS)

Rob von Behren, Jeremy Condit, Eric Brewer, University of California, Berkeley

Rob von Behren raised the argument of thread-based versus event-driven concurrency in high-concurrency servers, claiming that thread-based approaches are far better, due to their ease of programming. To counter the arguments that threaded systems have inherently higher overhead than events, Rob presented early results from a lightweight user-level thread system that performed as well as an event-driven system on a Web server benchmark. Furthermore, threads have better programming and debugging tools, leading to increased productivity. To address the problem of high overhead for per-thread stacks, Rob proposed the use of compiler support to automatically compress stacks, for example, by moving "dead" elements off the stack across a blocking operation. Using cooperative scheduling avoids the overhead of generic thread synchronization, but there are some issues to address here such as fairness, the use of multiprocessors, and how to handle spontaneous blocking events such as page faults.

Rob pointed out that events have the advantage of permitting very complex control flow structures, but very few programmers use these structures and threads can capture the vast majority of scenarios. Another problem with thread schedulers is that they are "generic" and have little knowledge of application structure. To permit greater cache locality, Rob proposed "2D" batch scheduling, in which the compiler annotates the application code to indicate to the scheduler system where the various stages of the thread's execution are located.

Rob presented some measurements of a simple Web server benchmark based on his user-level threads package, capable of supporting over 100,000 threads, implemented in about 5000 lines of C code. The server outperforms a Java-based event-driven Web server, probably due to the large number of context switches in the event-driven system. Rob concluded that it may be possible to achieve higher performance using threads than events, in part because events require dynamic dispatch through function pointers that makes it difficult to perform inlining and branch prediction.

Panel Discussion

Charles Blake kicked it off by asking why Val Henson's birthday paradox probability was so hard to compute. She responded that essentially it comes down to the infinitesimal numbers involved.

Eric Brewer pointed out that systems should use CRC, not MD5; since CRC is no good for preventing malicious collisions, there is no illusion that it is. One should also use a random salt with the checksum, which should help with the non-randomness of real data. Val responded that if you have to recompute the checksum across the actual data, then you are losing the benefits of this technique.

George Candea raised the point that although a P2P client that prevents a user from disconnecting appears less desirable at first, it would lead to higher availability for the service as a whole. This makes the service more valuable, and hence provides greater incentive to use it (i.e., download the client).

Ethan Miller asked whether people are really comfortable with the concept of probabilistic storage. Val agreed that the notion of dynamic, unreliable storage systems makes her uncomfortable.

Ranjita Bhagwan pointed out that Charles's calculations don't push P2P out of the picture, asking whether there may be a cost benefit to a peer-to-peer approach versus a centralized approach. Charles said that fundamentally his argument was economic, concerning the bandwidth versus storage requirements for these systems. Andrew Hume said that the best nodes are professionally managed and that high-bandwidth connections and support are expensive, so the economics of the two approaches are more similar than they are different.

Mohan Rajagopalan said that compiler optimizations actually perform very well for event-based systems, and that implementation is really what matters. Event-based systems permit a decoupling between caller and callee, so it is easier to write an event-based "adaptive" program than a threadbased one. Isn't this a fundamental benefit? Rob responded that events do make it easier to perform composition and interpositioning, but that this can also be done in the thread model. Eric mentioned that Click is very configurable and runs as a single large thread.

Peter Druschel was skeptical that we can do P2P storage based on home-connected desktops, but that the alternative is not centralized systems. For example, one can reap the benefits of unused desktop systems within a large organization. Charles did not disagree with that.

This was followed by an exchange between Peter and Rodrigo Rodrigues about using so-called "scalable lookup" vs. some other organization for P2P file storage. Basically, Rodrigo pointed out that in a scenario where the individual nodes are very available/reliable and the network isn't giant, there is no need for scalable lookup and other considerations should take priority. Peter responded that having a large number of nodes and security implied the need for small lookup-state optimizations.

SESSION: POPPING & PUSHING THE STACK

Summarized by Ranjita Bhagwan

TCP Offload Is a Dumb Idea Whose Time Has Come

Jeffrey C. Mogul, Hewlett Packard Laboratories

TCP offload in the traditional sense violates performance requirements, has practical deployment issues, and targets the wrong applications. TCP Offload Engines (TOEs) impose complex interfaces and cause suboptimal buffer management. Moreover, lots of small connections overwhelm savings because of connection management. Event management is a problem. Lots of virtual resources need to be managed. Also, one of the main motivations for TOE has been that TCP implementation in the OS is bad.

However, it is no longer a dumb idea, because now we are offloading higher-level protocols onto hardware. The justification for offloading TCP is simply that you can't offload the higher-level protocols without also offloading TCP. The sweet spot for TCP offload is when the application uses very high bandwidth and has relatively low end-to-end latency, long connection durations, and relatively few connections (e.g., storage server access and graphics). Also, several economic trends favor TCP offload. One would like to replace special-purpose hardware with cheap commodity parts, such as 1- or 10-gig Ethernet. This helps because with these in place, operators have only one kind of fabric to provision, connect, and manage. Still, many challenges remain. Data copy costs still dominate, and busses are too slow. Zero copy and single copy seem too hard to adopt in commercial OSes. However, with the advent of RDMA, vendors want to ship RNICs in volume, allowing one kind of chip for all applications. It would mean cheaper hardware. There are also several upper-level protocols available, such as NFSv4 and DAFS. Still, many problems of TCP offload remain:

There are security concerns, and so far the benefits have been elusive. The new networking model may require changes to traditional OS APIs. Systems people need to give this due consideration.

TCP Meets Mobile Code

Parveen Patel, Jay Lepreau, University of Utah; David Wetherall, Andrew Whitaker, University of Washington

The authors address the problem of deployment of transport protocols by proposing an extensible transport layer, called XTCP. The main argument is that transport protocols, such as TCP, need a self-upgrade mechanism, and untrusted mobile code can help build such a mechanism. Several modifications to TCP, as well as alternative transport protocols, have been proposed. However, as with any new protocol, deployment is an issue. Currently, it takes many years before a new protocol or an extension can be used by applications: A new protocol or extension has to be approved by standards committees, implemented by OS vendors, and finally enabled-by-default at both ends of communication.

In the proposed solution, untrusted peers can upgrade each other with new transport protocols using mobile code. A typical usage scenario is that of a Web server. A Web server can download a high-performance version of TCP, after which it tells every client to download the same version from it. Then the client and the server can speak the upgraded version of TCP. This solution avoids all the steps of the deployment process that need approval and support from third parties, such as standards committees and OS vendors.

There are several challenges to building such an extensible layer, notably host and network safety. The presenter contrasted XTCP with "active networking" and argued that the domain of transport protocols is restricted enough that host and network safety challenges can be met without degrading performance.

Host safety is assured by providing memory protection and resource control. Memory protection is achieved by using Cyclone, a typesafe C-like language. The stylized memory-usage pattern of TCP extensions – no shared state between extensions and predictable ownership of data buffers – makes resource control possible using traditional runtime methods. XTCP uses the well-understood notion of TCP-friendliness as a measure of network safety. All extensions written using the XTCP framework are forced to conform to TCP-friendliness using the ECN nonce mechanism. In contrast, active networking had no such well-defined notion of network safety, and host safety in the face of arbitrary code was costly.

XTCP has been implemented in FreeBSD 4.7. Support for user-level transports is being developed currently.

Exploiting the Synergy Between Peer-to-Peer and Mobile Ad Hoc Networks

Y. Charlie Hu, Saumitra M. Das, Himabindu Pucha, Purdue University

There appear to be a number of similarities in the problems addressed by research in peer-to-peer and mobile ad hoc networking. One such area is that of routing. The speaker showed the similarity between the problems solved by Pastry and how it can be used in ad hoc networking, too. He described a new protocol, DPSR, which stores routing state in a manner similar to Pastry. This reduces routing state per node from O(N) to O(log N). DPSR uses node ID assignment, node state, routing, node join procedures, and node failure or out of reach in much the same manner as Pastry; inherits all DSR optimizations on source routes; and contains a number of additional optimizations related to Pastry's routing structures and operations.

Simulations of DPSR for a 50-node system show that the routing overhead of DPSR scales better than that of DSR. In short, DPSR outperforms DSR when the number of connections per source is greater than 1; performance is otherwise equivalent.

Panel Discussion

Bogdan Popescu asked Parveen Patel if you could use a signing mechanism to detect unresponsive connections. Parveen said that the nice thing about XTCP is that it works well without it. Bogdan said that then you could have DoS attacks.

Rob von Behren said that it would be very easy to do DoS on XTCP, such as mallocing large amounts of memory, using a lot of CPU time, etc. Parveen said that each malloc call is accounted for. Rob responded that there is the problem of DDoS. With a considerable number of nodes using a little too much memory, one could perform a DDoS attack. Parveen said that this is possible and the only way to avoid it is strict admission control.

Jeff Mogul said that you have to make sure that XTCP itself is not subvertible. Because if it is, then it is a very rich environment for spreading worms.

Peter Steenkiste said that in the early '90s, after six months of effort, he had decided that TCP offloading is no good. In general, enthusiasm for TCP offload seems lukewarm. He asked Jeff if it would take off. Jeff responded that he does believe that it will take off, mainly for commercial reasons. Having only one fabric to manage for data centers seems good. Peter said that there appears to be a contradiction: Earlier on, we wanted to move things to the software level, and now attempts are being made to move them to the hardware. Jeff said that switches are clearly a larger investment than NICs. So commoditizing the NICs would be good.

Geoff Voelker asked Charlie Hu about how much the benefits of his approach depended on the amount of shared source routes. Did he have a sense of the minimum degree of shared source

routes needed for DPSR to work? Charlie answered that so far, the sharing was small, but even in this scenario, DPSR does no worse than DSR. So it seems like a total gain over DSR.

SESSION: DISTRIBUTED SYSTEMS

Summarized by Amit Purohit

Scheduling and Simulation: How to Upgrade Distributed Systems

Sameer Ajmani, Barbara Liskov, MIT Laboratory for Computer Science; Liuba Shrira, Brandeis University

Sameer Ajmani presented a solution to upgrade distributed software automatically with minimal service disruption. He described a technique that uses a combination of centralized and distributed components. The infrastructure consists of three main components: scheduling functions tell the node when to upgrade; simulation objects enable communication among nodes running different versions; and transform functions change a node's persistent state from one version to a higher one.

Development Tools for Distributed Applications

Mukesh Agrawal, Srinivasan Seshan, Carnegie Mellon University

Mukesh Agrawal explained the motivation for his current research. He claimed that the lack of distributed applications is because of implementation difficulties. He identified routing table upgrades for distributed applications as one of the harder problems. He mentioned the ns-2 simulator as a tool that helps to compare design choices. And DHT is developing building blocks to help implement distributed systems. Then he pointed out some inherent flaws in the current approaches. Research mainly concentrates on the initial stages of the life-cycle of the applications, while his work mainly addresses the issues with later life-cycle stages.

Virtual Appliances in the Collective: A Road to Hassle-Free Computing

Constantine Sapuntzakis and Monica S. Lam, Stanford University

Constantine Sapuntzakis envisioned a computing utility that runs not only Internet services but highly interactive applications commonly run on desktop computers. On desktops, patches arrive frequently and there is much multiple-application sharing of such things as OSes and libraries; hence, application upgrades can disrupt other applications. Constantine argued that it is possible to borrow an idea from "network connected computer appliances" to improve the manageability and usability of computers. In the architecture of their framework, groups of virtual appliances are maintained by makers without user involvement. Cheaper hardware made virtualization an attractive option.

POST: A Secure, Resilient, Cooperative Messaging System

Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, and Dan S. Wallach, Rice University; Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra, Université Paris VI

A P2P solution was presented that interoperates seamlessly with a wide range of collaborative services by providing one serverless platform. It provides three basic services to applications: secure single-copy message storage; event notification; and single-writer logs that allow applications to maintain metadata. The claim was made that these features are sufficient to support a variety of collaborative applications.

Panel Discussion

Mike Swift asked Constantine Sapuntzakis about the cost of complex virtual appliances. He also noted that device drivers talk to hardware, hence couldn't be virtualized, and can crash if they are buggy. Constantine said future device drivers could be written in user-land and the problem could be solved. But if the application crashes, not much can be done.

Eric Brewer stated the view that the hard part is sharing information: Having separate virtual appliances for everything only works if they don't share any information, which means that the user must replicate all "shared" information by hand (as we do now with real appliances, e.g., setting the clock). The path of safe sharing leads you to shared segments as in Multics, including layers (rings) and call gates for protected calls into shared resources. The author replied that Multics has some problems and that they are planning to address them as well.

OUTRAGEOUS OPINIONS SESSION

Summarized by David Oppenheimer and Matt Welsh

In classic HotOS tradition, the Outrageous Opinions session consisted of a stream of short presentations, some serious, some mundane, some hilarious.

Val Henson argued against the use of checksums at all levels in a storage system versus end-to-end checksums at the application. Andrew Hume countered that it's good to have accountability at each level when something goes wrong in the system.

Matt Welsh presented "a brief history of computing systems research," in which he urged computer scientists to think about how their research can help to address social problems. He pointed out that computer scientists have always worked on improving life for computer scientists, focusing on improving their own day-to-day tasks. He suggested that computer scientists should think more about social problems rather than "How can I download porn and pirate music more efficiently?" In particular, he cited education and health care, particularly outside of the United States and Europe, as social problems that computer scientists could help tackle – for example, by empowering local government and

remote communities. Specific technologies he cited were peer-to-peer wireless networks for emergency and disaster response systems; censorship-proof electronic publishing of dissenting political opinions; sensor networks for environmental monitoring, precision agriculture, and inexpensive medical diagnostics; highly reliable power environments; and maintenance-free PCs.

Mendel Rosenblum talked about the similarities between micro-kernels and virtual machine monitors (VMMs) for running multiple untrusted environments on top of an operating system. Both provide isolation, resource management, and a small operating environment with simple interfaces, leading to high assurance. The key difference is what runs on top of each – for a VMM you don't need to port applications to run on it, whereas for a micro-kernel you do. He pointed out that despite this advantage for VMMs, academics and industry researchers are often interested in micro-kernels because, by not leveraging existing software, micro-kernels provide academics an opportunity to write lots of papers and industry an opportunity to define APIs, leading to an industry control point.

Mike Chen presented "two easy techniques to improve MTTR": redirect users' attention to what's still available when something becomes unavailable (e.g., "Please read this new privacy policy"), and blame it on the user (e.g., "Did you forget your password? Please try again."). He pointed out that by tricking the user, perceived availability can easily be increased.

Timothy Roscoe railed against the construction of scalable systems, claiming that systems should only scale as far as needed and no further. For example, email does not need to scale globally; who needs to send an email to everybody? After all, whitelisting your email to reduce spam doesn't scale, but it works. Timothy used Google as an example of a system where extra scalability only concentrates power. He cited libraries as a non-scalable alternative to Google.

Dan Wallach talked about all of the recent work on virtualizing resources and getting multiple virtual machines to share resources, such as shared libraries and the OS kernel. He proposed an alternative to these approaches, a radical concept called a "process."

Eric Brewer issued a call for IT research for developing regions. He made five claims: (1) there is a need for a direct attack by developing new devices rather than giving developing regions hand-me-down machines, which are a bad fit on cost, power, user knowledge, administration, and user literacy; (2) there is a need for infrastructure to support thousands of projects which are currently not sharing any infrastructure; (3) building IT for developing regions is economically viable, in that there is a market of 4 billion people, but IT must create income for its users (e.g., by offering a franchise model akin to that used to provide cell phone service to rural Bangladesh) because users do not have disposable income; (4) the time is right with the availability of five-dollar 802.11 chipsets, systems on a chip, low-power designs, and solar panels; and (5) this work can have a big impact by reducing poverty and disease and improving the environment, by providing developing regions with a source of income that they can in turn use to improve their standard of living, stability, and security. Lots of research directions here, including very low-power and low-cost wireless communications, new speech-based user interfaces, network proxies, and sensors.

George Candea discussed why he believes that wide-area decentralized systems such as peer-to-peer networks are "a good idea whose time has passed." He argued that such systems are hard to build, test, debug, deploy, and manage, and that they have little economic incentive beyond "lack-of-accountability" applications. He suggested that the principles learned from building wide-area distributed systems – strong compo nentization, using open protocols, loose coupling, reducing correlated faults through diversity, and writing components while keeping emergent behaviors in mind – should be used to build highly dependable "distributed" systems within the data center. He summarized by saying, "Don't distribute centralizable apps into the wide-area network, take the good ideas from distributed systems and apply them in the system-area network."

Geoff Voelker presented a novel idea based on the notion of value prediction from hardware architecture: "result prediction." Rather than running the program, we can simply guess the results, leading to excellent speedup potential!

Emmett Witchel asked whether there is a use for anti-optimization. One use he suggested was to allow users to specify in advance the amount of resources a computation takes, possibly eliminating covert channels. This would be accomplished by intentionally adding delay loops to code to use all available CPU and by spreading allocated memory all over the address space.

Ethan Miller proposed SCUBA: Scalable Computing for Underwater Biota Assessment, in which 802.11 networks would be deployed on coral reefs.

Sameer Ajmani suggested that systems researchers consider work in computational biology: "We help biologists, then they help thousands of people through pharmaceuticals, genetics, etc. – it's easier than sending computers to Africa." As specific examples of computational biology problems that can directly apply well-known computer science algorithms, he cited string alignment (dynamic programming) and database searches for genes (hashtable with 2-tuples and 3-tuples). Andrew Hume added that the National Institutes of

Health also has more money than the National Science Foundation.

Armando Fox called for a bet whether a peer-to-peer application would exist before the next HotOS, that would make more sense (economically and technically) to deploy as a peer-to-peer system than as a centralized service. Seven people in the audience said yes, 17 said no. Armando offered to bet someone (Armando taking the "no" side) for a case of alcohol valued less than the conference registration fee in 2005, but there were no takers.

SESSION: WHEN THINGS GO WRONG

Summarized by Amit Purohit

Crash-Only Software

George Candea and Armando Fox, Stanford University

George Candea explained how to build Internet services that can be safely and cheaply recovered by crash-rebooting minimal subsets of components. He stated that most downtime-causing bugs are transient and intermittent and that it is not feasible to guarantee that an application can never crash. For recovery-safe applications, recovery can be too long. Crash-only software achieves crash-safety and fast-recovery by putting all the important non-volatile state outside the application components into crash-only state stores. For systems of crash-only components to be crash-only, the components must be decoupled from each other, from the resources they use, and from the requests they process. He conceded that steady-state performance of crash-only systems may suffer, but argued that (1) the overall goal is to maximize the number of requests successfully served, not to serve them fast and then be unavailable for a long time, and (2) that techniques will evolve that will improve performance of crash-only systems, the way compilers improved the performance of programs written in high-level languages.

The Phoenix Recovery System: Rebuilding from the Ashes of an Internet Catastrophe

Flavio Junqueira, Ranjita Bhagwan, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker, University of California, San Diego

This presentation explained the design of an operative, distributed remote backup system called the Phoenix. Operating systems and user applications have vulnerabilities. A large number of hosts may share vulnerabilities and this can result in major outbreaks. Phoenix uses a strategy with attributes and cores. By replicating data on a set of hosts with different values for each attribute, it is possible to reduce the probability of error to near zero. In the Phoenix system there is no single point of failure; copying with a large number of requests is achieved by exponential backoff.

Using Runtime Paths for Macroanalysis

Mike Chen, Eric Brewer, University of California, Berkeley; Emre Kiciman, Armando Fox, Stanford University; Anthony Accardi, Tellme Networks

Mike Chen emphasized the benefits of microanalysis, namely latency profiling, failure handling, and detection diagnosis. He introduced the concept of "runtime path analysis, where paths are traced through software components and then aggregated to understand global system behavior via statistical inference." Runtime paths are also used for failure handling and to "diagnose problems all in an application-generic fashion." The group explained that their work could be extended to P2P message paths, event-driven systems, forks, and joins.

Magpie: Online Modeling and Performance-Aware Systems

Paul Barham, Rebecca Isaacs, Richard Mortier, and Dushyanth Narayanan, Microsoft Research Ltd, Cambridge, UK

Magpie is "a modelling service that collates traces from multiple machines . . ., extracts request-specific audit trails, and constructs probabilistic models of request behaviour." The presenter mentioned that workload description and hardware modeling could be used to predict performance. It is possible to augment the system by getting feedback from past models.

Using Computers to Diagnose Computer Problems

Joshua A. Redstone, Michael M. Swift, Brian N. Bershad, University of Washington

Redstone described "building a global scale automated problem diagnosis system that captures the . . . workflow of system diagnosis and repair." A computer generates search terms and locates problem reports. It stores problem reports in a canonical global database. When a problem occurs the computer detects symptoms and searches the problem database. Joshua argued that expending more effort in building the database could be a key for cheaper diagnosis. He mentioned that the main challenge lies in creating a database structure in which it is possible to meet user expectations.

Panel Discussion

Jeff Mogul was concerned about whether the system effected time to recovery. The author said there aren't critical time recovery requirements, as it is more important to get the data back. He pointed out that it is also possible to make it faster by adding redundancy. But it is a trade-off between storage and time. Somebody from MIT also had a question regarding the deployment of the Phoenix system. The author said it seems reasonable if there are enough hosts running diversified OSes. Ranjitha from UCSD asked Joshua Redstone what would be the motivation for people to fill the database. Joshua said an organization uses external support and, hence, people would find it easier to resolve problems at the cost of the effort. Mike Jones proposed an alternative scheme that asks users for requests and then

posts any solutions. Joshua was not sure how to maintain the database. Mike said, you can save the entire request and response and then infer offline. Jay Lepreau asked George Candea about the impact of crash-only software on throughput. George said it's likely to be lower, because of the inherently distributed approach (loose coupling, explicit communication, etc.) to building the system, but with time performance will improve, as in the transition from assembly languages to high-level languages. High-level languages enabled a qualitative jump in the types of software able to be written, even if the process was slower than writing in assembly. He also argued that "goodput" (throughput of successfully handled requests) is more important than absolute throughput.

SESSION: PERFORMANCE OPTIMIZATION

Summarized by Ranjita Bhagwan

Using Performance Reflection in Systems Software

Robert Fowler and Alan Cox, Rice University; Sameh Elnikety and Willy Zwaenepoel, EPFL

The main idea of this work is to use application-independent measures such as hardware instrumentation mechanisms and general system statistics to adapt system behavior. Performance indicators such as TLB misses and cache misses can be used to measure overhead, while bytes sent to a network card and flop rate can be used to measure productivity. Productivity and overhead are used to determine if the system needs to be tuned. Sameh Elnikety showed results on how server throttling of mySQL using the TPC-W workload succeeded in keeping the throughput at the maximum level while load increased, whereas, without using reflection, the throughput dropped at higher loads.

Cassyopia: Compiler Assisted System Optimization

Mohan Rajagopalan and Saumya K. Debray, University of Arizona; Matti A. Hiltunen and Richard D. Schlichting, AT&T Labs–Research

The main idea of Cassyopia is to combine program analysis and OS design to do performance optimizations. While the compiler has a local perspective of optimizations, the OS has a more general view. Cassyopia merges the two and tries to bring about a symbiosis between the OS and the compiler. An example of this is system call optimization, which profiles system call sequences and clusters them together using compiler techniques. The clustered system calls are called multi-calls. With OS support for multi-calls, performance can be improved and, according to preliminary results, significant savings obtained.

Cosy: Develop in User-Land, Run in Kernel-Mode

Amit Purohit, Charles P. Wright, Joseph Spadavecchia, and Erez Zadok, Stony Brook University

User applications are only allowed restricted access, which causes a lot of crossings of the user-kernel boundary. To prevent this, Amit Purohit proposed a compound system call (Cosy) in which one can execute a user-level code segment in the kernel. The authors have a modified version of gcc that uses Cosy. Kernel safety is ensured by limiting kernel execution time; x86 segmentation and sandboxing techniques can also be used. The performance benefits of Cosy have been evaluated, and 20–80% performance improvements are reported.

Panel Discussion

Mike Swift said that using all the compiler techniques from Cassyopia for kernel execution might be one way to proceed. Mohan Rajagopalan clarified that they do not plan to move any user-level code into the kernel, since he believed that interpretation in the kernel had high overhead. They primarily wanted to reduce the boundary-crossing cost, and, hence, the ideologies are not the same. Amit Purohit said that interpretation in the kernel is a bottleneck, and so they use a small interpreter. Checking pointers would cost a lot, and so they are using TLBs. This has some overhead, but it's a one-time check. Mohan brought up the point that the aim of Cassyopia is to apply optimizations that are quite obvious but have not yet been done.

Ethan Miller said that the TLB overhead in Cosy could be unavoidably high. Amit said that is true, but the savings they are getting are a lot more than the TLB overhead.

Margo Seltzer hit the nail on the head, by saying that what matters finally is the kernel-user API. All this work on extensible OSes and moving code across the boundary is probably done because the kernel-user API needs to be revisited. So let's fix the API. Applause.

Andrew Hume said that for the applications he has looked at, apart from zero-copy and I/O, there is not much to be gained by putting code into the kernel. Apart from the stated scenarios, the chances of using a multi-call are small. Sometimes, reassurance outweighs performance benefits. Mohan said that they are also looking at smaller devices, such as cell phones. All devices are resource constrained. In these cases, there is also the issue of energy savings apart from performance. Eric Brewer asked why, for small devices, such as in sensor networks, you would even want a kernel boundary. Mohan answered that cell phones and iPAQs can now have JVMs running on them. They are not targeting reprogrammable devices.

Matt Welsh asked why one would care so much about performance on an iPAQ.

Timothy Roscoe said that since there are different kinds of devices, there should be different OSes for them, and then the question would be where to put the ker-

nel boundary, if any, on them. Whether there is a generic answer to that question is still unclear.

SESSION: STORAGE 1

Summarized by David Oppenheimer

Why Can't I Find My Files? New Methods for Automating Attribute Assignment

Craig A.N. Soules and Gregory R. Ganger, Carnegie Mellon University

Craig Soules described new approaches to automating attribute assignment to files, thereby enabling search and organization tools that leverage attribute-based names. He advocates context analysis to augment existing schemes based on user input or content analysis. Context analysis uses information about system state when the user creates and accesses files, using that state to assign attributes to the files. This is useful because context may be related to the content of the file and may be what a user remembers when searching for a file. Google has proven the usefulness of context analysis; it chooses attributes for a linked site by using the text associated with the link, and it analyzes user actions after a search to determine the user's original intent in the search. However, the kind of information Google's context analysis relies on cannot be applied directly to file systems. In particular, information such as links between pages does not exist in traditional file systems, and individual file systems do not have enough users or enough "hot documents" to make Google-like context statistics useful.

Soules described access-based context analysis, which exploits information about system state when a user accesses a file, and interfile context analysis, which propagates attributes among related files. The former relies on application assistance or existing user input (e.g., file names), while the latter relies on observing temporal user access patterns and content similarities and differences between potentially related files and versions of the same file. Based on a trace analysis of usage of a single graduate student's home directory tree over a one-month period, Soules concluded that a combination of the techniques he proposes could be useful for automatically assigning attributes. For example, a Web browser can relate search terms to the document the user ultimately downloads as a result of the search; files created and accessed in a single text editor session can be considered related; and attributes about documents used as input to a distiller such as LaTeX or an image manipulator program can be distilled for attachment as attributes of the output file. Soules also found that examining temporal relationships between file accesses in the trace successfully grouped many related files.

Soules stated that as future work he is investigating larger user studies, mechanisms for storing attribute mappings, appropriate user interfaces, and how to identify and take advantage of user context switches, e.g., users moving from one program to another.

Secure Data Replication over Untrusted Hosts

B.C. Popescu, B. Crispo, and A.S. Tanenbaum, Vrije Universiteit, Amsterdam, The Netherlands

B.C. Popescu described a system architecture that allows arbitrary queries on data content that is securely replicated on untrusted hosts. This system represents an improvement over systems based on state signing, which can support only semi-static data and predefined queries, and systems based on state machine replication, which require operations to be replicated across multiple machines. In the authors' system, every data item is associated with a public-private key pair; the private key is known only to the content owner, and the public key is known by every client that uses the data. There are four types of servers: master servers that hold copies of content and are run by the content owner; slave servers that hold copies of data content but are not controlled by a content owner and thus are not completely trusted; clients, which perform read/write operations on content; and an auditor server, described later. The master servers handle client write requests and lazily propagate updates to slave servers. Master servers also elect one of themselves to serve as an auditor, which performs background checking of computations performed by slaves, taking corrective action when a slave is found to be acting maliciously. Slave servers handle client read requests; they may use stale data to handle requests, but clients are guaranteed that once a time parameter maxLatency has passed since a write was committed at a master, no other client will accept a read that is not dependent on the write. All content in the system is versioned; the content version of a piece of data is initialized to zero when it is created and is incremented each time the data item is updated.

The key challenge in building this system is to enable clients to feel safe having their queries handled by untrusted slave hosts. This is accomplished probabilistically, by allowing clients to send the same request to a (trusted) master and (untrusted) slave when they wish, and to compare the results. When a slave returns the result of a read, it attaches a signed "pledge" packet containing a copy of the request, the content version timestamped by the master, and the secure hash of the result computed by the slave. If the slave returns an incorrect answer, the "pledge" packet can be used as proof of the slave's malfeasance. This probabilistic checking mechanism is augmented by an auditing mechanism in which after a client accepts a result from a slave, it forwards the slave's "pledge" packet to a special auditor server. The auditor server is a trusted server that does not have a slave set and serves just to check the validity of "pledge" packets by re-executing the requests and verifying that the secure hash of the result

matches the secure hash in the packet. The auditor is expected to lag behind when executing write requests, executing writes only after having audited all the read requests for the content version preceding the write.

Palimpsest: Soft-Capacity Storage for Planetary-Scale Services

Timothy Roscoe, Intel Research at Berkeley; Steven Hand, University of Cambridge Computer Laboratory

Timothy Roscoe described Palimpsest, a "soft-capacity storage service for planetary-scale applications." Palimpsest is designed to serve as a storage service for ephemeral data from planetary-scale applications running on a shared hosting platform like PlanetLab, XenoServers, or Denali. Examples of the type of data to be stored are static code and data for services, application logs of various sorts, and ephemeral system state such as checkpoints. Despite the temporary nature of the data, it must be highly available during its desired lifetime, thus making single-node local disk storage unsuitable. Traditional file systems such as NFS and CIFS provide facilities unnecessary for planetary-scale applications, and don't meet service provider requirements such as space management, billing, and security mechanisms that allow users to store their data on a shared infrastructure without having to trust the infrastructure provider. Palimpsest aims to provide high data availability for limited periods of time, data protection and resistance to denial-of-service attacks, flexible cost/reliability/performance trade-offs, charging mechanisms that make sense for service providers, capacity planning, and simplicity of operation and billing. To achieve these goals it uses soft capacity, congestion-based pricing, and automatic space reclamation.

To write a file, a Palimpsest client erasure codes the file, encrypts each resulting fragment, and sends each encrypted fragment to a fixed-length FIFO queue at the distributed hashtable node corresponding to the hash of the concatenation of the file name and the fragment identifier. To retrieve a file, a client generates a sufficient number of fragment IDs, requests them from the block stores, waits until a sufficient number of the requested fragments are returned, decrypts and verifies them, and recreates the original file. Because the queues are fixed-length, all files stored in Palimpsest are guaranteed to eventually disappear. To keep a file alive, the client periodically refreshes it, and to "delete" the file, the client simply abandons it. The key to predictable file lifetimes is the "time constant" (T) associated with each block store; T measures how long it takes a fragment to reach the end of the queue and disappear. Clients piggyback requests for information about T on read and write requests, and block stores provide a recent estimate of T by piggybacking on responses. Palimpsest providers charge per (fixed-length) write transaction. Clients can use information about each block store's T value and fee per write transaction to flexibly trade off data longevity, availability, retrieval latency, and robustness. Clients pay providers using anonymous cash transactions. Denial-of-service attacks are discouraged by charging for writes. Providers can perform traffic engineering by advertising values of T that deviate from the true value. Congestion pricing is used to encourage users to attain an efficient write rate.

Panel Discussion

Andrew Hume asked Timothy Roscoe whether a simpler scheme than Palimpsest could be used to store ephemeral files just like regular files and cycle them using a generational scheme, eventually deleting the files that graduate from the oldest generation. Roscoe responded that Palimpsest provides an easy charging and pricing mechanism, while standard network file systems like NFS do not. Val Henson asked Popescu what he thought of the "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two" talk, in light of the fact that his system is targeted toward storing data on untrusted hosts. Popescu responded that they're more interested in environments in which hosts are less transient than in standard peer-to-peer networks. Jeff Chase observed that Palimpsest clients have no control over the placement of their data, and he asked Roscoe whether he thought that was significant. Roscoe responded that selecting specific nodes on which to store data could be provided by an orthogonal mechanism. Benjamin Ling asked whether widespread adoption of Palimpsest would be hindered by the lack of hard guarantees about data longevity. Roscoe responded that legal contracts akin to service level agreements could be layered on top of Palimpsest to ease users' concerns about the inherent risk of data loss. Furthermore, this is really a futures market: Third parties can charge premiums for providing guarantees and taking on the risk of data loss themselves.

SESSION: TRUSTING HARDWARE

Summarized by Matt Welsh

This session turned out to be the most controversial of the conference, as two of the three talks discussed the use of secure hardware and systems such as Microsoft's Palladium architecture.

Certifying Program Execution with Secure Processors

Benjie Chen and Robert Morris, MIT Laboratory for Computer Science

Benjie Chen is interested in the potential uses for trusted computing hardware other than digital rights management (DRM). Since all PCs may include this hardware in the future, he is interested in exploring the hardware and software design for such systems. His running example was secure remote login, such as from a public terminal at an Internet cafe, where the client machine can attest to the server that it is running unadulterated software (OS, SSH client, etc.). Of course, this does not preclude low-

tech attacks such as a keyboard dongle that captures keystrokes, but that is outside of the immediate problem domain.

Benjie presented an overview of the Microsoft Palladium (or Next Generation Secure Computing Base) architecture, which uses a secure "Nexus" kernel and a secure chip that maintains the fingerprints of the BIOS, bootloader, and Nexus kernel. A remote login application would send an attestation certificate, generated by Nexus and the secure chip, to the server. The issues here are how to keep the Nexus kernel small and how to verify the OS services (such as memory paging or the network stack). Some ways to improve Palladium's security and verifiability were discussed, such as using a small micro-kernel that allows attestation of all OS modules above it, as well as a flexible security boundary (where some, not all, of the OS modules are verified). There is a connection with the XOM secure processor work, which prevents physical attacks on DRAM by storing data in encrypted form in memory and only decrypting it into a physically secure cache. Borrowing some of these ideas, one could run the micro-kernel within the secure processor that authenticates all data transfers to DRAM; the application, hardware drivers, network stack, etc., could all be encrypted in DRAM.

Hardware Works, Software Doesn't: Enforcing Modularity with Mondriaan Memory Protection

Emmett Witchel and Krste Asanović, MIT Laboratory for Computer Science

Emmett Witchel proposed the use of efficient, word-level memory protection to replace the use of page- or segment-based protection mechanisms. This is motivated by the use of fine-grained modules in software, with narrow interfaces in terms both of APIs and of memory sharing. He argued that safe languages are not the answer, in part because it is difficult to verify the compiler and runtime system. Rather, allowing hardware protection to operate at the word level is much simpler and permits a wide range of sharing scenarios. For example, when loading a new device driver into the kernel, the MMP hardware would be configured to permit the module to access its own code and data, as well as to make calls to other modules and share very fine-grained memory (e.g., part of a larger data structure in the kernel). Some of the challenges involve cross-domain calls through call gates; dealing with the stack (as no single protection domain "owns" the stack); and automatically determining module boundaries through information already present in code, such as symbol import/export information in kernel modules. Other potential uses include elimination of memory copies on system calls, specialized kernel entry points, and optimistic compiler optimizations (e.g., write-protect an object and run cleanup code if a write fault occurs).

Flexible OS Support and Applications for Trusted Computing

Tal Garfinkel, Mendel Rosenblum, Dan Boneh, Stanford University

Tal Garfinkel's talk returned to the question of using secure hardware for applications other than DRM, and shared much of the motivation and background of Benjie's talk. The core problem with open platforms (as opposed to closed platforms such as ATMs and cell phones) is that applications can be deployed across a wide range of existing hardware but it is difficult to manage trust. Tal proposed the use of virtual machine monitors as a potential solution to providing a trusted OS environment. For example, the VMM can run either an "open box" VM (such as a standard OS) or a "closed box" VM (a trusted system).

One closed box VM might be a virtual Playstation game console, which prevents cheating in a multiplayer game through attestation. Another potential application could be a distributed firewall, where one could push a customized firewall into the VMM to protect the network from the host, by preventing port scanning or IP spoofing, for example, or enforcing connection rate limits. Tal also discussed applications to reputation systems and third-party computing (à la SETI@Home). He concluded the talk with a review of current efforts in this area, including TCPA, Palladium, and LaGrande.

Panel Discussion

The political issues surrounding trusted computing platforms raised a number of interesting – and heated – questions from the audience. Dan Wallach started off by describing the recent XBOX hack, where that system was supposedly trusted hardware. Tal and Andrew Hume countered that there are no guarantees of correctness, just trade-offs in terms of risk assessment. Mendel suggested that cheating at Quake was not a big concern for industry, so this was not of paramount concern in the XBOX hack.

Timothy Roscoe was disturbed that the three speakers seemed to be too much in agreement, and raised the question of big protection domains (i.e., VMs) versus tiny protection domains (i.e., Mondrian memory protection). They didn't take the bait, though, and Tal said that these approaches were not mutually exclusive.

Jay Lepreau raised the concern that nobody has yet demonstrated an entire (real) operating system based on the micro-kernel model. He again argued that MPP does not solve the whole domain protection issue, since control flow protection is just as important as memory protection. Emmett admitted that there is some complexity involved, but by making memory protection domains both finer grained and more explicit, programmers have to think about them more carefully and will document them in the code. Jay argued that chopping up a monolithic system in a "half-assed way" makes it more complex, but Emmett argued that most sys-

tems software is already making use of well-defined modules, simply without strong protection between them.

Val Henson returned to the trusted computing discussion and argued that these platforms were more useful for restricting rights (as with DRM) than for giving us more rights. Benjie argued that the bleak view is that this hardware is going to get pushed out regardless, so we should be considering how to use it for something other than DRM. Tal concurred and said that this work was also about intellectual honesty.

Jay argued that Tal and Benjie were really just giving cover to the real commercial driver for this technology, which is DRM. Mike Swift wanted to know where the line between research and the corporate world should be drawn, and whether our community should support this work at all. Tal mentioned again that their work is just bringing more information to the table, but Mike drew an analogy with nuclear weapons research, claiming that expanding knowledge is not the only side effect of this work.

Dirk Grunwald wondered whether calling this "trusted computing" was like the rebranding of "out-of-order execution" to "dynamic execution" – trusted computing cannot deal with hardware attacks, so consumers may be misled into believing it's more safe. Tal pointed out that this is no different from calling a protocol "secure."

Jeff Mogul made the points that technology tends to reinforce existing power structures, and that the issue with trusted computing is not about security but rather, whether the technology reinforces existing power relationships or diffuses them. He summarized, "Are you advocating a system that helps the powerful or that helps the weak?" Eric Brewer claimed that he didn't want "trusted" software on his PC, since that only enforces a particular kind of trust relationship between two parties. He would rather have control over his trust relationships, but trusted computing platforms remove that control. Tal admitted that there is a real concern about using Palladium as an industry control point. Timothy wrapped up the session by pointing out that this discussion had been rather "sterile" and had not touched on any of the issues of politics, lawmaking, or market forces surrounding the DRM and trusted-technology debate.

SESSION: PERVASIVE COMPUTING

Summarized by Ranjita Bhagwan

Sensing User Intention and Context for Energy Management

Angela B. Dalton and Carla S. Ellis, Duke University

Angela Dalton spoke about the use of low-power sensors to monitor user behavior and to reduce system energy consumption. She described a case study called FaceOff, which uses a camera to perform image capture. Following face detection, the information is fed into a control loop that does the system-level energy management, by turning off the display. The authors have built a prototype of this, which uses image capture and skin detection. The authors have determined that FaceOff can provide significant energy savings. The authors also describe various ways of increasing the system's responsiveness and describe several optimizations with that objective.

Access Control to Information in Pervasive Computing Environments

Urs Hengartner and Peter Steenkiste, Carnegie Mellon University

Locating people in a pervasive computing environment can be done at many levels and granularities. But there should be an access control mechanism for such information. Access control mechanisms for conventional information, however, cannot be used as is for such environments. Urs Hengartner described a people locator service that uses digital certificates for defined location policies. The authors use three design policies: identify information early, design policies at the information level, and exploit information relationships. Their approach is to use a step-by-step model, where validation of a client node is done at every server node. They also use a policy description language and an information description language in their people locator service.

Privacy-Aware Location Sensor Networks

Marco Gruteser, Graham Schelle, Ashish Jain, Rick Han, and Dirk Grunwald, University of Colorado at Boulder

Marco Gruteser gave a brief description of a system that uses sensor networks to gather information while maintaining some degree of anonymity. Describing the problem, Marco said that sensor networks could be used to identify precise location of certain entities, and this could be undesirable. With an anonymous data collection scheme, you do not need to negotiate a privacy policy, and the risk of accidental data disclosures is reduced, since databases do not store this information anymore. The author described the notion of k-anonymity in database systems, and said that it could be applied to location information.

Panel Discussion

Urs Hengartner asked Angela Dalton whether she had thought about turning off only parts of the screen if there were multiple windows open but only one active? Angela said that there is related work that deals with energy adaptive displays. Presently this cannot be done. But you could use some form of gaze tracking to do this. Andrew Hume asked whether the partial screen turn-off works at the hardware level. Angela responded that currently there is no hardware that does that. New kinds of displays are assumed. But you would still need to control it through the system.

Andrew Hume commented that this could be used for covert operations: for example, a laptop screen shutting down would indicate that there is no one close to it. Wouldn't the security aspect be that you could detect location to some extent? Angela said that larger networks could do this, and yes, there are lots of security implications.

Val Henson asked what the trend of built-in sensors is. Angela replied that most devices can now have built-in cameras. In general, these sensors are low-power, cheap and increasingly pervasive, especially the cameras.

Andrew Hume asked what kind of camera resolution is needed to make this work well. Angela said that the detection method currently is skin color based, and you could do this even with a low-resolution black-and-white camera.

Geoff Voelker asked Urs if he had thought of foreign users coming into an administrative domain. Urs responded that since they use SPKI/SDSI digital certificates, they do not require a PKI and have the benefit that they could give access to anyone.

Val Henson asked Marco how you decide what the shape of the location is. Marco said that the current assumption is that the sensors are pre-configured in a room which has a certain room ID, the same applying to a building, and so on.

Jay Lepreau asked Angela whether she had data on how people use laptops so she could evaluate how well her scheme would work with these user behavior patterns. Angela said that more and more people are using laptops as their primary system. Moreover, energy awareness is important generically. However, this question is valid for cell phones, PDAs, etc., which could have widely varying usage patterns. Jay said that the power management on his laptop is frustrating, because it is stupid. It would be good to have a diagnostic tool, with the user being able to guide the system to some extent. Has Angela considered providing the user with a diagnostic tool? Angela said that you can imagine a user interface, that could be used to measure the annoyance factor of the power management. Yes, there can be problems when things kick in at the wrong time.

SESSION: STORAGE 2

Summarized by David Oppenheimer

FAB: Enterprise Storage Systems on a Shoestring

Svend Frølund, Arif Merchant, Yasushi Saito, Susan Spence, and Alistair Veitch, Hewlett Packard Laboratories

Alistair Veitch of HP Labs presented "FAB: Federated Array of Bricks." He described a project that is aimed at making a set of low-cost storage bricks behave, in terms of reliability and performance, like an enterprise disk array, but at lower cost and with greater scalability. The FAB array is built from bricks, each of which consists of a CPU, memory, NVRAM, a RAID-5 array of 12 disks, and network cards. Clients connect to the array using a standard protocol such as iSCSI, Fibre Channel, or SCSI, and the storage bricks communicate amongst themselves using a special FAB protocol running on Gigabit Ethernet. The goals of the array are zero data loss, continuous availability, competitive performance, scalable performance and capacity, management as a single entity, online upgrade and replacement of all components, and higher-level features such as efficient snapshots and cloning. The principal research challenges are failure tolerance without losing data or delaying clients, asynchronous coordination that does not rely on timely responses from disks or the operating system, and the ability to maximize performance and availability in the face of heterogeneous hardware. The techniques FAB incorporates to achieve these goals are a quorum-based replication scheme, dynamic load balancing, and online reconfiguration.

After outlining FAB's goals and the high-level techniques used to achieve those goals, Veitch described the quorum-based replication scheme used to achieve reliability. It uses at least three replicas for each piece of data, and reads and writes a majority of replicas. It survives arbitrary sequences of failures, achieves fast recovery, and can be lazy about failure detection and recovery as opposed to needing to do explicit fail-over. The array configuration that the designers envision achieves a mean time to data loss of about a million years, which Veitch described as "at the bottom end of what's acceptable."

The Case for a Session State Storage Layer

Benjamin C. Ling and Armando Fox, Stanford University

Benjamin Ling presented "SSM, a recovery-friendly, self-managing session state store." SSM is a storage system specialized for storing session state typically associated with user interactions with e-commerce systems. The system assumes a single user making serial access to semi-persistent data. Ling explained that existing solutions such as file systems, databases, and in-memory replication exhibit poor failure behavior, recovery performance, or both, and that they are difficult to administer and tune. SSM is designed to be recovery friendly, meaning it can recover instantly without data loss, and self-managing in terms of handling overload and performance heterogeneity of components.

SSM is based on a redundant, in-memory hashtable distributed across nodes called bricks and on stateless client stubs, linked in with application servers, that communicate with the bricks. Bricks consist of RAM, CPU, and a network interface (no disk). SSM uses a redundancy scheme similar to quorums. A stub writes to some N random nodes (four in Ling's example) and waits for the first M of the writes to complete (two in Ling's example); this scheme is

used to avoid performance coupling. The remaining writes are ignored. Reads are issued to a single brick. SSM is "recovery friendly" in that no data is lost as long as no more than M-1 disks in a write quorum fail. Moreover, state is available for reading and writing during a brick failure. SSM is "crash-only," using no special-case recovery code; when a brick is added to the system or returns to service after a failure, it is simply started up without any need to run a recovery procedure. SSM is "self-managing" in that it dynamically discovers the performance capabilities of each brick, as follows. Each stub maintains a count of the maximum allowable inflight requests to each brick, using additive increase to grow this window upon each successful request and multiplicative decrease to shrink the window when a brick operation times out. If an insufficient number of bricks are available for writing, either due to failures or due to a full window, the stub refuses the request issued by the client of the stub, thereby propagating back-pressure from bricks to clients.

Towards a Semantic-Aware File Store

Zhichen Xu, Magnus Karlsson, and Christos Karamanolis, Hewlett Packard Laboratories; Chunqiang Tang, University of Rochester

Zhichen Xu explained that storage systems are in some ways an extension of human memory, but that computer-based storage systems have been "dumb" because, unlike humans, they do not associate meanings (semantics) with the data that is stored. For example, when humans search, they consider abstract properties and relationships among objects, and when they "store" data, they may group objects into categories or record only the differences between objects. Moreover, humans discover the meanings of objects incrementally. These observations motivate the incorporation of versions, deltas, and dependencies among objects stored in the semantic-aware file store.

The semantic-aware file store seeks to create a framework that captures and uses semantic information for fast searching and retrieval of data; stores data efficiently by using data compression based on semantic relationships of data; provides high performance through semantic data hoarding, data placement, replication, and caching; and enables highly available data sharing by balancing consistency and availability according to the data semantics. The semantic-aware file store uses a generic data model based on RDF to capture semistructured semantic metadata. The challenges Xu described are identifying the common semantic relations of interest, finding ways to capture semantic information, finding ways to handle dynamic evolution of semantics, and defining the tools and APIs that users and applications require.

Panel Discussion

Margo Seltzer asked Alistair Veitch whether in 10 years HotOS will have a paper explaining why FABs are just as expensive as today's storage arrays. Veitch answered, "I hope not, but who knows."

Andrew Hume asked Veitch whether the FAB design makes it difficult to predict performance as compared to a disk connected to a single machine, especially in the face of failures, due to the large number of potential interactions among bricks in the FAB. Veitch responded that the performance can in fact be modeled as a function of factors such as the overhead of doing a disk I/O, the number of nodes, the desired mean time to data loss, and so on. He added that even today no storage system gives you hard-and-fast performance guarantees, and that the more difficult question is how to model overall system reliability and the optimal trade-offs among design parameters.

Rob von Behren asked Veitch whether he had considered using non-RAID disks instead of RAID arrays inside each brick. Veitch responded that by using RAID, the mean time to data loss is controlled by the failure rate of each brick rather than the failure rate of individual disks, thereby increasing reliability. He added that it's very difficult to come by good numbers on actual failure rates of system components, so estimating overall reliability is difficult.