conference reports

THANKS TO OUR SUMMARIZERS

FAST '09: 7th USENIX Conference on File and Storage Technologies

San Francisco, CA
February 24–27, 2009

OPENING REMARKS AND AWARDS

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

Program Chairs Margo Seltzer and Ric Wheeler opened FAST '09 by thanking the contributors and stressing the value of interaction between students and attendees. Best Paper awards were presented to "CA-NFS: A Congestion-Aware Network File System" by Batsakis, Burns, Kanevsky, Lentini, and Talpey of NetApp and Johns Hopkins, and to "Generating Realistic Impressions for File System Benchmarking" by Agrawal, Arpaci-Dusseau, and Arpaci-Dusseau of the University of Wisconsin, Madison.

Next, Garth Gibson took the floor to present the 2009 IEEE Reynold B. Johnson Information Storage Systems Award to Marshall Kirk McKusick. McKusick was recognized "for fundamental contributions in file system design, mentoring file system designers, and disseminating file system research." In his acceptance speech, McKusick stressed two themes: first, the collaboration between hardware and software experts, and second, the lessons drawn from his work on the Berkeley Fast File System (FFS).

In thanking the awards committee, McKusick praised their equal consideration of hardware and software nominees, a trend he hoped would continue. He stressed that it is necessary to incorporate both sides of the hardware and software interface with respect to storage. As an example, he pointed to what he termed the "don't lie to me" bit, which tells mechanical disk drives to confirm that data is written only when it actually reaches the persistent medium. He also pointed to the FAST conference itself as a unique forum in that it draws heavily from a diverse community of hardware, software, academic, and industry experts. This characteristic, he argued, was key to the conference's success.

Reflecting on his own work, McKusick began by noting that FFS was initially created in a "target rich environment." Because comparable systems were so slow, it was relatively easy to demonstrate significant improvements quickly. However, to remain relevant has required constant effort. In its initial version, FFS weighed in at a mere 1,200 lines of code. The current version, which remains a canonical file system after 30 years, has grown to 55,000 lines of code. This increase is the result of steady improvement in the attempt to remain competitive with new file system offerings and ideas. As he finished, McKusick quipped that in comparison, ZFS's 120,000 lines of code had been written in just a few years.

KEYNOTE ADDRESS

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

- ***Cloud Storage FUD (Failure, Uncertainty, and Durability)***
  Alyssa Henry, General Manager of Amazon Simple Storage Service (S3)

Alyssa Henry presented Simple Storage Service's (S3) goals and her team's experiences with developing an Internet-scalable and accessible storage system over the past four years. The presentation mixed colorful anecdotes with a description of the project's motivation and design.

Henry's major theme was that the S3's broad audience had many different requirements and workloads. To meet such a wide range of users, S3 needed to be designed to tolerate a large degree of uncertainty. It is possible to identify trends when a service is in active use, but these trends are dynamic in practice and cannot be relied upon. Underlying the design of the service is a low-cost "pay as you go" model, which is supported by leveraging Amazon's software expertise against commodity hardware and balancing the system's architecture against the need to keep costs low.

Failures are intrinsic to systems at this large scale, and Henry pointed out that even low probability errors begin to happen regularly. She classified service disruptions across two axes: duration, with errors ranging from temporary to permanent, and scope, ranging from few clients to many. Failures with sufficiently small scope and low duration are essentially harmless, while persistent errors with large scope are catastrophic.

The overarching strategy employed by S3 is to broaden the class of harmless failures into the region that would otherwise be considered catastrophic. Since perfection is not attainable, the goal is to balance the odds of service disruption against financial and complexity costs. The specific methods used toward this end are a mix of traditional approaches such as redundancy, rate limiting, and hardware diversity, with some newer ideas. Amazon's focus on the eventual consistency model was cited as one way in which they break from tradition. Amazon Web Services also routinely force failures to occur—for example, by yanking power plugs when a system is to receive downtime and by turning off whole datacenters for management. By stressing error paths, often one can work with more assurance that the countermeasures employed continue to be effective.

In closing, Henry commented that storage services represent lasting relationships that require trust. She also noted that reliability at low cost remains a difficult problem. She directed parties interested in more information to visit Werner Vogels' blog (http://www.allthingsdistributed.com/) and the Amazon Web Services blog (http://aws.typepad.com/aws/).

Sameer Ajmani from Google followed up on the hardware diversity comment to ask if software diversity was also a viable strategy. To this, the presenter noted that all aspects of the system represent design tradeoffs. For S3's needs, software diversity would be going too far, at too great expense. David Rosenthal from Stanford University asked why the system doesn't publish a numerical goal for reliability, as there is for availability. He pointed out that 100% reliability is not realistic and that the provided EULA has no penalty for data loss. Henry reiterated that the team's reliability goal was 100%, while in practice the service level agreement specified 99.9%, and this seemed to satisfy customers. S3 performs internal measurements of the error rate, but the resulting data is not disclosed. Stephen Spackman from Quantum asked how Amazon balances the trade-off between centralizing data in one country and offshoring it. S3 pushes this issue to the end user, who can choose between any combination of US and EU S3 offerings.

AUGMENTING FILE SYSTEM FUNCTIONALITY

Summarized by Phillipa Gill (phillipa@cs.toronto.edu)

- ***The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance***
  Ragib Hasan, University of Illinois at Urbana-Champaign; Radu Sion, Stony Brook University; Marianne Winslett, University of Illinois at Urbana-Champaign

Ragib Hasan presented a secure provenance scheme implemented at the application level. Hasan highlighted that most previously developed provenance schemes were applied in the domain of scientific applications. The provenance system developed in this paper is designed to be more secure so that it may be applied in finance or business applications. The authors designed a system that aims to prevent undetectable history rewriting. That is, any change or deletion of provenance information must be detectable.

Hasan described their method for providing secure provenance. The method uses the concept of a provenance chain which is made up of provenance entries (records of users' modifications and context). Adversaries in the system include users who may add or delete provenance entries or collude with each other to modify entries (more detail about adversaries in Hasan et al., *ACM Storage SS 2007*). Auditors are trusted entities who may verify the accuracy of the provenance chain by using checksums to prevent undetectable changes to the provenance entries. This is done by computing the checksum of an entry as a function of the previous entry's checksum and the new entry. Selective confidentiality is provided by encrypting the modification details and distributing the encryption key in such a way that only the auditors trusted by a user can see modification details. The scheme also allows selective disclosure to third parties by redaction of sensitive attributes without invalidating the integrity checksums. Hasan also talked about experimental results: for most typical real-life workloads, this secure provenance scheme incurs only 1%–13% runtime overhead.

The audience raised the issue of how the provenance system could handle a document that may be composed of multiple elements (e.g., an HTML page). Hasan stated that the cur-

rent scheme is applied to a single file but that a provenance chain may also be constructed for a whole document. The issue of security guarantees when multiple users collude and create multiple entries was also raised. Hasan stated that the colluding users can only modify their own entries and would not be able to tamper with provenance entries from benign users.

Hasan provided a URL for more information: http://tinyurl.com/secprov. [See p. 12 for an article about this paper.]

- ***Causality-Based Versioning***
  Kiran-Kumar Muniswamy-Reddy and David A. Holland, Harvard University

Kiran-Kumar Muniswamy-Reddy began by explaining a motivating example for causality-based versioning. He described the situation in which a piece of software is installed, but when it is uninstalled some changed files remain. A versioning system would enable a user to roll back the system but would not provide information about which files were modified. In another example, users continue their database work even after a malicious entity begins to tamper with the database. In these cases it is important not only to roll back the system, but also to know which pieces of data were modified (that is, which bits to keep).

Muniswamy-Reddy contrasted two different versioning schemes, open-close and version-on-write. Open-close is coarse-grained and versions when files are opened or closed, while version-on-write is fine-grained, creating a version for each write. Open-close has much lower overhead than version-on-write, but also provides less fine-grained versioning.

Muniswamy-Reddy then described the Cycle-Avoidance and Graph-Finesse algorithms proposed by his paper. Cycle-Avoidance preserves causality, but only uses local information when deciding to create a new version. As a result, it creates more versions than necessary. However, by only using local information, it has lower overhead. The Graph-Finesse algorithm uses global knowledge and as a result creates fewer versions. However, it has higher overhead than the Cycle-Avoidance algorithm. Implementation results were shown and the authors concluded that adding causality to versioning-based systems only increases overhead by 7%.

The audience asked how the system would compare to the open-close method on a single process. Muniswamy-Reddy emphasized that the causality becomes necessary only when a second process is present. The audience also asked how the algorithm would perform in a system with a large number of files. The author stated that for Cycle-Avoidance it would depend on the size of the local data, but that Cycle-Avoidance would perform much better than Graph-Finesse, which uses global data.

- ***Enabling Transactional File Access via Lightweight Kernel Extensions***
  Richard P. Spillane, Sachin Gaikwad, Manjunath Chinni, and Erez Zadok, Stony Brook University; Charles P. Wright, IBM T.J. Watson Research Center

Richard Spillane presented the audience with the results of his work extending the kernel to support transactional file access. Spillane described Valor, a file interface that requires only a small amount of modification to the page writeback mechanism and some additional module code. Valor adds seven new system calls to the kernel that allow processes to utilize atomic, consistent, isolated, and optionally durable transactions. In serial overwrite benchmarking tests, Spillane noted that Valor is 2.75 times slower than ext3, but it has lower overhead than Stasis, which runs 4.8 times slower than ext3. Spillane also showed that Valor outperforms Berkeley DB by a factor of 8.22.

An audience member asked if Valor performed dependency resolution between transactions. Spillane referred the questioner to the paper for details of Valor's isolation semantics. The audience also asked for more detail on the kernel and user-space implementations. Spillane explained that moving transactional support to the user level is difficult because performance will be impacted. Also, kernel support gives transactional support true transparency. The impacts of the transactional support on non-transactional I/O were also discussed. Spillane stated that non-transactional writes to a page that was being written to by a transaction would have to wait for both page writeback and any isolation locks on that page to be released. Margo Seltzer commented that the benchmarking workload used was not one that Berkeley DB (BDB) was made for and that the configuration would have also impacted the performance of BDB. She also noted that the page size chosen for BDB was sub-optimal. Spillane pointed out that Stasis provides an upper bound on the performance of BDB since it also utilizes a user-space page cache implementation, but Stasis is not restricted to writing into a B-Tree.

DIAGNOSIS

Summarized by Ragib Hasan (rhasan@uiuc.edu)

- ***Understanding Customer Problem Troubleshooting from Storage System Logs***
  Weihang Jiang and Chongfeng Hu, University of Illinois at Urbana-Champaign; Shankar Pasupathy and Arkady Kanevsky, NetApp, Inc.; Zhenmin Li, Pattern Insight, Inc.; Yuanyuan Zhou, University of Illinois at Urbana-Champaign

Weihang Jiang presented a tool for analyzing storage system logs to assist in customer troubleshooting. Today's complex storage systems need to deal with constant failures, which cause costly service downtime. Manual troubleshooting is also very costly for vendors. Problems may happen at different layers. Customer problem issues are reported to vendor support centers in two ways: human-generated reports (e.g.,

phone call, email) and automated built-in monitoring tools (e.g., storage system logs). Jiang said that manual processing of customer service requests often has a long turnaround time. He argued that by analyzing logs in a systematic way, it is possible to troubleshoot many problems automatically.

The authors analyzed a large problem database containing 600,000 problem cases and 300,000 logs. They found that hardware fault and misconfiguration are the main causes of problems. Software bugs caused only 3% of the errors, but they required a larger amount of troubleshooting time. Most of the customer problems are low-impact, and only 3% caused system crashes. Jiang described three techniques for analyzing logs: using critical events only; using single events; or combining multiple events. A score is computed based on how well the event signature can uniquely identify the cause. Jiang commented that they found logs to be noisy and verbose. Important log events are not easy to locate or link together. However, it is possible to identify and link patterns in the logs with specific types of problems. By applying clustering techniques, the tool described in the paper can help identify the causes of problems and help in troubleshooting.

An audience member asked what the authors do when an error starts as a hardware error but later causes software errors. Jiang replied that they only consider the initial cause when classifying errors. When asked why software errors are so expensive, Jiang explained that hardware errors are easy to solve simply by replacing the malfunctioning hardware component. But replacing software is not so easy. The audience raised the question of whether proactive action can be taken on the fly when a SCSI bus problem is detected. Jiang argued that not all SCSI errors lead to a problem, and finding the correlation between the SCSI error and a problem/crash is challenging. Finally, a question was raised about the nature of the study and whether the underlying system's properties changed during the study, e.g., after a software version update. Jiang replied that instead of a single type of system, they had studied a large number of logs from different storage systems, and most of the errors were caused by failing hardware rather than software. [See the related article on p. 31.]

- ***DIADS: Addressing the "My-Problem-or-Yours" Syndrome with Integrated SAN and Database Diagnosis***
  Shivnath Babu and Nedyalko Borisov, Duke University; Sandeep Uttamchandani, Ramani Routray, and Aameek Singh, IBM Almaden Research Center

Nedyalko Borisov presented DIADS, a tool that provides a holistic view of query execution and assists SAN and DBMS administrators during troubleshooting. The authors applied machine learning techniques and expert knowledge, and also implemented a data abstraction called Annotated Plan Graph (APG) that carefully integrates the DBMS and SAN monitoring data.

Borisov discussed various challenges in building the APG and how to solve them. He introduced a running example of SAN misconfiguration that causes performance degradation of a business intelligence query. In the DIADS workflow, the administrator first specifies the queries that had satisfactory and those that had unsatisfactory performance. The set of operators correlated with the query's performance are then identified, and an anomaly score is computed using the kernel density function. Later, these operators are used to look into related SAN components and calculate the anomaly score for them. Next, a symptom database is used to identify the root cause(s). Borisov discussed several challenges, such as expressiveness of the symptoms and missing symptoms. Once a root cause is identified, DIADS calculates its impact on the query performance. This reduces false positives and negatives and allows identification of high-impact root causes. Borisov concluded by presenting DIADS' evaluation scenarios, which consisted of incremental increase in complexity of the problems and investigation of the tool's ability to diagnose them.

An audience member asked whether DIADS can diagnose problems when the real SAN is replaced with a virtualized one. Borisov explained that monitoring data needs to be collected from the virtualization level of the SAN, and that DIADS will then be able to provide diagnosis. Another questioner asked whether the authors have considered creation of a performance metrics library to provide more useful monitoring data. Borisov argued that it is not known in advance when a problem will happen; thus, having intrusive data collection enabled all the time would cause a significant performance hit on the production system.

WORK-IN-PROGRESS REPORTS (WIPS) PART ONE

Summarized by Michelle Mazurek (mmazurek@andrew.cmu.edu)

- ***Progress on FileBench***
  Andrew Wilson, Sun Microsystems

Wilson discussed recent additions to FileBench, a model-based approach to improving file system benchmarking. Existing macro benchmarks are too time-consuming, while existing micro benchmarks are not comprehensive enough. In addition, the wide variety and lack of standardization in benchmarking can be frustrating. FileBench solves this problem by allowing the tester to model workloads in a high-level language, quickly run the test, and collect results. Recently added features include random variables, multi-client support, extension to support file sets as well as individual files, and composite operations that resemble inline subroutines. An NFSv3 plugin is nearing completion.

- ***When "More and More" Does Not Help: Sensible Partitioning of Cache***
  Hamza Bin Sohail, Purdue University

Bin Sohail presented a new approach to partitioning the buffer cache among applications. Current algorithms that do not partition the cache can result in cache hogging by certain processes, to their detriment. Bin Sohail proposes partitioning the cache and allocating larger portions to "good" processes whose instantaneous hit ratio increases as cache allocation increases. Simulation results indicate that statically allocating more cache to a historically "good" process and less cache to a historically "bad" process results in increased system performance. The next step is to partition the cache dynamically by monitoring processes as they run; it remains to be seen how effective this method will be and how much overhead it will require.

- ***Solving TCP Incast in Cluster Storage Systems***
  Vijay Vasudevan, Hiral Shah, Amar Phanishayee, Elie Krevat, David Andersen, Greg Ganger, and Garth Gibson, Carnegie Mellon University

Vasudevan discussed the TCP incast problem, which occurs when synchronized reads in a cluster environment cause TCP timeouts, resulting in a throughput collapse. The default 200 ms delay between TCP retransmissions is too long and wastes resources; can reducing or eliminating this lower bound provide a safe, effective, and practical solution? The standard TCP implementation relies on timers with millisecond granularity; a 5 ms timeout improves performance on systems with small stripe widths but fails for larger stripe widths. Using the Linux kernel's high-resolution timer with microsecond granularity, the incast problem can be avoided for at least 47 concurrent senders. As datacenters move toward increased bandwidth and more servers, response time at the latency of the network will be increasingly important. For more information, see tinyurl.com/incast.

- ***Improving I/O Performance by Co-scheduling of I/O and Computation on Commodity-Based Clusters***
  Saba Sehrish, Grant Mackey, and Jun Wang, University of Central Florida

Sehrish presented a framework for more efficient scheduling of Map-Reduce tasks in a fault-tolerant system like Hadoop. The scheduling algorithm improves efficiency by intelligently assigning multiple DFS (Distributed File System) blocks per map task, based on data locality. The algorithm considers three cases: dependent DFS blocks combined statically as an application requirement, independent DFS blocks combined statically to improve performance, and independent DFS blocks combined dynamically to improve performance. In the first case, the node with the most participating DFS blocks is chosen as the host node for the task, and each remaining block is retrieved from the node with minimal latency. The second case resembles the first, but the latency of transferring data to the host node is compared with the overhead of creating a separate task for the remote node to determine the optimal configuration. In the third case, one task is created for each set of co-located participating blocks; the number of tasks and the number of blocks per task are determined dynamically.

- ***Predictable and Guaranteeable Performance with Throughput, Latency, and Firmness Controls in Buffer-Cache***
  Roberto Pineiro and Scott Brandt, University of California, Santa Cruz

Tolerance of I/O performance degradation ranges from services that require hard realtime guarantees to those that require soft guarantees and even those that tolerate best-effort. Pineiro proposed a system that supports a mix of such services by providing different levels of predictable and guaranteeable performance in the buffer cache. To enforce hard guarantees, Pineiro focused on coordination of components in addition to conservative assumptions. The system uses device time utilization rather than softer metrics like bandwidth to manage devices, and it enforces hard isolation of components. I/O rate and deadline requirements are enforced both into and out of the buffer cache, which is partitioned according to I/O properties and performance requirements. Test results comparing this system to Linux using CFQ (Completely Fair Queuing) yield more stream isolation, more stable performance relative to the guarantee type, and a slight improvement in overall throughput.

- ***NFSv4 Proxy in User Space on a Massive Cluster Architecture: Issues and Perspectives***
  Philippe Deniel, Commissariat à l'Énergie Atomique, France

Deniel noted that server architectures are evolving from individual clusters to aggregations of clusters. This presents a problem for using NFS, as exponential growth in clients will overwhelm the servers. Deniel proposes using proxy servers based on the existing NFS-GANESHA tool and running in user space to solve this problem. Serving an aggregation of clusters with one main proxy server would be perfect for read-only workloads; in the real world, write, create, and delete operations lead to cache incoherency. To solve this, Deniel is developing a protocol for communication among proxy servers. The first implementation is expected by the end of the year.

- ***Comparing the Performance of Different Parallel File System Placement Strategies***
  Esteban Molina-Estolano, Carlos Maltzahn, and Scott Brandt, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory

Molina-Estolano presented a trace-driven simulation approach to comparing file-placement strategies used by different parallel file systems. The goal is to compare only the file placement strategies rather than the file systems themselves. Simulated clients using various placement strategies are driven by traces from different workloads, including

scientific computing and Web server workloads. The effects of normalizing chunk size and turning off redundancy were also considered. Preliminary results, which measured balance across the cluster, found the PanFS and Ceph strategies to be comparably balanced. Turning off redundancy in Ceph has limited effect, but turning off redundancy in PanFS increases balance. In addition, reducing chunk size in Ceph increases balance. Future simulations will measure performance in addition to balance. Molina-Estolano also discussed the need for more workload traces, particularly those related to data-mining and enterprise workloads.

- ***Overlapped HPC Checkpointing with Hardware Assist***
  Christopher Mitchell and Jun Wang, University of Central Florida; James Nunez and Andrew Nelson, Los Alamos National Laboratory

Mitchell noted that as high-performance computing systems get larger, they spend more and more time on failure mitigation processes such as checkpointing and error recovery, reducing system utilization. To solve this problem, either checkpoints must write less data or they must happen faster. Mitchell proposed adding a fast, non-volatile checkpoint buffer between the application and the file system. The prototype system will have three main components: a fleet of servers with connected buffers, a daemon to migrate checkpoint data from the buffer to the file system, and an API allowing application developers to access this checkpoint method. Currently, the servers and daemon are operational and the API is nearing completion. Preliminary testing shows significant improvements over existing methods.

- ***Moderated Collaboration to Modify Shared Files Among Wireless Users***
  Surendar Chandra and Nathan Regola, University of Notre Dame

Chandra presented a system for wireless file collaboration among multiple authors. Analysis indicates that contemporary users mainly use wireless devices such as laptops and are typically available for only short sessions with relatively long duration between them. Traditional approaches such as mandatory locking and epidemic propagation fail when multiple users are online at the same time. Chandra proposed creating one writable version of each collaboration file per user; users can also hoard read-only copies of other users' versions, distributed via epidemic propagation. Each user manually reconciles his changes with those of the other authors until convergence is achieved. Logs that track causal provenance allow users to determine whether their changes have been incorporated into the latest version. A prototype of this system, developed using FUSE, achieves acceptable performance in terms of memory used as well as file transfer time.

- ***Probabilistic Reputation for Personal Trust Networks***
  Avani Wildani and Ethan Miller, University of California, Santa Cruz

Wildani discussed the issue of trust verification in peer-to-peer storage. Out-of-band trust verification such as OpenPGP's web of trust is difficult and expensive. Wildani proposed an alternative approach where trust is calculated dynamically by individual nodes and used to make locally optimal decisions. Individual nodes sort peers into trust clusters, where the innermost cluster is most trusted. When a node successfully reads a file from a peer, the reputation of that peer is updated. When writing, a node sends out a number of replicas proportional to the perceived trustworthiness of the recipients. This system limits the severity of an attack: if a node is compromised, the attacker only gains information about that node's trust relationships. Because there is no central repository of reputation, there is no single attack point for poisoning trust relationships. Development of a simulated system using PlanetLab is in progress.

WORK-IN-PROGRESS REPORTS (WIPS) PART TWO

Summarized by Rik Farrow

- ***Can Clustered File Systems Support Data Intensive Applications?***
  Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, and Renu Tewari, IBM Research

Mansi Shah said that extreme data applications, such as Web page indexing and genome searches, demand a storage layer that is scalable and cost-effective, as well as fault-tolerant. Special file systems like Hadoop Distributed File System (HDFS) and the Google File System can also ship computation to the nodes that contain the data to be searched. Shah argued that cluster file systems, such as Lustre and IBM's GPFS, can do that as well. They experimented with GPFS, first increasing the block size, which worked poorly. But by changing the block allocation scheme to mimic a large block size, and through exposing block location via an ioctl() call, they were able to match the performance of HDFS while still maintaining performance for legacy tasks in GPFS.

- ***Data Destruction: How Can You Destroy Data and Prove It Is Destroyed?***
  Dan Pollack, AOL LLC

AOL leases systems and thus has a strong interest in data destruction techniques that do not involve destroying hardware. When they return equipment, they must be able to prove to auditors that they have destroyed any data in storage. At the same time, storage systems are getting larger: overwriting a disk at 1GB/sec translates into 3.6TB/hour, which is slow if you have petabytes to destroy. And the increased workload on drives during the overwriting pro-

cess can result in failed drives before the overwrite can be completed. They saw a five-fold increase in device failures during the most recent attempt at destroying data. Pollack asked the community for help in coming up with effective ways of destroying data.

- ***SmartStore: A New Metadata Organization Paradigm with Semantic-Awareness***
  Yu Hua, Huazhong University of Science and Technology; Hong Jiang, University of Nebraska—Lincoln; Yifeng Zhu, University of Maine; Dan Feng, Huazhong University of Science and Technology; Lei Tian, Huazhong University of Science and Technology and University of Nebraska—Lincoln

Lei Tian posed the problem of finding files in very large systems where there may be millions of files and nearly an exabyte of storage. Tian said their system, SmartStore, is different from Spyglass (FAST '09) in that it groups and stores files according to their metadata semantic correlations. They use Latent Semantic Indexing to measure semantic correlations and construct multiple logical R-trees that improve search. Tian used graphs to demonstrate the dramatically improved performance of range searches (e.g., all files that took less than 30 minutes to generate and are less than 2.6GB) and top-k queries (e.g., find the top ten matching files) over a conventional file system and a DBMS with stored metadata. Their prototype emulates I/O behaviors of a large storage system by scaling up I/O traces both spatially and temporally for testing purposes.

- ***Making the Most of Your SSD: A Case for Differentiated Storage Services***
  Michael Mesnier and Scott Hahn, Intel Corporation; Brian McKean, LSI Corporation

Michael Mesnier introduced the notion of Differentiated Storage Services (DSS) as a method for making the most out of SSDs when mated with disks. They modified ext3 by adding policies that assign quality of service (QoS) levels for different classes of writes, with metadata, journal, directory, and small files being given priority. The main idea is to separate policy from hardware implementation so that file systems can assign a QoS to a write request that the hardware can optionally respond to. Mesnier argued that simply using an SSD coupled with a hard drive as a cache wastes potential performance gains. He also said that this work applies not only to SSD but to other storage hierarchies.

- ***On the Consistability of Storage Systems***
  Amitanand Aiyer, Eric Anderson, Xiaozhou Li, Mehul Shah, and Jay J. Wylie, HP Laboratories

Amitanand Aiyer defined consistability as an attempt to describe the different levels of consistency found in a storage system at any point in time. In a perfectly performing storage system, the system may provide atomic consistency. But in a system that experiences some fault 20% of the time, the system provides atomic consistency 80% of the time and regular consistency 100% of the time. An example of a fault would be network partitioning, where the ability to get that last value put into the system degrades into the ability to get one of K most recent values.

- ***Speedy and Scalable File-System Benchmarking with Compressions***
  Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Nitin Agrawal explained how their tool, Compressions, can be used to simulate very large storage systems. As the amount of disk storage has grown, it has become more difficult to create realistic simulations for benchmarking, and synthetic benchmarks are also hard to create. Compressions allows an evaluator to run a large benchmark using a much smaller disk—for example, 100GB to simulate 1 terabyte. They do this by doing away with all data blocks and laying out metadata blocks (inodes, directories, indirect blocks, etc.) more efficiently on disk. All data writes are discarded, and reads are supplied with mock data that may resemble what is expected. Delays are added to disk operations to simulate full-scale operation while using just 10% as much disk space.

- ***Out-of-Place Journaling***
  Ping Ge, Saba Sehrish, and Jun Wang, University of Central Florida

Ping Ge presented work on improving the performance of journaling file systems. Journaling file systems maintain system integrity through atomic writes by writing to the log (journal) and then by committing the log entries. The second step proceeds by copy-on-write (COW) or by updating pointers to the data written by the log. Ping Ge presented a third mechanism, which they have implemented and tested in ext3: they use a mapping layer between the file system and the device driver to map logical blocks. After data gets written to the log, the mapping layer commits this data by redirecting requests for the data to the blocks in the log. If the system crashes, description records in the log can be used to rebuild the mapping layer.

POSTER SESSION

Summarized by Madalin Mihailescu (madalin@cs.toronto.edu)

- ***On the Consistability of Storage Systems***
  Amitanand Aiyer, Eric Anderson, Xiaozhou Li, Mehul Shah, and Jay J. Wylie, HP Laboratories

Amitanand Aiyer proposed quantifying the consistability modes of a storage system under various operating conditions. The intuition is that systems that offer different consistency levels can be compared by estimating the percentage breakdown of the levels each system achieves in the presence of various failure scenarios. Current work is being conducted to better understand design/implementation trade-offs for a key-value store.

- ***Can Clustered File Systems Support Data Intensive Applications?***
  Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha, Prasenjit Sarkar, Mansi Shah, and Renu Tewari, IBM Research

Mansi Shah argues that cluster file systems such as Lustre and PVFS can be tweaked to support data-intensive applications, thus eliminating the need for specialized file systems, e.g., GFS and Hadoop DFS. One advantage with this approach comes from deploying a single file system that supports both data-intensive and legacy applications. The authors changed IBM's GPFS to accommodate the Map-Reduce framework. Preliminary evaluation shows comparable performance with Hadoop DFS.

- ***Adaptive Context Switch for Very Fast Block Device***
  Jongmin Gim, Kwangho Lee, and Youjip Won, Hanyang University, Korea

Jongmin Gim notices that context-switching processes when performing I/O could be inefficient when using current non-volatile memory drives, such as SSDs. This is based on the observation that context-switch overhead can be as high as 300µs, while I/O access time on SSDs is an order of magnitude lower. To address this problem, the authors built an adaptive context-switch algorithm. The algorithm tries, by analyzing I/O response times, to determine whether context switch is beneficial. Preliminary results show performance improvements of up to 16%.

- ***SmartStore: A New Metadata Organization Paradigm with Semantic-Awareness***
  Yu Hua, Huazhong University of Science and Technology; Hong Jiang, University of Nebraska—Lincoln; Yifeng Zhu, University of Maine; Dan Feng, Huazhong University of Science and Technology; Lei Tian, Huazhong University of Science and Technology and University of Nebraska—Lincoln

SmartStore targets the problem of efficient metadata retrieval in large-scale storage systems. In particular, it focuses on range queries and top-k queries. It uses Latent Semantic Indexing to group semantically correlated files, based on their metadata. Furthermore, an R-tree is used to store the metadata, based on the grouping obtained from LSI. Compared against an R-tree scheme without semantic knowledge and a DBMS, SmartStore has very low query latency numbers.

- ***Comparing the Performance of Different Parallel File System Placement Strategies***
  Esteban Molina-Estolano, Carlos Maltzahn, and Scott Brandt, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory

Esteban Molina-Estolano proposed a comparison among the various placement strategies implemented in current parallel file systems. He implemented a basic simulator for the placement strategies used in Ceph, PanFS, and PVFS. Preliminary evaluation using real and synthetic I/O traces showed the three file systems having comparable placement techniques in terms of balance. The chunk size and redundancy strategy used by each file system have an impact on the balance. Future work includes improving the simulator to allow for performance comparison.

- ***Supporting Data-Intensive Applications on Accelerator-Based Distributed Systems***
  M. Mustafa Rafique, Ali R. Butt, and Dimitrios S. Nikolopoulos, Virginia Tech

Mustafa Rafique argues that current large-scale clusters that leverage computational accelerators, such as GPUs, for high-performance computing implement either ad hoc or specific solutions. Thus, there is a need for understanding alternative designs in this space, depending on the capabilities of various accelerators, in the context of data-intensive applications. Accelerators are classified based on their compute power and, mainly, the extent to which they can manage external resources, e.g., I/O devices. This led to four configurations, which were evaluated against a number of Map-Reduce applications. The experimental setup was built using Sony PS3s and a multicore cluster. Future work will try to make the framework more generic.

- ***Exploiting the Overlap Between Temporal Redundancy and Spatial Redundancy in Storage System***
  Pengju Shang, Saba Sehrish, and Jun Wang, University of Central Florida

Pengju Shang proposed bridging the gap between application-level temporal redundancy techniques (e.g., a database log record) and storage-level spatial ones (e.g., RAID). The authors built a transactional RAID, or TRAID, for database systems, which aims to take advantage of the redundancy overlap. TRAID lowers the log wait time and log size while ensuring the database ACID semantics. Results show that TRAID can improve RAID by 40–50%, depending on the RAID version. Current work is being done to adapt this technique to versioning file systems, e.g., ext3cow.

- ***Using Realistic Simulation to Identify I/O Bottlenecks in MapReduce Setups***
  Guanying Wang and Ali R. Butt, Virginia Tech; Prashant Pandey and Karan Gupta, IBM Almaden Research

Dumbo is a simulator for the MapReduce framework. It can be used to analyze application performance by understanding the impact of various configuration parameters for typical MapReduce deployments, e.g., storage, compute capacity, network topology, or data layout. The analysis is done with minimal resources. A prototype implementation managed to uncover a network-related performance inefficiency in Hadoop, an open source version of MapReduce.

[Editor's Note: Many more posters were not summarized or have been summarized as WiPs. See http://www.usenix.org/events/fast09/poster.html for the full list of posters and their abstracts.]

SCHEDULING

Summarized by Brandon Salmon (bsalmon@ece.cmu.edu)

- ***Dynamic Resource Allocation for Database Servers Running on Virtual Storage***
  Gokul Soundararajan, Daniel Lupei, Saeed Ghanbari, Adrian Daniel Popescu, Jin Chen, and Cristiana Amza, University of Toronto

The paper includes two parts: building a latency model, and building a resource partitioner which operates on this latency model. They consider three resources in their models and controllers: file system cache, database buffer pool, and disk bandwidth.

To build the cache models, they observe that if the caches are LRU, then the larger cache dominates the smaller, so they simulate both the file system and database caches as a single cache of the size of the larger cache. To model the disk, they model the latency based on the latency the application would have with the disk to itself, and they assume a large sharing quanta. To handle cases where the models are inaccurate, they use cross-validation, and in regions where the models are inaccurate, they sample and interpolate.

Given these models, they can provide multi-level resource allocations, which give up to 3x performance improvement over a conventional single-level resource allocator.

- ***PARDA: Proportional Allocation of Resources for Distributed Storage Access***
  Ajay Gulati, Irfan Ahmad, and Carl A. Waldspurger, VMware Inc.

This paper focused on providing proportional resource allocation, based on tickets given to each VM to specify its relative importance, to virtual machines running on several different physical hosts without requiring coordination between the physical machines.

Host-level schedulers are not sufficient, since each host may have multiple VMs. To address the problem they keep a queue for each host based on the proportion of the tickets given to that host. To avoid problems with choosing metrics, each VM is given some number of slots which it is allowed to keep full at any given time. However, PARDA needs to be sure that the queue is appropriately deep to avoid increasing latency above a threshold.

To do so, each VM tracks the latency to a common shared file and then adjusts the queue length appropriately. Evaluations show that this allows PARDA to provide proportional sharing with minimal impact on performance.

- ***CA-NFS: A Congestion-Aware Network File System***
  Alexandros Batsakis, NetApp and Johns Hopkins University; Randal Burns, Johns Hopkins University; Arkady Kanevsky, James Lentini, and Thomas Talpey, NetApp

  Awarded Best Paper!

Conventional NFS systems are prone to problems with congestion on the network when too much traffic is going to the servers. However, in conventional systems the clients do not know how to trade off the various resources consumed by operations. CA-NFS approaches this problem by assigning a price to each resource in the system based on the current scarcity of that resource, and then combining to provide a utilization metric. CA-NFS considers the resources: server CPU, client and server network, server disk, client and server memory, and client read-ahead effectiveness.

This allows clients to make decisions about whether to send asynchronous writes or reads back to the server, for example, or whether to hold them in local memory instead. It does so by comparing the price it would be willing to pay to free its resources with those consumed by the server. Evaluations show that these methods provide a 20% performance improvement over standard NFS for several workloads.

TOOLS YOU WISH YOU HAD

Summarized by James Hendricks (James.Hendricks@cs.cmu.edu)

- ***Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality***
  Mark Lillibridge and Kave Eshghi, HP Labs; Deepavali Bhagwat, University of California, Santa Cruz; Vinay Deolalikar, HP Labs; Greg Trezise and Peter Camble, HP Storage Works Division

Mark Lillibridge presented his results on data deduplication for disk-to-disk backup. As disks get bigger and cheaper, backing up to disk rather than tape makes sense and provides many benefits. Unlike tape, the random access available with disks allows for deduplication of data. Data deduplication replaces duplicate data with pointer(s) to the original data. One approach to data deduplication is chunk-based deduplication, in which data is broken into chunks and the chunks are hashed. Under the standard of implementation, the hash values are then looked up in a table kept in RAM. If the hash value for a particular chunk is already present in the table, that chunk has already been stored, so only a pointer to that chunk is stored. If no hash value is found, that entire chunk is stored and its hash value is added to the table. The problem with this approach is that 100 terabytes of physical disk requires over 1 terabyte of hash values, which exhausts available RAM. One option is to store hashes on disk, but then each chunk lookup requires a slow disk lookup.

Storing hashes on disk but caching recently used hash values does not work, because backup streams exhibit little temporal locality. For example, a file will be read today,

then terabytes of other data will be read, and then the file will be read tomorrow. Instead of temporal locality, backup workloads exhibit chunk locality, which means that if a chunk reoccurs, it tends to occur near other chunks that were nearby when it was last seen. Rather than tracking all hashes in RAM, this paper uses a technique called sparse indexing. Consecutive chunks of data are grouped into segments, and only a few hashes are sampled per segment. The sampled hashes form a sparse index that fits in RAM. To back up a segment, its samples are looked up in the sparse index to find previously backed up segments that contain a lot of chunks in common with the new segment. The new segment is then deduplicated against a few of the found segments by loading in from disk lists of the chunks contained in those segments. By chunk locality, over 99% of the duplicate data can be removed this way even though only a small number of segments rather than the entire store are deduplicated against. Thus, sparse indexing allows deduplication of large-scale backup data.

Hakim Weatherspoon of Cornell University asked how much of the gain was due to a common chunk such as the chunk of all zeroes. Mark said that removing the top 100 most common chunks often reduced the data size by 1%, but sometimes by up to 10%. Hugo Patterson, CTO of Data Domain, asked about read performance compared to the approach in last year's paper from Data Domain. Mark said that chunks are stored in the same way as last year's paper proposed, so read performance should be similar. Bill Bolosky of Microsoft Research asked why chunk size makes much difference. Mark said that files often aren't quite the same. Bill also questioned whether compressing data before storing would do as well as suggested in the paper, because a lot of data is already compressed (music, archives, etc.). Mark replied that it varies widely.

- ***Generating Realistic* Impressions *for File System Benchmarking***
  Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

  Awarded Best Paper!

"For better or for worse, benchmarks shape a field" (Dave Patterson). Nitin Agrawal presented a tool, Impressions, that creates representative file system images for benchmarking. The community already knows properties of metadata and disk fragmentation, and Nitin argued that there is a need for an easy-to-use tool to create representative, controllable, and reproducible file system images. Impressions provides such functionality by taking file system distributions as input along with user-specified parameters such as total file system size. For example, Impressions can use the metadata distribution from the presenter's FAST '07 paper. Impressions has an advanced mode where several knobs can be turned and a basic mode that provides reasonable defaults.

Impressions uses a generative probabilistic model to create files and directories. Each directory is created, then a parent is chosen according to a probability model. Each file is created and its size, extension, and parent are created by a similar model. Traditionally, file sizes were assumed to be distributed lognormal. More recent studies have shown a bimodal distribution. Impressions' model of file system size is hybrid, using a lognormal body but a Pareto tail. Files are generated according to size model and then are attached to directories.

The tool will be available soon at http://www.cs.wisc.edu/adsl/Software/Impressions.

Ajay Gulati of VMWare asked if the tool could be augmented to run in reverse, such that one could run it on a machine to extract statistical properties and then generate a reasonable file system. Nitin said he had something similar for in-house testing but a full version for release is not available. Drew Wilson of Sun said FileBench already does this. Nitin replied that his contribution was to allow one to contribute newer designs and data sets and to make it easier to plug in distributions. Mike McThrow of Cal Poly San Luis Obispo asked if image creation is reproducible. Nitin said the random seeds can be set for reproducibility. Geoff Kuenning of Harvey Mudd University asked how long it takes to build a big image. Nitin said the tool currently writes images at10MB/sec, but the tool could be optimized to go much faster. The last questioner asked if filename length was modeled. Nitin said filename lengths in the data sets were anonymized.

- ***Capture, Conversion, and Analysis of an Intense NFS Workload***
  Eric Anderson, HP Labs

This paper describes new industrial-strength NFS tracing techniques needed to capture workloads at scale. The goal was to collect a customer's NFS traces, but standard techniques failed due to the huge volume of data. Many improvements were incremental, but all were needed to achieve the goal. Eric wanted to highlight two big takeaway points. First, if you take traces, read the paper and apply the techniques. For example, future traces should not drop packets—lindump, driverdump, or endacedump can capture traces without dropping packets. The improved data analysis techniques allow for handling these gigantic traces on modest systems. Second, if you need workloads, look at the ones he describes here. The workload is much different and significantly larger than prior workloads. All of the data and tools are open-sourced under a BSD license, so anyone can reproduce the results of the paper or conduct further analysis. Eric argued that there is an acute need for new traces. There are many different workloads but few traces over the past decade, so trace-based studies do not reflect most modern workloads. Eric encouraged the community to publish more traces and more trace analysis.

The customer was a feature animation (movie) company. The applications read models, textures, and animation curves and wrote intermediates and pictures. There were

thousands of clients, tens of NFS servers, twenties of NFS caches, many rack switches, and a few core routers. The workload was different from many that have been previously studied. For example, most files have a single read, so any prefetch mechanism must work across files. The workload was very intense, which reduces the need to arbitrarily speed up a replay of the trace to evaluate a system. The workload also had many small files, which means that replaying the trace would really stress a system's performance.

The tools are available from http://tesla.hpl.hp.com/opensource/, and the traces are available from http://apotheca.hpl.hp.com/pub/datasets/animation-bear/.

Brent Welch of Panasas asked if Eric could figure out the working set for the textures and models. Eric replied that he didn't do much analysis on the trace, but he hopes other researchers will explore the trace in more depth. Brent Callaghan of Apple asked if packet reassembly was done to process readdir. Eric replied that a flaw discussed in the paper prevented this.

METADATA AND OPTIMIZATION

Summarized by Avani Wildani (agadani@gmail.com)

- ***Spyglass: Fast, Scalable Metadata Search for Large-Scale Storage Systems***
  Andrew W. Leung, University of California, Santa Cruz; Minglong Shao, Timothy Bisson, and Shankar Pasupathy, NetApp; Ethan L. Miller, University of California, Santa Cruz

Searching through petascale storage systems has become more and more difficult to manage. Current techniques include crawling metadata or building a DBMS that mirrors the system's metadata. Leung described the approach used in Spyglass as creating a versioned metadata index that gets stored as part of the storage server.

The authors surveyed users to get a list of requirements and analyzed storage systems used at NetApp and UCSC. They found that most searches involved multiple search parameters and had strong locality. For example, a user searching for lost files only needs to search within her own partition of the storage system. Spyglass takes this further, with hierarchical partitioning and signature files that use a bit map to provide hints for the classes of metadata found within each partition. Spyglass also takes advantage of a feature of NetApp's WAFL file system, snapshots, so that updating the index only involves looking at new versions of files. They compared the performance of Spyglass to systems based on PostgreSQL and MySQL and found that Spyglass could answer most queries in less than a second, something that the DBMS versions could only occasionally accomplish.

An audience member asked how they deal with directory renaming. Andrew answered that since the index itself is versioned, the next version reflects the rename. Versions are merged over time to recover space, and the partitioning strategy isn't strict, so even if the directory is moved, the worst that happens is that two partitions get searched.

- ***Perspective: Semantic Data Management for the Home***
  Brandon Salmon, Carnegie Mellon University; Steven W. Schlosser, Intel Research Pittsburgh; Lorrie Faith Cranor and Gregory R. Ganger, Carnegie Mellon University

Brandon Salmon made a polished presentation clearly demonstrating the need for semantic file naming and transparent file migration for nontechnical users. He began by describing case studies of users to learn how file names are used. As an example, Brandon described how a student might want to find a song by a particular artist and copy it to another device she owns. The problem is that iTunes, for example, organizes files semantically, by artist, album, and song, whereas the Mac Finder uses hierarchical names that do not map clearly to the iTunes view.

Perspective addresses the core issues in several ways. It captures the decentralized nature of devices by being P2P. It allows semantic management of data, allows for rule-based data management, and provides a way for automation tools to do things for the user while giving the user readable feedback. Perspective provides a global namespace across devices. Files are accessible through FUSE, and any replica of a file can be modified at any time. Devices aren't forced into a topology, and conflicts are handled similarly to previous systems. Views are used in file management: if a user wants all of her files on a given device—on a cell phone or a desktop, for example—she can specify that in a view. An automated system may ask to move files across from the phone to the desktop if the phone fills, and it will modify the views as it does. The views are human-readable, to make it easy to customize any automated decisions.

The first question was what happens if devices run out of space when copying a file for redundancy. Brandon answered that Perspective will never drop a file. The next questioner asked about results showing 60% accuracy in user testing of Perspective and wondered what was difficult. Brandon said that the interface they designed is overwhelming at first, and that the notion of hierarchy, as displayed with click and expand, is difficult for some people. After the Q&A completed, Brandon continued to be plied with questions.

- ***BORG: Block-reORGanization for Self-optimizing Storage Systems***
  Medha Bhadkamkar, Jorge Guerra, and Luis Useche, Florida International University; Sam Burnett, Carnegie Mellon University; Jason Liptak, Syracuse University; Raju Rangaswami and Vagelis Hristidis, Florida International University

Medha Bhadkamkar explained that BORG uses a special partition, called BOPT, as a write cache and for storing frequently accessed blocks. They created heat maps for different workloads (office, developer, subversion server, and Web server) and discovered that unique reads are a small portion of disk data but are spread out over the entire volume. Also,

non-sequential block accesses repeat on certain workloads, and there is substantial overlap in the working sets across days. Thus, past I/O information can be used to reorganize data and improve performance. BORG identifies block access patterns in the workloads and copies them sequentially to the BOPT partition. The size of the partition is controlled by an administrator and includes room for a write buffer, reducing seek latency.

BORG operates in the background, is independent of the file system, and can be dynamically inserted and removed. It maintains consistency with a page-level consistency map. The architecture consists of user space components, the analyzer and planner, and kernel space components, the profiler, indirector, and the BOPT-space reconfigurator. The analyzer operates when needed and creates a weighted, directed graph representing the frequency of the accesses between nodes. These graphs become a master graph for the planner component to create a new layout. The indirector directs all writes to the BOPT partition as well as reading blocks stored in this partition. This process is iterative and continuous. If the BORG module is removed, the dirty blocks are copied back into the file system. Disk-busy times were reduced by up to 80% with optimal parameters.

Someone wondered that as the workload on the Web server was 1% busy time, what the effect was on response time. There was an improvement of up to 46%. Another person wondered what happens if the BOPT partition is full and a write occurs. The write buffer was full during the sensitivity analysis but not for the rest of the experiments. The final questioner asked Medha to compare this work to the performance of a logging file system (LFS). Medha responded that this work is based on LFS in that it tries to make writes sequential, but it also has good read performance, unlike LFS.

DISTRIBUTED STORAGE

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

- ***HYDRAstor: A Scalable Secondary Storage***
  Cezary Dubnicki, Leszek Gryz, Lukasz Heldt, Michal Kaczmarczyk, Wojciech Kilian, Przemyslaw Strzelczak, and Jerzy Szczepkowski, 9LivesData, LLC; Cristian Ungureanu, NEC Laboratories America; Michal Welnicki, 9LivesData, LLC

Michal Welnicki described HYDRAstor as a scalable deduplication system for the enterprise market. Deduplication systems are faced with huge volumes of data, requiring scalability at a low cost per terabyte. They also must perform deduplication globally, be able to differentiate between high and low value data, and provide failure-tolerant restore performance. HYDRAstor is a system initially from NEC research but now successfully commercialized as a grid storage platform at a variety of capacities. Its architecture consists of front-end nodes that export an NFS or CIFS interface and can be scaled for performance, and back-end nodes that can be scaled for capacity.

The basic unit of storage in HYDRAstor is a synchron, a collection of subsequently written blocks stored linearly on disk. Metadata for the system is stored separately for performance reasons. Failure tolerance is provided through erasure coding with a standard N of M loss model. Data in HYDRAstor can be located at any of the back-end nodes, and request routing is performed with a DHT. Scalability, load balancing, and data relocation provide dynamic performance stability, which was said to be challenging to implement in concert with deduplication. In his evaluation, Welnicki showed an instance of HYDRAstor operating at a full throughput of 600MB/sec. Then a node failure is introduced and the system begins reconstruction. This has the effect of dropping the user's throughput to 400MB/sec.

Niraj Tolia of HP Labs asked Welnicki if hashes were evaluated when data is written out to avoid hash collision. Welnicki clarified that they weren't, but they agreed that they themselves didn't believe that such precautions were necessary, given the unlikelihood of such an event.

- ***Smoke and Mirrors: Reflecting Files at a Geographically Remote Location Without Loss of Performance***
  Hakim Weatherspoon, Lakshmi Ganesh, and Tudor Marian, Cornell University; Mahesh Balakrishnan, Microsoft Research, Silicon Valley; Ken Birman, Cornell University

Hakim Weatherspoon presented Smoke and Mirrors, which attempts to provide a higher-performance solution to safely mirroring data on geographically remote servers, specifically targeting industrial applications. Data stored in a single cluster is vulnerable to loss if the cluster experiences a catastrophic error. Since such a data-loss event could be disastrous, geographically remote shadowing of data is a very attractive option. However, designers of such a system are faced with a key decision with respect to performance and consistency: when can data be reported as safely reaching persistent storage? The performant option is to send a confirmation when the data reaches nonvolatile RAM on the local cluster; however, this is not the most reliable option. A safer but slower approach is to wait for data to be confirmed at the remote location, a process that is fundamentally limited by the speed of light.

Smoke considers a middle-ground position where confirmations can be sent once the border router transmits data towards the remote site. Since datacenters operate over low error-rate fiber, error rates nearing those of disks can be achieved by sending multiple copies of each packet to the network. Forward error correction is used to send the redundant packets and works as an efficient way to mitigate network error conditions. The result is a system that is safer than the local synchronization case and faster than the remote synchronization case. Weatherspoon was very clear in stating that the position was a compromise in which data could still be lost.

The system was evaluated with Emulab over 1GB links. Congestion and latency were introduced to simulate a geo-

graphically remote fiber connection. Weatherspoon showed how in this scenario Smoke was able to operate without data loss, at the cost of 3 extra network packets per 8 packets of goodput, and with very low overhead. Weatherspoon pointed to moving the experiments onto large private ring networks as the next step towards wider adoption.

The audience's response included concern from two attendees about the assumption that each packet would experience error in a probabilistically independent manner. Weatherspoon pointed to the forward error correction as handling some of the errors associated with, for example, transient network congestion that could result in a batch of dropped packets. David Rosenthal of Stanford admonished everyone present to monitor the loss rate on their networks (as was done in Smoke), and Stephen Spackman at Quantum praised Weatherspoon for "standing up to the tyranny of TCP."

- ***Cumulus: File System Backup to the Cloud***
  Michael Vrable, Stefan Savage, and Geoffrey M. Voelker, University of California, San Diego

Michael Vrable presented his work on Cumulus, which is a backup system designed for use with cloud-based storage systems. Cloud storage services are an emerging area of interest. The very high reliability that can be offered promises to greatly simplify backup, which continues to be a difficult problem. However, there is no clear solution to integrating with such a system.

Vrable began by classifying cloud storage systems along a spectrum from thick to thin. At the thick end, every aspect of storage is integrated, providing better efficiency and easier use. At the thin end, the user is provided with basic building blocks, but must develop their storage application independently; Amazon's S3 is an example.

Cumulus addresses the question of how effectively one can develop a backup service with the simple interface provided by thin cloud services. It operates over a simple get/put object API, with other operations being developed to run on the client. The goal of the system is to minimize resources and costs. Internally, a backup is structured as a directed acyclic graph that mirrors the file system hierarchy. Over time, new backups are created by duplicating the previous root node and using copy-on-write. The backup model is therefore incremental after the initial backup. Complicating this design is the fact that some storage providers may have per-file costs. To optimize for this, files can be collected into larger segments, but at the cost of potential segment fragmentation. Such a segment would increase costs by adding storage overhead. Cumulus implements a defragmentation utility to mitigate this problem.

In evaluating the system, Vrable showed the result of a seven-month trace of user data consisting of small data updates on the order of 10MB/day of new content and 30MB/day of modified content. Cumulus was compared in simulation against an optimal backup strategy. The assessment concluded that cleaning is necessary, as it maintains 95% segment utilization versus 50% without cleaning. It also established optimal values for segment size and cleaning threshold given a particular pricing structure. Two existing tools for S3 were found to operate at storage costs 19%–200% higher than Cumulus. Cumulus was also said to be competitive with Mozy, a thick cloud service that provides unlimited storage at $5 per month for noncommercial users.

Irfan Ahmad from VMware asked if Cumulus could provide deduplication or data scrubbing services. Vrable answered that Cumulus relies on the provider to provide reliability, although there is related work on auditing a service provider. As a client-oriented service, deduplication could be provided on the client side but could not capture duplication across clients.

DATA INTEGRITY

Summarized by Brandon Salmon (bsalmon@ece.cmu.edu)

- ***WorkOut: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance***
  Suzhen Wu, Huazhong University of Science and Technology; Hong Jiang, University of NebraskaóLincoln; Dan Feng, Huazhong University of Science and Technology; Lei Tian, Huazhong University of Science and Technology and University of Nebraska—Lincoln; Bo Mao, Huazhong University of Science and Technology

As systems increase in scale, online RAID reconstruction is likely to become a common mode of operation. To address this problem, WorkOut uses a "surrogate array" made of spare resources, such as spare RAID disks, to provide improved performance for arrays currently performing reconstruction.

Writes to new data, or writes that hit in cache sent to the reconstructing array, will be redirected to the surrogate array, and subsequent reads to this data can also be sent to the surrogate array. To route requests, the array keeps a mapping table in NVRAM. Once the array has finished reconstructing, the data on the surrogate array is recopied onto the original array, although overwrites may help speed this process as well.

Evaluation shows that the use of a surrogate array can decrease reconstruction time by up to 5x and also improves the latency of the foreground workload.

- ***A Performance Evaluation and Examination of Open-Source Erasure Coding Libraries for Storage***
  James S. Plank, University of Tennessee; Jianqiang Luo, Wayne State University; Catherine D. Schuman, University of Tennessee; Lihao Xu, Wayne State University; Zooko Wilcox-O'Hearn, AllMyData, Inc.

James Plank summarized and evaluated a variety of erasure coding schemes and libraries in order to give system designers without deep erasure coding expertise the ability to evaluate libraries for use in their systems.

The evaluations had several key results. First, the open-source erasure coding libraries can keep up with disks, even on slow processors. Within the codes, special-purpose RAID-6 codes were more efficient than general-purpose Reed-Solomon codes. Cauchy Reed-Solomon codes were also more efficient than Reed-Solomon codes. For Cauchy Reed-Solomon codes, the matrix choice was important to performance.

On the Mac machine they tested, the number of XORs was a good measure of performance, but caching behavior was also important on the Dell machine tested.

- ***Tiered Fault Tolerance for Long-Term Integrity***
  Byung-Gon Chun and Petros Maniatis, Intel Research Berkeley; Scott Shenker and John Kubiatowicz, University of California, Berkeley

This paper describes Bonafide, a key/value pair store designed to maintain fault tolerance over a long period in the face of Byzantine faults. To attack the problem, Bonafide divides the service into two tiers: a trusted tier, which must not fail, and an untrusted tier, which can fail frequently without compromising the service.

The trusted tier is able to make changes to state, while the untrusted tier is able to respond to read requests but not change the actual state of the system. This division allows Bonafide to leverage a large number of untrusted devices to prolong the life of the service while still maintaining appropriate fault-tolerance.

CONTROLLERS AND CACHING

Summarized by Chris Frost (frost@cs.ucla.edu)

- ***A Systematic Approach to System State Restoration during Storage Controller Micro-Recovery***
  Sangeetha Seshadri, Georgia Institute of Technology; Lawrence Chiu, IBM Almaden Research Center; Ling Liu, Georgia Institute of Technology

Disk controller firmware contains many interacting components (e.g., RAID, I/O routing, and error detection) and completes many asynchronous (concurrent) and short-running tasks per second. Current firmware recovers from transient errors by rebooting, halting all drive progress for around four seconds. Seshadri et al. want to increase the availability of drives without rewriting the large software base of existing controllers while supporting high performance and dealing with dynamic dependencies and complex recovery semantics. Their approach is to enable per-task recovery when possible, falling back to a system restart only very rarely.

When an error occurs within a task, the task may continue (ignore/correct for error) or retry (rollback), but only if no other tasks have been affected by state changes made by the failed task. The basis of Log(Lock) follows from all global state modifications being protected by locks (or similar primitives). These locks guide recovery. Log(Lock) tracks recovery points, starting points for recovery upon error, and whether each recovery point is safe to use given the current system state. Log housekeeping is split between the system and code the developer has added. The developer adds start-and-stop tracking calls associated with locks. With Log(Lock), a typical error causes a 35% throughput decrease for six seconds. A reboot would take four seconds and have zero throughput. In their experiments, task-level recovery is applicable for 99% of errors, and runtime overhead is less than 10%.

A student asked how interdependent the threads of context are or how often dirty reads exist. Seshadri answered that although dirty reads are present most of the time in many types of systems, they have seen that in disk controllers they rarely exist, because the time scales are so small. For example, a typical task completes in five milliseconds.

- ***CLIC: CLient-Informed Caching for Storage Servers***
  Xin Liu, Ashraf Aboulnaga, Kenneth Salem, and Xuhui Li, University of Waterloo

Kenneth Salem presented CLIC, an approach to implementing a hinted two-tier caching system. CLIC learns how to respond to hints rather than using prebuilt, ad hoc rules. Multiple levels of caching introduce two issues: caching an item at multiple levels can waste cache space, and caches farther from the client do not see all requests and so their temporal locality is poor. Cache hints—for example, a client writing a page with the intent to replace another page—allow caches to behave appropriately. However, existing hint implementations use manual rules, which do not support new hints without changes, may have poor responses to hints, and can be difficult to implement when multiple clients use the cache. CLIC is a hint-aware cache replacement policy for second-tier caches that learns appropriate hint responses rather than relying on manually specified rules. CLIC separates the generation of hints (generated by clients) from the interpretation of hints (the storage server).

In CLIC, each page in the cache is associated with the hint set with which it was most recently read or written. CLIC orders the hint sets by their learned priority and evicts pages associated with the lowest-priority hint set. Priorities are determined using the results of previous requests for the given hint set. CLIC takes steps to reduce the space needed

to track these statistics; in their measurements, CLIC needed less than 1% of the cache space for this tracking.

Kenneth compared the TPC-C and TPC-H hit ratios achieved by CLIC, an ad hoc hint policy (TQ), two policies that do not use hints (LRU and ARC), and the optimal policy (knowledge of the future) for varying cache sizes. CLIC and TQ usually dominated LRU and ARC, CLIC often dominated TQ, and OPT typically dominated all.

One audience member asked how CLIC performs with a large number of clients and with a varying dynamic workload. Kenneth answered that they have not tested with a large number of clients or time-varying workloads. However, at present, CLIC occasionally throws out all logs, using exponential decay. Another person asked how CLIC responds to clients that use hints to try to hurt other clients. Kenneth responded that CLIC helps the clients that benefit the most from the cache. Finally, someone asked whether the authors had thought of CLIC giving feedback to the client about which hints are the most useful. Kenneth said they had not and that this might be interesting.

- ***Minuet: Rethinking Concurrency Control in Storage Area Networks***
  Andrey Ermolinskiy and Daekyeong Moon, University of California, Berkeley; Byung-Gon Chun, Intel Research, Berkeley; Scott Shenker, University of California, Berkeley, and ICSI

Andrey Ermolinskiy discussed two limitations of using distributed locking to coordinate reads and writes among clients of a Storage Area Network (SAN), and he presented a new coordination approach that addresses these two limitations through optimistic, instead of strict, concurrency by adding logic to storage system nodes.

Using distributed locking to coordinate access to shared state has two issues: (1) it does not guarantee correct serialization of requests, and (2) it requires a majority of the locking server nodes to be available. Minuet guarantees the correct serialization of disk requests and removes the need to contact any locking server. Instead, the storage nodes themselves, via a guard, mediate requests and can reject incorrectly ordered requests. In Minuet, requests are augmented with session annotations that are used to order the requests. Distributed transactions can be constructed atop Minuet using logging and recovery.

Remaining challenges to Minuet-like systems include the adoption of guard logic into storage arrays, storage and bandwidth overheads of session metadata, and the programming model change of request rejection and forced lock revocations.

One audience member asked why an equivalent system cannot be built on top of SCSI-3 reservations. Andrey responded that he had not considered this. Another person asked about the efficacy of caching version numbers for concurrency control. Andrey answered that their implementation does scale; version numbers are stored in NVRAM on storage devices and perhaps could be stored on disk with some also in RAM.