conference reports

THANKS TO OUR SUMMARIZERS

8th USENIX Conference on File and Storage Technologies (FAST '10)

Sponsored by USENIX, the Advanced Computing Systems Association, in cooperation with ACM SIGOPS

San Jose, CA
February 23–26, 2010

OPENING REMARKS AND BEST PAPER AWARDS

FAST '10 Program Co-Chairs: Randal Burns, Johns Hopkins University; Kimberly Keeton, Hewlett-Packard Labs

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

Conference Co-Chair Randal Burns opened the 2010 File and Storage Technologies conference by thanking his fellow chair Kimberly Keeton and the individuals and groups that made the conference a success. He also announced the chairs for FAST '11: John Wilkes of Google and Greg Ganger of Carnegie Mellon University.

Kimberly Keeton followed Randal to present the Best Paper awards for the year. Kaushik Veeraraghavan accepted one award for "quFiles: The Right File at the Right Time." The paper considers a new data abstraction that allows multiple different physical representations of data to be held in a single logical container. His coauthors include Jason Flinn and Brian Noble at the University of Michigan and Edmund B. Nightingale of Microsoft Research. The second award was accepted by Swaminathan Sundararaman, Sriram Subramanian, and Remzi H. Arpaci-Dusseau. Their paper, "Membrane: Operating System Support for Restartable File Systems" details a set of operating system changes sufficient to support a file system that can transparently restart after an error and continue servicing all requests. They and their co-authors Abhishek Rajimwale, Andrea C. Arpaci-Dusseau, and Michael M. Swift all hail from the University of Wisconsin.

The opening ceremony also brought attention to the recent passing of Tom Clark. Mr. Clark was an innovator in SAN over IP, whose distinguished 20-year career took him to the forefront of companies such as Brocade, McDATA, and Nishan. He was an active member of SNIA, where he chaired the interoperability committee. He was also the author of three books that detail the technologies, protocols, and designs that constitute contemporary SANs and other storage virtualization systems.

Mr. Clark continues to be honored in an online memorial space at http://wtomclark.blogspot.com/ in lieu of a traditional funeral service. This site contains photographs, writing, and the memories of his many friends, colleagues, and admirers.

KEYNOTE ADDRESS

- ***Technology for Developing Regions***
  Eric Brewer, University of California, Berkeley

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

Eric Brewer provided insights into the challenges and successes that his group, TIER (Technology and Infrastructure for Emerging Regions), faces in working around the globe and the role of research in technology that aids the health and economies of developing areas. His presentation spoke to technology's potential to change the lives of the majority of the world's population—those who live on less than two dollars per day.

Dr. Brewer began by providing an overview of development work and described how his focus relates to traditional approaches. Technological research generally caters to the more than 1 billion people in the developed world to the exclusion of the rest of the world's population, which is expected to reach 6–8 billion in the next 25 years. For most of the past half-century, development has taken a top-down view, with agencies operating at large scale to improve macroeconomic indicators. Leveraging technology in such an approach is very difficult. Existing agencies lack experience with disseminating technological advances and the commercial distribution channels used in the developed world are a poor fit for the fragmented, rural, and impoverished markets that characterize the developing world. However, Dr. Brewer sees potential in fostering development in rural areas through technological advances that grow from the bottom up.

One of his group's goals is to improve rural connectivity. To do this, they have developed a WiFi-based system called WiLDNet. Although WiFi ranges are typically shorter, the cost is an order of magnitude lower, it is incrementally deployable, and it operates in unlicensed frequencies. In deploying this system, they developed a new IP stack that better utilizes the spectrum and set a new WiFi distance record at 382 kilometers. In another project, 25,000 patients have recovered their sight though small vision centers connected to doctors through a rural telemedicine system. These clinics in Tamil Nadu, India, are staffed by a nurse who can facilitate a computer-mediated consultation between a local patient and a doctor in a remote location. This project is expanding to 50 centers that, when complete, will service 2.5 million people.

Storage concerns were highlighted in the presentation, including lack of high-quality electrical service, use of flash and optical media to transmit data, and the urgent need to deploy more storage to safeguard history and culture. For example, radio stations in Guinea-Bissau and Madagascar broadcast in local languages but are unable to record their broadcasts. Meanwhile, most of the 6000 languages in Africa are dying and few recordings exist. A storage system to address these needs may have interesting characteristics. It would synchronize infrequently with a remote system, perhaps going weeks without connectivity. Thus, the focus must be on locally self-consistent versioning that can intermittently upload deltas to bulk data repositories.

Many members of the audience had questions for Dr. Brewer. When Margo Seltzer of Harvard asked how he got started, Dr. Brewer acknowledged that it was difficult. Three years after they began, none of their three initial projects was still in operation. However, their experiences can help other researchers enter the field more easily. Good early choices included starting in a country like India that is easy to work in and partnering with established NGOs with proven records.

In response to questions from Kimberly Keeton of HP Labs and others, Dr. Brewer described the Ph.D. path for his students. Considerable field work and human subjects approval from Berkeley are required. Theses focus on technical issues and contain "one or two chapters that any technologist could call their own." While each student may produce fewer publications, Dr. Brewer argues that they have more impact. In the last two years, seven doctoral students have followed this technology-oriented track, along with three to four social science students. It was also mentioned that while the initial funding for the project came from an NSF ITR grant, that grant is no longer available, which necessitates a larger number of smaller area-specific grants.

In closing, Dr. Brewer stressed that more emphasis should be placed on technology's ability to influence development efforts and that rural areas are currently the best place to focus. Decentralized development does work, but challenges such as connectivity and power need to continue to be addressed, and storage has a significant role to play.

BUILD A BETTER FILE SYSTEM AND THE WORLD WILL BEAT A PATH TO YOUR DOOR.

Summarized by Mike Kasick (mkasick@andrew.cmu.edu)

- ***quFiles: The Right File at the Right Time***
  Kaushik Veeraraghavan and Jason Flinn, University of Michigan; Edmund B. Nightingale, Microsoft Research, Redmond; Brian Noble, University of Michigan

  Awarded Best Paper!

Kaushik Veeraraghavan described quFiles, a file storage abstraction that enables users to access different views of data in different contexts. He presented the running example of a user accessing a video file from a desktop, laptop, TiVo, or smartphone: access from each of these has different demands on resolution, decoding complexity, and required network bandwidth and latency. quFiles serves as a unifying abstraction that multiplexes different views of a single file for each of these contexts.

At its core, quFiles consists of a context-aware mechanism that selects the best representation (view) of a file for a given context. The selection mechanism is encoded in four policies provided by a type-specific quFile creation utility.

Given a particular accessing device or context, the name policy serves zero or more file names for a given quFile, which also serves to specify the file type. The content policy provides a context-specific view (file content) for a given file name. The edit policy specifies whether a particular view is allowed or disallowed to be edited, or if an edit should automatically be versioned. Finally, the cache policy determines which views of a file should be cached on devices. These policies enable views of a quFile to be generated statically at quFile-creation time, or dynamically on access.

Kaushik then described four case studies on the use of quFiles in power management, copy-on-write versioning, resource-aware directory listings, and application-aware adaptation. Each of these studies illustrates the ease with which quFile policies may be implemented—each with less than 100 lines of code, written in a week or two. He then evaluated the cost of quFiles by showing they have application-level overheads of 1% and 6% in the warm and cold cache cases, respectively.

During the question period, Garth Gibson (Carnegie Mellon) asked if Kaushik thought about providing users with a means of locating and parsing all the policies supported by a quFile system, in order to understand the data manipulation being performed. Kaushik suggested that either a debugging mode or an additional view may be added to show, for a particular quFile, what data has been statically generated and what data can be generated dynamically.

- ***Tracking Back References in a Write-Anywhere File System***
  Peter Macko and Margo Seltzer, Harvard University; Keith A. Smith, NetApp, Inc.

Peter Macko presented a method for implementing data-block to inode back references that is generally applicable to write-anywhere file systems. Consolidating free space, data migration, partition resizing, and file defragmentation all require a costly-to-generate mapping of data blocks to the inodes referring to them. By encoding such a mapping within the file system itself, back references eliminate the need to generate such a mapping.

Macko described a set of challenges faced when implementing back references: the need to ensure a low, stable overhead of operations and to support deduplication (block sharing), to enable both inode versioning (read-only snapshots) and copy-on-write block duplicates (writable clones). He then described his log-structured back reference approach, using data block allocation and deallocation records stored in multiple B+ trees to adequately address each of the implementational challenges. He stated that this approach has been implemented in the btrfs (replacing the native back reference implementation) and ext3 file systems. Finally, he evaluated the time and space overheads of his back reference implementation, concluding that the space overhead is stable and the time overhead is less than 2%.

The audience expressed enthusiasm for the approach. Chris Small (NetApp) commented that the real advantage of the scheme is that instead of reading a huge data volume into memory, potentially taking hours, back references can be read into memory incrementally as needed. Rick Spillane (Stony Brook) asked if back references are implemented using a separate mechanism from file systems for snapshotting. Macko claimed that the mechanism is separate for the sake of a generalizable implementation, but that using existing snapshotting mechanisms would be a good optimization.

- ***End-to-end Data Integrity for File Systems: A ZFS Case Study***
  Yupu Zhang, Abhishek Rajimwale, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Zhang claimed that, according to the end-to-end argument, applications should be in charge of data integrity. However, since applications must share data and therefore must agree on a way to verify data integrity, such support is most feasibly placed in the file system. While current file systems attempt to preserve data integrity in the face of imperfect storage components (media, controllers, etc.) memory corruption due to hardware faults or software bugs is also a concern, particularly as memory capacity grows and file systems continue to cache a large amount of data in memory for performance.

In this case study, Zhang performed fault injection experiments on ZFS and showed that ZFS is robust against a wide range of disk corruption but fails to maintain data integrity in the presence of memory corruption. In particular, one-bit-flip memory errors are shown to have a non-negligible chance of corruption, which—depending on workload—may manifest when reading corrupt data (up to 7%), permanent writing of corrupt data (up to 4%), system crashes (up to 2%), and even operation errors due to corrupt interpretation of metadata. Zhang stated that, in most cases, the more memory consumed by the page cache, the more likely a corruption event occurs. He concluded that, as effort has gone into protecting against disk failures, it is worth the effort to protect against memory corruption as well.

When Dominic Giampaolo asked about the rate of bit flips in workloads, Zhang described their method of injecting k bit-flips (k>1) per workload and deriving the one-bit-flip probability from the k flip data. Another audience member asked how the rate of bit flip injections matches the rate of bit flip failures seen in recent memory corruption studies. Zhang explained that they did not yet determine how often corruption and crashes happen given real bit flip data, but suggested how that could be estimated. Bill Bolosky (Microsoft) asked whether the probability of memory corruption is uniform whether the system is busy or idle, and how much corruption could be improved by pushing the checksum earlier on reads. Zhang stated that more data is needed to draw a conclusion.

Finally, Roger Haskin (IBM) asked when one stops worrying about memory corruption, since that corruption could happen before any usage, in user buffers, and even in the very code responsible (with reduced probability due to size, but significant risk due to impact) for verifying integrity. He asked, more generally, why the file system should be the responsible entity for keeping memory in check versus hardware. Yupu answered that there are instances (e.g., memory corruptions induced by software bugs) in which higher-level techniques might be more appropriate.

LOOKING FOR TROUBLE

Summarized by Liangong Xu (lianghon@andrew.cmu.edu)

- ***Black-Box Problem Diagnosis in Parallel File Systems***
  Michael P. Kasick, Carnegie Mellon University; Jiaqi Tan, DSO National Labs, Singapore; Rajeev Gandhi and Priya Narasimhan, Carnegie Mellon University

Michael Kasick presented his results in diagnosing problems typically encountered by PVFS developers in off-the-shelf parallel file systems: for example, the "limping-but-alive" server problems where a single faulty server impacts overall system performance but can't be identified with logs. The problems people are particularly interested in are storage-related (e.g., an accidental launch of rogue processes) and network-related (e.g., packet loss). The target parallel file systems in the paper are PVFS and Lustre. The key insight about these system is that clients typically communicate with all the servers in a request, while servers are isolated. With this important feature, it is assumed that fault-free servers exhibit similar performance metrics, while faulty servers exhibit dissimilarities only in certain metrics.

The proposed diagnostic algorithm is based on three assumptions. First, hardware is homogeneous and identically configured. Second, workloads are non-pathological, that is, requests are well distributed across the servers. Large requests touch almost every server, and small requests, which may exhibit migrating load imbalances, are well balanced over a period of time. Third, the majority of servers exhibit fault-free behavior. The diagnostic algorithm contains two phases: node indictment to find the faulty node, and root-cause analysis to find the cause of the faulty behavior. A histogram-based approach is used for most metrics in node indictment. This is done in four steps: computing a probability distribution function metric for each server over a sliding window, computing a Kullback-Leibler divergence for each server pair, flagging an anomalous pair if its divergence exceeds the threshold, and flagging a server if over half of its server pairs are anomalous. The threshold is selected in a fault-free training session. For root-cause analysis, the approach is to build a predefined table of metrics and faults. For example, if storage throughput is diagnosed as a problem in the first phase, then by simply looking up the table we know the root cause is probably a disk hog.

Someone asked about the scalability of the algorithm, since it has to compute the K-L divergence for each server pair. Kasick gave a potential solution in which the nodes can be partitioned into several groups and the computation only happens within a group. This way the complexity of the algorithm can be reduce to O(N). Another concern was the assumption that workload across all the disks would be evenly distributed, which may not hold in real systems. Kasick argued that this is acceptable as long as the workload is well balanced over a period of time.

- ***A Clean-Slate Look at Disk Scrubbing***
  Alina Oprea and Ari Juels, RSA Laboratories

Arkady Kanevsky from EMC gave the presentation.

The authors propose a smart disk-scrubbing strategy called "staggering" to adaptively change the scrubbing rate following an error event and to sample across disk regions in order to discover errors faster than by sequential reading. Latent sector errors (LSEs) are discovered only when a sector is read and so have significant impact on a system without redundancy or even RAID 5. The primary approaches used to counter LSEs are intra-disk redundancy and disk scrubbing; this paper focuses on the latter. Traditional disk scrubbing uses sequential reading of disk sectors with a fixed predetermined rate. However, according to the Sigmetrics 2007 study, LSEs exhibit temporal decay, temporal locality, and spatial locality. These features enlarge the design space of smarter scrubbing strategies to account for distribution of LSEs and disk history.

According to Bairavasundaram et al. (2007), inter-arrival time distribution has very long tails; more LSEs develop shortly after a first LSE, and LSEs develop clustered on disk at block logical level. Taking advantage of these observations, the staggering strategy partitions the disk into multiple regions, each consisting of a number of segments. Disk scrubbing is done in multiple rounds of scanning. In every round the regions are accessed one by one, but for every region only one segment is touched at a time. Once an error is detected in a segment, the entire region containing the segment is considered suspect and scanned. Due to the Sigmetrics 2007 result that all errors are clustered in a 128MB region with a very high probability, the region size is set to 128MB, while the segment size is set to 1MB to amortize the overhead of positioning. Also according to Bairavasundaram et al., the LSE rate is fairly low and constant in the first two months of drive operation, after which it increases but remains fairly constant. As a result, the staggered strategy adaptively changes the scrubbing rate. Disk lifetime is divided into four periods. In the first two months the scrubbing rate is fairly low. After this the disk enters the pre-LSE period, with a higher scrubbing rate. When an error is detected, the rate is raised to the highest level. After a period during which no further errors are detected, the disk enters the post-LSE period, when the scrubbing rate is decreased but still higher than in the pre-LSE period. Note that the authors use a new metric, mean latent error time, for single drive reliability in evaluating their simulation results.

Arkady referred the audience to the authors for answers to their questions.

- ***Understanding Latent Sector Errors and How to Protect Against Them***
  Bianca Schroeder, Sotirios Damouras, and Phillipa Gill, University of Toronto

Bianca Schroeder examined the effectiveness of current protection schemes against latent sector errors (LSEs) and provided detailed insight into LSEs' characteristics. Currently there are mainly two ways to protect against LSEs: periodic scrubbing and intra-disk redundancy. In order to evaluate how effective these approaches are, the authors leveraged the data from NetApp storage systems, which provides time of detection and logical block number for each LSE in 1.5 million drives over a period of 32 months. Analysis of this real-world data revealed that among disk scrubbing schemes, localized scrubbing and accelerated scrubbing don't bring significant improvement, but staggered scrubbing performs very well. For intra-disk redundancy, the simplest approach is single parity check (SPC); however, this fails to recover 20–25% of the disks. A stronger approach is to introduce additional parity, which, unfortunately, also adds more overhead in updating parity but may be useful in certain environments. Interleaved Parity Check (IPC) requires only one parity update per data update, can tolerate up to m consecutive errors, and is claimed to perform as well as MDS due to simulation results. Nevertheless, the result in the paper shows this scheme is much weaker than MDS, which implies the importance of using real-world data to evaluate these algorithms.

Besides the high-level summaries above, Schroeder also talked about a number of interesting questions in practice. First of all, what level of protection is it appropriate to use when? Data analysis shows that more previous errors implies a higher chance and higher number of future errors; the number of errors in the first error interval increases the expected number of future errors but doesn't significantly increase the probability of future occurrence; the probability of errors since the first occurrence drops off exponentially. Taking all these pieces together allows people to do more careful adaptive design. The second concern is whether all areas of the drive are equally likely to develop errors. According to the analysis, up to 50% of errors are concentrated in the top and bottom area of a drive; the cause of this, however, remains an open question. Is scrubbing potentially harmful to drives? NetApp doesn't provide workload data, so the authors use data collected in the Google datacenter. The result shows that there is no correlation between LSEs and the number of reads or writes. However, Schroeder stated that this conclusion needs more investigation. What is the common distance between errors? The answer is that there does exist a probability concentration, which potentially implies the insufficiency of SPC. Are errors that are close in space also close in time? Yes.

Brent Callaghan from Apple asked whether the high error rate in the beginning of the drive is caused by heavy use of this region—for example, frequent metadata operations. Shroeder answered that this question needs further investigation, because she didn't see support for this from Google's data. Another person asked about the number of parity sectors in MDS that were needed to tolerate such centralized errors. Shroeder said that spreading the parity group may help but MDS doesn't really need very large regions to protect. The real issue about MDS is its expensive update, but that is fine as long as update operations are not frequent.

WORK-IN-PROGRESS REPORTS (WIPS)

Summarized by Avani Wildani (agadani@gmail.com)

- ***MTTDLs Are Meaningless: Searching for a Better Metric for Storage Reliability***
  James S. Plank and William E. Pierce, University of Tennessee

James Plank made the case that Mean Time To Data Loss (MTTDL) is a useless, malicious metric for conveying information about disk failure probabilities. He pointed out that it is difficult to create honestly, since it requires foreknowledge about when a particular device will fail and is sensitive to parameters that are difficult to get. MTTDLs are alluring because they are relatively easy to calculate from data and derive from analytical models. Instead, Plank proposed that we look for a metric that focuses on the probability that a given disk will fail in a given, near-term timeframe. He ended the talk by listing existing competitors to MTTDL, such as bit half-life (which he describes as a median measurement), DDF per 1000 RAID groups, and data loss per petabyte-year. A commentator added "bits lost per year" during the questions phase. Plank ended his talk by mentioning a simulation environment in place to evaluate different failure time estimates comparatively.

- ***Upgrades-as-a-Service in Distributed Systems***
  Tudor Dumitras and Priya Narasimhan, Carnegie Mellon University

Dumitras addressed the problem of having to take down large data stores for scheduled maintenance such as system upgrades. He pointed out that most episodes of downtime in distributed systems are a result of these scheduled events. Using Wikipedia as a sample service, he examined what prevents an in-place upgrade. It turned out that the leading reason for offline upgrades is incompatible database schemas between releases. Over time, this results in more and more downtime for larger upgrades.

The proposed solution to this problem is to isolate the production system from the upgrade. This can be expressed as trading extra resources (which most large distributed systems have) for a reduction in scheduled downtime. Doing an upgrade online requires having two copies in production, typically, to avoid inconsistencies as data keeps coming in. The new solution allows you to put a new version on a different hardware or virtualization platform entirely and not worry about its consistency until it's swapped into place for the running system. He finished by describing case studies.

- ***Down with the VFS and His Insidious Caches!***
  Richard P. Spillane and Erez Zadok, Stony Brook University

Spillane made a strong case for taking VFS out of the kernel and putting its functionality into user space. It's simple but unsafe for clients to directly access the server cache, since they are accessing each other's shared memory. The solution Spillane proposes is to take advantage of an unused ring in the CPU to introduce a new security level between user space and kernel space that lets you trap calls and essentially gives every client their own message stack. Spillane's results show that working in user space has almost no effect on performance with this construction: his system rivals ReiserFS for speed. In the future, he intends to generalize the construct so that anyone can be a privileged library, in addition to further extending mmap. He points out that letting you drop in alternative caching would let the cache be customized for the workload and include additions such as provenance information that are less widely implemented in current caches, and he ended with a description of how one size doesn't fit all for VFS. A questioner pointed out that some CPU architectures, such as the Alpha and MIPS, only have two rings.

- ***DiskReduce: RAIDing the Cloud***
  Bin Fan, Wittawat Tantisiriroj, Lin Xiao, and Garth Gibson, Carnegie Mellon University

Even in the cloud, there is a strong need to save space as we keep generating more and more data. Yet, given the number of unknowns in cloud storage, data reliability is at the forefront of customers' minds. Unfortunately, the triplication of GFS or HDFS is not space-efficient, and traditional RAID is complex for developers. Xiao proposed a simple solution to address this complexity to bring RAID into the cloud. The core idea is that the data would be triplicated first and then gradually RAID-ed as the data became staler or resources got tight. Pushing off the encoding step has the additional benefit of letting you encode while the system is idle, hiding the computation cost. Xiao's implementation employs a metadata server to keep track of what data blocks need to be grouped together and what to repair in case of failure.

- ***Enabling Scientific Application I/O on Cloud File Systems***
  Milo Polte, Carnegie Mellon University; Esteban Molina-Estolano, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory; Scott Brandt, University of California, Santa Cruz; Garth Gibson, Carnegie Mellon University; Maya Gokhale, Lawrence Livermore National Laboratory; Carlos Maltzahn, University of California, Santa Cruz; Meghan Wingate, Los Alamos National Laboratory

Polte addressed the problem of running legacy high-performance computing software on modern cloud-based supercomputers such as those at Google. He pointed out that these computers can, in some areas, vastly outperform localized supercomputers, even large ones like Jaguar which can reach 1.75 petaflops. In contrast, he estimates Google at maybe 100 petaflops four years ago. Parallel scientific applications are written assuming a standard POSIX model with MPI that cloud systems cannot directly replicate. To counter this, Polte proposes the addition of a transition layer to easily port these legacy applications to take advantage of modern cloud systems. Polte's virtual interposition layer allows the applications to pass their writes to the underlying hardware without any application modifications. The transition layer creates a directory in the cloud with one datalog per layer and permits reopens and concurrent writes. Writes are decoupled and reads are aggregated.

- ***Non-Volatile Transactional Memory***
  Joel Coburn, Adrian M. Caulfied, Laura M. Grupp, Ameen Akel, Rajesh K. Gupta, and Steven Swanson, University of California, San Diego

Coburn introduced non-volatile transactional memory (NVTM). He claimed that storage will eventually be as fast as DRAM, even though the current cost in latency of going to disk is prohibitive. As new storage-class memory devices such as phase change RAM, scalable two-transistor memory, and memristors arrive, we will be faced with technologies that are so fast that system latency is OS-dominated instead of disk-dominated. In this situation, not addressing the OS dominance is equivalent to leaving 100x performance on the table. Coburn proposed to revise the entire stack with a focus on OS latency. The application would directly access the NVTM while the OS would handle allocation, mapping, and other administrative tasks. This was compared with Stasis, which is half as fast at best.

- ***Verifying Massively Data Parallel Multi-Stage Computations***
  Osama Khan and Randal Burns, Johns Hopkins University

Khan concentrated on the assurance of the correctness of parallel computations. Since parallelizing constructs are popular over untrusted machines, there is a ready need for the security of a verified model. Khan proposes a graph-based verification model that encapsulates parallel computations without restricting the message format. His mechanism is adaptable for other platforms since it is purely probabilistic. He proceeded to demonstrate his model and describe the underlying hash tree mechanism. His system shows 99% problem detection with only 5% overhead.

- ***Hifire: A High Fidelity Trace Replayer for Large-Scale Storage Systems***
  Lei Xu and Hong Jiang, University of Nebraska; Lei Tian, Huazhong University of Science and Technology

Accurate replaying of traces is critical to researching a new system. Xu pointed out that current approaches are not scalable or portable, and they have performance issues. He maintained that a good storage benchmarking tool needs high fidelity, scalability, performance, and portability. Hifire combines different sources of inputs and issues I/Os to external devices. Hifire primarily consists of a scheduler and I/O device layer. Currently, Hifire is implemented on a Linux server in 1500 lines of C++ as a user-space process. The preliminary results show that 96% of I/Os were issued in 10 microseconds, with 80% in just one microsecond. This is significantly better than Buttress, which issued 90% of I/Os in 50 microseconds. Xu intends to extend this work to support large-scale systems and filesystem imaging. Finally, he intends to open source Hifire for the research community.

- ***Energy Efficient Striping in the Energy Aware Virtual File System***
  Adam Manzanares and Xiao Qin, Auburn University

Manzanares talked about energy-efficient striping in virtual file systems. The VFS manages locations of files, load balances, manages disk states, and allows clients to access data in a storage system. Manzanares introduced the energy-aware virtual filesystem (EAVFS) server, which distributes data across storage nodes. Instead of following the current model of putting disks into standby mode, which can be inefficient, Manzanares proposes a tiered approach that simplifies management of files and disks. The file system tracks file accesses, and groups of storage nodes together handle writing and striping a file. Popular data ends up on the buffer disk, and the number of disks can be matched to the nearest bottleneck. The net result of these tactics was a 12.5% energy savings per storage node.

- ***The Hot Pages Associative Translation Layer for Solid State Drives***
  Luke McNeese, Guanying Wu, and Xubin He, Tennessee Technological University

Solid state drives typically need a translation layer to map data to blocks of flash to ensure sufficient wear leveling on an SSD. Unsurprisingly, the mapping unit of the flash translation layer (FTL) has a direct correlation with its performance. For a write workload, overwrites dominate. In hot pages, the uneven accesses can cause significant overhead, whereas cold pages need fewer resources. The main idea of McNeese's work is to separate hot and cold pages so that the best schemes are used for the workload in question. His system, HPAT, sends the workload to the appropriate handler. HPAT also decides where random writes go, ensuring even wear leveling.

- ***Security-Aware Partitioning for Efficient File Systems Search***
  Aleatha Parker-Wood, Christina Strong, Ethan L. Miller, and Darrell D. E. Long, University of California, Santa Cruz

Parker-Wood described her system for security-aware partitioning for fast filesystem search. Partitioning techniques that subdivide indices are a proven way to improve metadata search speeds and scalability for large file systems, permitting early triage of the file system. A partitioned metadata index can rule out irrelevant files and quickly focus on files that are more likely to match the search criteria. However, security is also a concern: in a multi-user file system, a user's search should not include files the user doesn't have permission to view.

Security-aware partitioning, unlike Smartstore or Spyglass, incorporates security from the ground up. The system is set up such that if you can see anything in a partition, you can see everything in a partition. Partitions are created and re-arranged as dictated by changes in permissions in user data. Parker-Wood also outlined a set of evaluation criteria for evaluating partitioning algorithms focusing on information-theoretic indicators instead of statistics on synthesized queries.

- ***InfoGarden: A Casual-Game Approach to Digital Archive Management***
  Carlos Maltzahn, Michael Mateas, and Jim Whitehead, University of California, Santa Cruz

Maltzahn presented InfoGarden, a video game designed to make the task of tagging personal archives entertaining, thus leading to richer metadata for the archived system. People find digital archives overwhelming; it's easy to store data, and it's easy to lose it. Since much of this data is private, it cannot be crowd-sourced using currently available technologies. Tagging metadata as a game follows in the footsteps of successful programs such as PSDoom, which enticed system administrators to kill rogue processes by presenting those processes in the context of a first-person shooter. Similarly, InfoGarden shows untagged files as weeds in a garden representing a personal archive. Crosshairs allow you to tag a weed, turning it into a plant. Points are awarded based on the number of weeds successfully converted.

- ***Fuse for Windows***
  Mark Cariddi, Tony Mason, Scott Noone, and Peter Viscarola, OSR—Open Systems Resources

Mason talked about helping developers build kernel technologies under Windows. The goal is to enable customers to build customized file systems that run in user mode. Their system, FUSE, was designed to look as much like a Microsoft-provided framework as possible. Mason claims that much of the effort was in figuring out cygwin and gcc, among other tools from the UNIX community. Currently, they've implemented a system that can handle much of their intended domain, though chown and extended attributes are not yet implemented. The performance impact is significant: up to 40% for operations such as "open" that are small and involve a significant amount of context switching. This system will be freely available for non-commercial use.

- ***Revisiting I/O Middleware for the Cloud***
  Karthik Kambatla, Naresh Rapolu, Jalaja Padma, Patrick Eugster, and Ananth Grama, Purdue University

Kambatla discussed I/O middleware for the cloud. Applications in the cloud have different consistency and availability requirements, and current scenarios use specialized systems for each of these requirements. Poor resource utilization can increase cost. Although many sophisticated storage systems have been proposed, many of them are overkill in terms of resource overhead. Kambatla compared writes over different cloud file systems to track memory usage. The key idea is to just index the data and store it in a log-structured file system. This gives you efficient random reads, writes, and lock-free reads. One goal that they have met is to get sequential access performance better than a standard key-value store such as Bigtable, but they are still working on an efficient implementation for random accesses.

- ***Determining SLO Violations at Compile Time***
  Kristal Curtis, Peter Bodik, Michael Armbrust, Armando Fox, Michael Franklin, Michael Jordan, and David Patterson, University of California, Berkeley

Web applications are a competitive field, and the most interesting applications answer non-trivial queries which in turn imply heavy computation costs. Everyone wants the interesting applications, and they want some idea of how well they will work. Curtis introduced the idea of SLOs for Web applications to gain some understanding of level of service being provided. SLOs could examine the trade-offs between strong consistency and availability that applications make. PICL is a query language that can estimate, at compile time, the latency of a query by sampling from a historical histogram of operators. While it's easy to combine latency estimates, it's difficult to make the model portable across queries and load conditions. Currently, the system is written and works for queries that have already been seen. Curtis is working to extend the system to new queries.

- ***LazyBase: Freshness, Performance, and Scale***
  Craig A.N. Soules, Kimberly Keeton, and Charles B. Morrey III, Hewlett-Packard Laboratories

Soules discussed enterprise information management in the context of the LazyBase management system. Typically, one runs analysis, collects metadata, and stuffs it all into one giant database. This ends up with lots of updates, and ideally these updates do not update queries dramatically. If the new data was continuously in demand, the system would fall over, but luckily many applications can work with stale data. The solution Soules proposes involves batching updates to increase throughput and isolating queries from the ingest pipeline. In LazyBase, applications can specify the level of freshness that they desire. When applications send in queries, they will initially get staler data, and as they wait LazyBase will return fresher and fresher data. There are several questions remaining in this work. Soules is working on understanding the trade-off between ingest and query performance, as well as the consistency and security ramifications of the system.

- ***Gridmix3: Emulating Production IO Workload for Apache Hadoop***
  Chris Douglas and Hong Tang, Yahoo! Inc.

Douglas introduced Gridmix3, a system to emulate production I/O for Hadoop. The goal is to take conditions seen in production and replicate them in a controlled environment to predict how these conditions will affect the system when it is deployed. Basically, they want to catch bottlenecks and bugs using a synthetic workload, since it's difficult to mirror actual production servers. There are many different types of jobs that they hope to emulate in this system, and they need to accommodate many factors to reliably predict performance of the cluster. They hope to distill sample workloads into an anonymized digest and make it available.

- ***The Case for a New Sequential Prefetching Technique***
  Mingju Li, Swapnil Bhatia, and Elizabeth Varki, University of New Hampshire

Li made the case for sequential prefetching techniques. These techniques are simple but effective and commonly used in storage caches. The system has some extra costs: the system will see some extra traffic from un-needed prefetches and early eviction, and cache pollution will continue to be a concern. However, sequential prefetching has the benefit of low mean response time along with the possibility of combating the extra costs with piggybacked prefetching, which could reduce the net system traffic. She compares several techniques and comes to the conclusion that while prefetch on hit (PoH) is the best technique naively, this is workload-dependent and could fail spectacularly on workloads that, say, tend to use data once sequentially. Li proposes to take advantage of a combination of techniques based on using a minimal amount of cache space to help reduce workload dependency and increase the hit ratio. The system load is auto-detected so that there is less prefetching when the load is high. Her ongoing work involves performance evaluation and further design details.

POSTER SESSION

First set summarized by Simona Boboila (simona@ccs.neu.edu)

- ***Router Caching for Video Streaming Systems***
  Jiawu Zhong, Zhicong He, Jun Li, Xin Wang, and Jin Zhao, Fudan University

Xin Wang proposed using the capabilities of storage and routers to increase the performance of applications. Preliminary evaluation was performed for content distribution networks and peer-to-peer systems. In content distribution networks, results show reduction of bandwidth cost with the use of caching routers. For peer-to-peer systems, the authors obtained a reduction of the server load of approximately 30%.

- ***Router-supported Data Regeneration in Distributed Storage Systems***
  Jun Li, Tiegang Zeng, Lei Liu, Xin Wang, and Xiangyang Xue, Fudan University

Xin Wang observed that in a distributed network, links have to share bandwidth, which can generate too many flows in the network. To address this problem, the authors propose the use of routers which encode the flows along the same link in a single flow. Preliminary results on a network topology with 100 routers and 197 links show that traffic was significantly reduced with supporting routers.

- ***P-Warn: Adaptive Modeling of Energy Consumption Predictor and Early-Warning in Datacenters***
  Jianzong Wang, Rice University; Changsheng Xie, Jiguang Wan, Zhuo Liu, and Peng Wang, Huazhong University of Science and Technology, Wuhan National Laboratory for Optoelectronics

Jianzong Wang noticed that energy consumption changes with configuration and usage in datacenters. The authors built P-Warn, whose goal is to provide power predictors and give early warnings about energy consumption. The prediction model is based on several parameters: CPU utilization, I/O bandwidth, and temperature. Current work shows a trade-off between the performance overhead and the accuracy of predictions.

- ***Unix-like Access Permissions in Fully Decentralized File Systems***
  Bernhard Amann and Thomas Fuhrmann, Technische Universität Munchen

Bernhard Amann proposed securing a directory tree in a file system with a hash tree, thus providing confidentiality, authenticity, and access permissions for fully decentralized untrusted storage. This approach has several advantages: it achieves fork consistency against rollback attacks, uses fast asymmetric cryptography for improved performance, and enables ACLs to be layered on top.

- ***An Adaptive Chunking Method for Personal Data Backup and Sharing***
  Woojoong Lee and Chanik Park, Pohang University of Science and Technology

Woojoong Lee proposed an adaptive chunking model to optimize file I/Os using data deduplication. With this approach, the appropriate chunking method of a file (fixed-size static chunking, SC, or content-defined chunking, CDC) is dynamically determined based on deduplication efficiency in accordance with file types and the device's capabilities for computation. The authors also propose an algorithm to minimize the switching overhead from SC to CDC.

- ***PosFFS2: A New NAND Flash Memory File System Supporting Snapshot in Embedded Linux***
  Woojoong Lee and Chanik Park, Pohang University of Science and Technology

Sejin Park pointed out the problem of recovering data that users erase by mistake in a flash-based file system for embedded Linux. The proposed solution is building a file system, PosFFS2, which supports snapshots. With this approach, the old copy of a page remains in the system, as a snapshot and can be recovered later by the user. Preliminary results show a low storage overhead, about 5%, created by stored metadata.

- ***SLIM: Network Decongestion for Storage Systems***
  Madalin Mihailescu, Gokul Soundararajan, and Cristiana Amza, University of Toronto

Madalin Mihailescu pointed out the problem of increasing I/O needs in datacenters. To address this issue the authors propose SLIM, which uses rack-level resources to reduce network-storage traffic, thus increasing performance for low-cost network storage. In particular, SLIM uses rack-level persistent write-back cache to facilitate I/O optimizations. Preliminary results show reduced network traffic of up to 80%.

- ***Protecting a File System from Itself***
  Daniel Fryer, Angela Demke Brown, and Ashvin Goel, University of Toronto

Daniel Fryer noticed that even stable file systems have bugs which can corrupt data. He proposes to address this problem by checking transactions against some invariants before committing to disk. This approach raises a few research questions: what kind of invariants can be checked quickly, how do we specify the invariants, how thoroughly can we specify file system correction? In the current prototype implementation, the authors choose reproducible bugs, identify violated invariants, and implement checking functions for the invariants.

- ***Performance Assurance of Distributed Storage by Application-driven, Advanced and Time-based Reservation***
  Yusuke Tanimura, Hidetaka Koie, Tomohiro Kudoh, Isao Kojima, and Yoshio Tanaka, National Institute of Advanced Industrial Science and Technology

Yusuke Tanimura argued that storage access is an important bottleneck of IT systems. The proposed solution is to allow application users to explicitly reserve I/O throughput in advance. The allocation is done on a first-come reservation basis. The authors presented the system architecture and two case studies of simultaneous access to a server, which may result in access conflict. The implementation uses EBOFS developed by Ceph, enhanced with space reservation functionality.

- ***Design and Implementation of a Metadata-Rich File System***
  Sasha Ames, University of California, Santa Cruz/Lawrence Livermore National Laboratory; Maya B. Gokhale, Lawrence Livermore National Laboratory; Carlos Maltzahn, University of California, Santa Cruz

Sasha Ames pointed out the problem of separating raw data stored in traditional file systems from related, application-specific metadata stored in relational databases. He argued that this separation triggers consistency and efficiency concerns and proposed a metadata-rich file system in which files, user-defined attributes, and file relationships are all first class objects. Unlike previous approaches, the authors use a graph data model composed of files and their relationships. Preliminary results show an increase in performance of up to 20 times obtained with the current prototype, compared to relational databases, e.g., PostgreSQL.

- ***Frequency Based Chunking for Backup Streams***
  Guanlin Lu, Yu Jin, and David H.C. Du, University of Minnesota

Guanlin Lu argued that data backup is a necessity in several contexts—for example, to minimize financial and business loss. This work focuses on chunking deduplication, a method to eliminate duplication among data backups. The authors designed a chunking algorithm with the goal of identifying as much duplicate data as possible while delivering a small number of chunks. Their approach significantly reduces the overhead of data processing and the cost of metadata.

- ***Upgrades-as-a-Service in Distributed Systems***
  Tudor Dumitras and Priya Narasimhan, Carnegie Mellon University

Tudor Dumitras argued that current approaches addressing dependable, online updates in enterprise systems are prone to failures, because the upgrade is not an atomic operation. Thus, hidden dependencies among the distributed system components may break during the update. To ensure atomicity, the authors propose isolating the old version from the upgrade procedure and dedicating separate resources to the new version. Current results obtained through fault injection prove that their system is more reliable than online-upgrade approaches.

Second set summarized by Priya Sehgal (priya.sehgal@gmail.com)

- ***NSFv4 Implementations: Who Performs Better, When, and Why***
  Vasily Tarasov, Sujay Godbole, and Erez Zadok, Stony Brook University

In this study, Vasily Tarasov and his team performed an extensive end-to-end NFSv4 performance evaluation across the multi-dimensional space of client and server implementations, workloads, NFS topology, and OS parameters. They created AuDiN, an automated evaluation framework capable of distributed filesystem benchmarking across diverse workloads. Based on the configuration file, AuDiN automatically installs the required OSes, sets up local file systems and other server- and client-specific parameters, prepares NFS export and mount points, then runs the benchmarks. Vasily observed a lot of variation under different setups. Under certain workloads (e.g., Web server) NFS performance was sensitive to the client's selection, whereas in another case (file server) performance was not affected by the client selection. He also observed performance improvements across different server platforms(~2–3x).

- ***Enabling Scientific Application I/O on Cloud FileSystems***
  Milo Polte, Carnegie Mellon University; Esteban Molina-Estolano, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory; Scott Brandt, University of California, Santa Cruz; Garth Gibson, Carnegie Mellon University; Maya Gokhale, Lawrence Livermore National Laboratory; Carlos Maltzahn, University of California, Santa Cruz; Meghan Wingate, Los Alamos National Laboratory

Milo Polte claimed that there are a large variety of scientific applications, such as climate simulations or astrophysics, that require a POSIX file system or MPI I/O interface. These semantics are not supported by cloud file systems (e.g., HDFS). So, the goal of their research was to allow unmodified scientific applications to run on a cloud file system. They achieved this goal by creating an interposition layer between the applications and the cloud file system that provides a POSIX interface to the applications and performs HDFS translations on the other side. The interposition layer is implemented in the parallel-log structured file system.

- ***Verifying Massively Data Parallel Multi-Stage Computations***
  Osama Khan and Randal Burns, Johns Hopkins University

Osama Khan presented a technique to ensure the correctness of massively parallel computations and data analysis that take place in remote untrusted machines. This approach uses a graph-based model, where the vertices denote sequential code blocks, while the edges denote data paths in the system. The verification mechanism is based on collecting commitments to the input and output data at each stage of the computation and then redundantly computing the results of a small subset of computations in that stage. Their technique relies on random sampling and outputs probabilistic guarantees. Their verification mechanism can be easily adapted to a variety of platforms such as Dryad and MapReduce.

- ***An Erase and Destage-Efficient Write-Buffer Management Algorithm for Flash Memory SSD***
  Jian Hu and Hong Jiang, University of Nebraska; Lei Tian, Huazhong University of Science and Technology

Jian Hu presented a flash-aware write-buffer management algorithm called PUD-aware LRU algorithm (PUD-LRU) that was based on the Predicted average Update Distance (PUD) as the key block replacement criterion on top of FTL schemes. This work was prompted by the strong temporal locality observed in a few server workloads (e.g., TPC). The main idea of PUD-LRU is to differentiate blocks and destage them judiciously based on their frequency and recency so as to avoid unnecessary erasures due to repetitive updates. They have implemented PUD-LRU in FlashSim. The preliminary results showed that PUD-LRU reduced the number of erasures and average response time over BPLRU by up to 65% and 64%, respectively.

- ***Energy Efficient Striping in the Energy Aware Virtual File System***
  Adam Manzanares and Xiao Qin, Auburn University

Adam Manzanares presented the energy-aware virtual file system (EAVFS) and energy-aware striping within it. EAVFS consists of a server node and one or more storage nodes. The server node is responsible for distributing data across the storage nodes, while the storage nodes consist of the actual data disks and manage the placement of the data on these disks. The storage nodes consist of a "buffer disk," which contains the most popular data. These storage nodes are divided into groups; a file is contained or striped within one group of storage nodes. Adam explained that to implement energy-aware striping, the storage nodes duplicate the stripes of the popular files into the buffer disks, putting the other disks into standby mode or powering them off, thereby reducing energy.

- ***MIND: Modeling Power Consumptions for Disk Arrays***
  Zhuo Liu, Fei Wu, Changsheng Xie, and Jianzong Wang, Huazhong University of Science and Technology and Wuhan National Laboratory for Optoeletronics; Shu Yin and Xiao Qin, Auburn University

In this work, Zhuo Liu talked about how they modeled disk array power consumption under certain workloads. This technique made use of DiskSim, the disk simulator, and tried to determine the amount of power consumed by the disk array when the workload is subjected to different RAID algorithms. The energy calculator estimated the power based on the different modes (e.g., spin up/down) of the disk and transitioning costs from one mode to the other.

- ***Investigating Locality Reformations for Cluster Virtualization***
  Ferrol Aderholdt, Benjamin Eckart, and Xubin He, Tennessee Technological University; Stephen L. Scott, Oak Ridge National Laboratory

Benjamin Eckart proposed the idea of improving performance of workloads running on a cluster of virtual machines by exploiting the property of locality of data. He proposed that if we moved the virtual machines closer or onto the nodes that contained the actual data, it would help improve performance. Currently, they pin the I/O bound applications running on a VM to a dedicated core in a cluster node. This improves the locality, as it avoids cache misses, resulting in better performance.

- ***VM Aware Journaling: Improving Journaling File System Performance in Virtualization Environments***
  Ting-Chang Huang, National Chiao Tung University, Taiwan; Da-Wei Chang, National Cheng Kung University, Taiwan

Ting-Chang Huang proposed improving the performance of journaling file systems running in a virtualized environment. The main idea is that, unlike traditional journaling approaches, which write journal data to the on-storage journal area, VM-aware journaling retains the data in the memory of the virtual machine and stores the information for locating the journal data (called the journal information) in the Virtual Machine Monitor (VMM). As a consequence, journal writes to storage are eliminated. Since only the metadata for locating the journal data is maintained in the VMM memory, it does not increase memory pressure.

- ***Security Aware Partitioning for Efficient File Systems Search***
  Aleatha Parker-Wood, Christina Strong, Ethan L. Miller, and Darrell D.E. Long, University of California, Santa Cruz

Aleatha Parker-Wood explained that partitioning is a system designed for indexing a file system. They have designed a security-aware partitioning algorithm, where if someone can see a file in a partition they can access every file in that partition. Their algorithm can add or eliminate the partitions for search depending upon the user's permissions, thereby obviating the need for expensive filtering operations. To evaluate their algorithm, she proposed building a mathematical model of what a query might look like and how their algorithm would perform under different types of queries. They are looking at different dimensions for evaluation: intra-partition similarity, inter-partition similarity, and partition size.

- ***InfoGarden: A Casual-Game Approach to Digital Archive Management***
  Carlos Maltzahn, Michael Mateas, and Jim Whitehead, University of California, Santa Cruz

Carlos Maltzahn presented an interesting way to tackle the tedious task of digital archiving through gaming, which has been used to increase productivity in the past (e.g., Chao's PSDoom). The main idea of InfoGarden is that it will make document tagging a fun activity through a gaming approach, thereby converting a neglected archive (garden) into a well-maintained one. InfoGarden considers all the documents without a tag as weeds. Once a person starts tagging his documents, the weeds get converted into plants with one or more fruits. As the garden of documents gets cleared up with more plants, the score increases.

- ***RAID4S: Adding SSDs to RAID Arrays***
  Rosie Wacha and Scott A. Brandt, University of California, Santa Cruz; John Bent, Los Alamos National Laboratory; Carlos Maltzahn, University of California, Santa Cruz

Rosie Wacha proposed a technique for improving the performance of RAID 4, which is usually bottlenecked by a common parity disk. She suggested a hybrid RAID 4 that consists of normal hard drives to store data chunks, while faster SSDs would store the parity information. Since SSDs are much faster than hard drives, they could potentially overcome the single-parity disk-performance bottleneck.

- ***Fuse for Windows***
  Mark Cariddi, Tony Mason, Scott Noone, and Peter Viscarola, OSR—Open Systems Resources

Mark Cariddi presented a FUSE implementation on Windows that enables user mode filesystem development on this platform. Mark and his team implemented a user-level service that communicates with the kernel-level library to achieve this goal. The current status is that FUSE is successfully ported to Windows. They have not implemented chown and extended attributes. Some of the difficulties they face result from semantics mismatches between Windows and Linux file systems.

- ***Revisiting I/O Middleware for the Cloud***
  Karthik Kambatla, Naresh Rapolu, Jalaja Padma, Patrick Eugster, and Ananth Grama, Purdue University

Karthik Kambatla explained the problem of how different cloud applications vary in their requirements, such as consistency, availability, or bandwidth, leading to underutilization of resources available to the existing storage systems. He targeted a few cloud applications that need key-value store and proposed to incorporate the key-value (KV) store into the file system. The KV file format consisted of data blocks sorted on the basis of keys. The file also consisted of a data index, with key and data block number pairs, denoting where the information about a certain key was stored. All updates to a file were appended in a log-structured manner.

- ***Determining SLO Violations at Compile Time***
  Kristal Curtis, Peter Bodik, Michael Armbrust, Armando Fox, Michael Franklin, Michael Jordan, and David Patterson, University of California, Berkeley

Kristal Curtis presented this research focused on answering a few questions raised by developers of interactive Web applications: (1) what is the impact of peak workload on a query's latency and will it be within the service level objectives (SLO)? (2) is the new query meeting its SLOs? In this work, Kristal's group tried to estimate the latency of a query by first breaking it into different query operators. They estimated the latency of each query operator based on its latency distribution available from past queries. Later, they combined the individual query operator's latency, giving the total query latency. Based on the total latency, developers can determine whether their queries are meeting the SLOs and take appropriate steps. Since this analysis can be done at compile time, it can save a lot of testing and performance-analysis time.

- ***LazyBase: Freshness, Performance, and Scale***
  Craig A.N. Soules, Kimberly Keeton, and Charles B. Morrey III, Hewlett-Packard Laboratories

Craig Soules said that enterprise data management applications exhibit variations in query performance and result freshness goals. Some applications, such as Web search, require interactive performance but can operate on stale data. Others might require much up-to-date data but not have any stringent performance criteria. LazyBase is a system that allows users to trade off query performance and result freshness in order to satisfy the full range of user goals. LazyBase breaks up data ingestion into a pipeline of operations to minimize ingest time, and uses models of processing and query performance to execute user queries. Craig said that this system is ready but they are still investigating things such as: (1) how do applications specify freshness requirements? (2) how does LazyBase convert freshness to query, security, privacy, etc.?

- ***Study on Performance of Energy-efficient High-speed Tiered-Storage System***
  Hirotoshi Akaike, Hitachi Ltd; Kazuhisa Fujimoto, Naoya Okada, Kenji Miura, and Hiroaki Muraoka, Tohoku University

Kazuhisa Fujimoto proposed eHiTs, an energy-efficient, high-speed tiered-storage system that minimizes performance loss. eHiTs' first tier consists of high-speed online storage, with low-powered nearline storage as the second tier. The main idea behind eHiTs is to conserve energy by minimizing online storage capacity and powering off the HDD enclosures of the nearline storage when not needed. When an HPC job is submitted to eHiTs, it copies the required files (as specified in the job script) from the nearline to online storage, and powers off the nearline disks until the results are ready. The results and the original data are copied back after the results are available and the nearline disks are put to sleep again.

KEYNOTE ADDRESS

- ***Enterprise Analytics on Demand***
  Oliver Ratzesberger, eBay, Inc.

Summarized by Daniel Rosenthal (danielr@cs.ucsc.edu)

Oliver Ratzesberger gave Thursday's keynote address, lending unique insight into the challenges of providing analytics at the scale of a large organization. Today, eBay runs multiple highly available, high-throughput datacenters, each of which has storage capacity exceeding 10PB. Employees are allowed to run advanced SQL queries, which support sophisticated analyses such as time-series computations, as well as custom C and Java code. This analytics infrastructure is provided indiscriminately to all employees at eBay, enabling analysis of any data to which they have access permission. This is a view of what eBay's infrastructure is in its current form; however, it was not always this way. Ratzesberger gave a brief history of the evolution of analytics at eBay and how it overcame various challenges. He also noted that many other companies, particularly smaller ones, face the same challenges today that eBay had to overcome.

Analytics began at eBay with various departments requesting access to customer or user data. Marketing might have been interested in customer purchasing behavior, whereas the Web design team might have been interested in customer click-through data. Each department would then purchase its own database, housing and maintaining that database with the rest of the departmental IT infrastructure. During his presentation, Ratzesberger described these per-department databases as "data marts." Setting up a data mart took time and was not amenable to change once deployed, and each data mart was paid for by the respective department requiring analytics. Eventually, as data marts became more and more pervasive throughout eBay, they began to contain duplicate information and would need to be synchronized with one another periodically to maintain currency and consistency. All the while, eBay was running centralized computing infrastructure as part of its normal operations.

Eventually, around 2003–2004, one marketing data mart reached a size of 15TB, and eBay decided to try and consolidate that data mart with its centralized infrastructure. Ratzesberger recounts that the marketing data mart was eventually merged into the centralized infrastructure, and they were astounded to find out that the merge resulted in only 350GB of net new data on the centralized infrastructure. eBay had previously thought there might be 30% waste in the system, but there turned out to be orders of magnitude waste. Furthermore, the marketing data mart was returning different answers to queries than the centralized infrastructure was, because the two were not synchronized. This resulted in wrong answers to some queries that the marketing department had been running. In addition to technical issues, data marts don't show up on a single budget item; rather, they are cost-distributed, and the company never realizes how much it is spending on them.

eBay eventually consolidated all of its data marts into its centralized infrastructure using virtualization, but virtualization at a much higher level than is typically associated with the word. eBay introduced virtual data marts, which can easily be created and deleted on demand, and provided analytics as a service (AaaS) into its compute and storage infrastructure. Users are provided with a dashboard that allows creation of data marts up to 100GB in size without requiring special permissions. The virtual data mart has an associated expiration date, defaulting to a few months, but it can be renewed indefinitely. With virtual data marts, queries dynamically increase or decrease their computational resource usage depending on system load and job priority, thus attaining a major benefit of virtualization. Another benefit of virtualization is that it allows for completely automated management of data marts. This reduced the number of full-time employees at eBay required for data mart management from over 40 down to 2–4.

In closing, Ratzesberger reiterated that analytics should be dynamic and allow agile prototyping, and should allow users to fail fast when trying out new ideas. Most metrics have a hype cycle, with ROI peaking when first discovered, but eventually going into sustainment; the undiscovered metrics have the highest potential ROI. Analytics as a service reduces time to market and eliminates stray physical data marts located throughout a company.

Rik Farrow from USENIX and another audience member both asked about eBay's high utilization. Farrow pointed out that running at 100% utilization causes intermittent thrashing (since live loads are bursty), while the other audience member cited a result from queuing theory that states that at high utilization, latency grows without bound. Ratzesberger replied that, while high memory utilization causes thrashing, high CPU utilization does not. Setting SLAs for gating efficiency (runtime over runtime plus queuing time) can be used to prioritize tasks and appropriately distribute latency. Another audience member asked about how damage to data is prevented, since jobs share data. Ratzesberger replied that virtual data marts are fenced off from each other, and production tables are (mostly) read-only. The same audience member then asked if most queries scan all data, since so little indexing is used. Ratzesberger responded affirmatively, but noted that indexes are not the only way to improve performance. Data is distributed using a virtual hash, and the system supports multi-dimensional partitioning of data (e.g., time and customer segment as two dimensions). Many MPP systems, such as Teradata used by eBay, support hash-based joins, which also helps avoid sequential scans.

David Chambliss from IBM asked why Ratzesberger avoided the word "cloud" during the presentation. Ratzesberger commented that cloud computing is inadequate for I/O-intensive workloads on petabytes of data, where processing must be moved to the data rather than vice versa. Another audience member asked about potential uses for SSDs. Ratzesberger replied that, since current busses are not designed to handle the 600–800MB/s sustained data rate of SSDs, replacing hard drives with SSDs simply shifts the bottleneck to the bus. Amandeep Khurana from the University of California, Santa Cruz, asked how the underlying data is stored (e.g., in relational databases). Ratzesberger replied that more than 90% of the data is in relational databases, but with hash-based tables that are very simple in structure and can contain semi-structured data (e.g., XML, JSON, and name-value pairs). Only very frequently accessed data is laid out into columns. Ratzesberger also commented that the majority of data analysis is done in SQL and that user-defined functions are also allowed. Randal Burns from Johns Hopkins University asked about using relational interfaces to scan engines or object storage as a cost-saving opportunity over relational databases. Ratzesberger replied that, while eBay does use very expensive storage in the form of 15,000 RPM drives, the storage is also very performant, and it is difficult for lower-cost hardware to achieve the same performance in an equally cost-effective manner. Vidya Sakar from Sun Microsystems asked how eBay provides data reliability. Ratzesberger replied that eBay uses hardware RAID levels as well as software replication. For example, some data blocks are stored in two different cliques of processing nodes, preserving data even in the event of an entire clique failure.

FLASH: SAVIOR OF THE UNIVERSE?

Summarized by Lianghong Xu (lianghon@andrew.cmu.edu)

- ***DFS: A File System for Virtualized Flash Storage***
  William K. Josephson and Lars A. Bongo, Princeton University; David Flynn, Fusion-io; Kai Li, Princeton University

William Josephson presented the design and implementation of a flash-based file system, the Direct File System

(DFS). The primary feature of DFS is that it uses a new abstraction level between file system and the underlying device, which provides a large virtualized address space and reduces the redundancy of block allocation in traditional systems. Traditional file systems are designed for disks. A lot of work has been done for filesystem layout to deal with expensive disk positioning time, which isn't an issue in flash. Therefore, the good random I/O of flash isn't reflected in these systems. Also, traditional file systems designed for disks use a very complex block allocator, which seems to be redundant with flash, because flash devices themselves already embed this functionality inside the flash translation layer (FTL). DFS removes this extra level of indirection and increases the transparency of the system. Previous flash file systems were initially designed for embedded applications instead of high-performance applications and are not generally suitable for use with the current generation of high-performance flash devices.

The key design idea of DFS is to switch from the traditional block storage layer to a virtualized flash storage layer that sits in the device driver and embeds features like remapping, wear leveling, and reliability, while not disturbing the user interface. One drawback of this design is that the device driver may consume much CPU and memory resources. There are four requirements for DFS: a large virtualized address space, crash recoverability, the ability to atomically update one or more blocks, and an interface to the garbage collector for deallocating a range of blocks. According to Jefferson, these are not very strong assumptions and the trend already exists in some flash systems. In DFS, each filesystem object is assigned a contiguous range of logical block addresses. Large files and small files are distinguished and are assigned different allocation chunks. This approach enjoys its simplicity but may suffer from the waste of virtual address space.

One person asked how much memory would be consumed by the device driver in DFS. David Flynn answered that this is actually highly dependent on the write extent size. Someone from Sun raised the issue of considering ZFS as an option in their research. Jefferson answered that the Fusion-io device required driver support, and it didn't support Sun at the time they started the project. A person from NetApp pointed out that coalescing two allocators into one may lose some information and asked about the possibility of making this work better by using an object disk. Jefferson said he wasn't sure about whether this could make this work better but he thought an object-interface might be the right way to go in the long term.

- ***Extending SSD Lifetimes with Disk-Based Write Caches***
  Gokul Soundararajan, University of Toronto; Vijayan Prabhakaran, Mahesh Balakrishnan, and Ted Wobber, Microsoft Research Silicon Valley

Gokul Soundararajan showed how to use a disk as write cache for a solid-state device (SSD) to extend its lifetime. SSD has desirable features such as fast reads, low power consumption, and high reliability, but it suffers from limited write times because of the notorious "write amplification" problem. As a result, the practical write-lifetime for SSD is much less than ideal. In their approach to extending SSD lifetimes, the authors chose to add a level of indirection—adding a disk as the write cache for SSD. While it may seem odd at first glance, this decision was made out after comparison with other alternatives and serious consideration. Disks can provide comparable sequential write speed to SSDs and much higher capacity per dollar, while other choices are not suitable, for various reasons.

The authors built a Griffin hybrid device to mitigate the large volume of writes to SSDs. The Griffin hybrid device is composed of three parts: an SSD as the final destination of all the persistent data, a disk (log-structured) to cache write blocks, and a hybrid device controller which maintains the mapping of block numbers to log offsets. The basic idea is to sequentially cache all block writes to the disk log instead of writing them directly to the SSD. Then, at an appropriate time, the blocks in the disk are migrated to the flash. For every read request, the mapping table in the hybrid device controller performs a lookup to decide whether the request should go to the disk or the SSD.

In order to enhance the performance of the basic algorithm, the authors examined various I/O traces in the real world, including the I/O workload from desktops, servers, and Linux machines. The result shows that a large fraction of overwrites appear in the workload and that overwrites happen quickly while reads occur after a long interval. Based on these observations, the authors made several trade-offs between the write savings and read penalty introduced by the use of disk. Accordingly, several schemes are proposed as potential enhancement, such as selective caching and hybrid migration trigger. The evaluation shows that disk-based write cache improves SSD lifetime by a factor of two and reduces average I/O latency by 56%.

Someone from Stony Brook University asked why not use the disk as a journal and simply write all the data to the flash to reduce the overhead of migration. Soundararajan argued that this would require understanding the mechanisms of file systems' buffer cache; their device sits in the block I/O level and doesn't distinguish between eviction writes and consistency writes. Someone from NetApp asked if it is possible to use the trace data to estimate the real lifetime of current drives. Soundararajan answered that it is hard to evaluate because some of the data is collected from hard drive traces. A person from Northeastern University raised the concern about the memory overhead required to do the migration. Soundararajan said their approach only migrates 16 or 32MB of data at a time and thus doesn't incur much memory consumption.

- ***Write Endurance in Flash Drives: Measurements and Analysis***
  Simona Boboila and Peter Desnoyers, Northeastern University

Simona Boboila presented her study on write endurance of USB flash drives using reverse engineering, timing analysis, and whole-device testing. The migration of USB flash drives from mobile devices to desktop devices raised people's concern about USB flash drives' write endurance. The choice of targeting USB flash drives in this work is largely due to their simplicity, while device disassembling, destructive testing, and reverse engineering are more difficult to do for more sophisticated devices.

Device lifespan can be predicted from chip-level endurance and the internal algorithms implemented in the flash translation level (FTL). The internal algorithms can be obtained using techniques in reverse engineering. Specifically, the authors studied block update mechanisms with different complexity in three devices: a generic device, a House device, and a Memorex device. The lifespans of these devices are predicted as a function of chip-level endurance and the internal algorithms, which are pretty close to the measured result. Another technique, timing analysis, can be employed to determine whether the device is approaching its end of life. Specifically, the authors reveal that at 25,000 operations before the end, all operations slow down to 40ms. Based on the observation that the same update block is used when writes are issued to the same data block, the authors also propose a scheduling scheme to reduce garbage collection overhead and improve performance. The authors expect their results to apply to most removable devices and low-end SSDs with little free space and RAM.

Someone asked if the implementation of the scheduling mechanism should be integrated into file systems or the flash itself. Boboila said this needs further consideration but flash-based file systems are probably a good choice. Another person was curious about the extensibility of these techniques to more high-end and complex SSDs. Boboila answered that their approaches are good for low-end devices because they use simple internal algorithms and have few free blocks, but they may not be suitable for high-end devices, due to the complexity of the algorithms implemented. Another person asked if the authors had looked into the power consumption of these USB drives. Simona said no.

I/O, I/O, TO PARALLEL I/O WE GO

Summarized by Daniel Rosenthal (danielr@cs.ucsc.edu)

- ***Accelerating Parallel Analysis of Scientific Simulation Data via Zazen***
  Tiankai Tu, Charles A. Rendleman, Patrick J. Miller, Federico Sacerdoti, and Ron O. Dror, D.E. Shaw Research; David E. Shaw, D.E. Shaw Research and Columbia University

Tiankai Tu presented a technique for caching scientific simulation data on analysis nodes as it is generated in order to speed up the overall process of data analysis. The authors focus on analysis of molecular dynamics (MD) simulations, which occur over micro- or millisecond time scales and generate tens or hundreds of millions of discrete-time output frames. Each simulation consists of two potentially overlapping phases: simulation, which outputs generated data to a central file system, and analysis, which reads and analyzes simulation output data from the central file system. MD simulation output frames have a strong inter-frame dependence, causing data analysis to be tightly coupled with data retrieval. This coupling, in combination with the use of a central file system, creates an I/O bottleneck. The authors overcome this bottleneck by, at simulation time, proactively pushing frames from simulation nodes into both the central file system and the local caches of analysis nodes as the frames are generated. Then, at analysis time, the analysis nodes simply fetch data in parallel from their local caches. Any frames not in an analysis node's local cache can be fetched on-demand from the central file server.

The authors implemented their solution in a distributed consensus protocol called Zazen. The Zazen protocol ensures that no frame is analyzed more than once. Zazen requires each analysis node to generate a bitmap of its locally cached frames. These bitmaps are then exchanged between analysis nodes using an all-to-all reduction algorithm. In the course of exchanging these bitmaps, each node determines which frames it is responsible for analyzing. One key feature of Zazen is that it does not require any central servers for coordination. The authors showed that Zazen reduces file read time by nearly two orders of magnitude compared to reading data over NFS from central filesystem servers. Tu concluded by noting that Zazen only works for a certain class of time-dependent simulations, but the authors believe Zazen may be applicable to analysis of any data with a total ordering.

Craig Soules from HP Labs asked if these techniques could be applied to HDFS. Tu replied that this is an orthogonal design choice because HDFS has a central metadata server, whereas Zazen does not use centralized coordination.

- ***Efficient Object Storage Journaling in a Distributed Parallel File System***
  Sarp Oral, Feiyi Wang, David Dillow, Galen Shipman, and Ross Miller, National Center for Computational Sciences at Oak Ridge National Laboratory; Oleg Drokin, Lustre Center of Excellence at Oak Ridge National Laboratory and Sun Microsystems Inc.

Sarp Oral presented a software technique for improving journaling performance in a supercomputing environment by providing asynchronous commits of metadata. This technique is in production use on the Spider storage system, which provides storage to the Jaguar XT5 supercomputer (currently number one on the TOP500 list of supercomputers). Spider runs the Lustre file system. The authors' work was motivated by an observed factor of four difference in raw-block throughput and filesystem throughput on Spider, caused by seeks attributed to filesystem journaling. Lustre nodes run ldiskfs locally, which is a journaling file system similar to ext3. The default journaling mode for ldiskfs is ordered mode, in which only metadata blocks are journaled, but all data blocks are written to their final locations on disk before any associated metadata blocks are committed to the journal. The size of the journal is limited to one-quarter of the disk capacity, and clients with outstanding RPC requests block on the server whenever the journal commits. Blocking clients with outstanding RPC requests prevents those clients from issuing further requests until their current request completes, creating a point of serialization.

The authors' solution is to synchronously write data but asynchronously journal metadata. This lowers the number of seeks that are required for journal commits and provides the possibility of coalescing metadata updates. The authors show this solution to be superior to hardware solutions and demonstrate that it allows ldiskfs to achieve 93.1% of the raw-block throughput, as opposed to 24.9% of the raw-block throughput measured with synchronous journaling.

Craig Soules from HP Labs asked if performance could be improved further by delaying journal commits even longer. Oral responded by noting that the journals reach their capacity very quickly due to the volume of data being generated, so delaying further might be of little benefit, since the journal must eventually be flushed anyway. Additionally, a longer delay requires replaying more data when recovering from a failure.

- ***Panache: A Parallel File System Cache for Global File Access***

  Marc Eshel, Roger Haskin, Dean Hildebrand, Manoj Naik, Frank Schmuck, and Renu Tewari, IBM Almaden Research

Renu Tewari presented a technique for cluster-to-cluster caching in a wide-area file system. Tewari opened by stating that often data sources (e.g., radar and satellites), datacenters, and clients are geographically disparate, resulting in unreliable network connections with high latency but reasonable bandwidth. The authors attempt to leverage these characteristics to provide global data access at local speeds through the use of caching. Their solution is Panache, which is a caching system built on top of GPFS. It assumes a storage model consisting of a "home" cluster, which holds the authoritative copy of a file system, and one or more cache clusters, which service clients (applications or users) requiring access to files stored on the home cluster. Panache caches files when they are first read in order to reduce the latency of future accesses. It also uses delayed write-back to reduce write latency and supports disconnected operation in the event of network failures (latent update conflicts are handled as described in Coda). Panache performs all data transfer operations in parallel across multiple nodes using pNFS, making full use of available inter-cluster bandwidth.

Sometimes dependencies arise between metadata operations, which can be problematic when using delayed write-back at a cache cluster. For example, a directory might be created, and a file subsequently created within that directory. To maintain consistency, the cache cluster must flush these operations back to the home cluster in the proper order. However, since Panache operates in parallel, there might be multiple queues of pending updates waiting to be flushed to the home cluster, potentially causing dependent operations to be permuted. Panache handles this by tracking all dependencies and enforcing a write-back order consistent with the dependencies.

Benny Halevy from Panasas asked whether file handles are stored locally or remotely, and how stale file handles are detected. Tewari responded that file handles are stored remotely, and so standard NFS error codes will be returned if a stale file handle is used.

MAKING MANAGEMENT MORE MANAGEABLE

Summarized by Sriram Subramanian (srirams@cs.wisc.edu)

- ***BASIL: Automated IO Load Balancing Across Storage Devices***

  Ajay Gulati, Chethan Kumar, and Irfan Ahmad, VMware, Inc.; Karan Kumar, Carnegie Mellon University

Ajay Gulati presented BASIL, a system for automated I/O load balancing across a group of storage devices. Storage is one of the most expensive components of any datacenter, and management of these resources presents a tremendous opportunity to avoid unnecessary investment in storage when careful utilization of resources could solve the problem. Live migration of VM hosts has been a popular technique to alleviate CPU and memory overloading. This technique doesn't directly help manage a similar situation in storage arrays. Storage bottlenecks have to be manually identified. This problem is very complex,involving identification of the problem-causing storage device, of the virtual disks that are causing the problem, and, most importantly, of the destination machine (but without causing the same problem in that machine).

The two key aspects of storage management are I/O load balancing and virtual-disk placement. The novel aspects of this work include using latency as the main metric for characterizing workload. Latency and throughput are both reflective of the workload being run—throughput tends to saturate with increasing load, but latency doesn't. Latency is more sensitive to load and tends to have linear properties, making modeling easier. For example, latency tends to linearly increase with metrics such as outstanding I/O, I/O size, and read-write ratio. Randomness present in the I/O provides an interesting point in the study—linearity breaks down with fully sequential workloads. These factors are then empirically combined to produce a workload model. Device modeling has its own set of challenges, as the performance of the underlying devices tends to vary widely and all the lower-level information is abstracted away from the hosts. The models developed here allow both load balancing and initial placement through use of a normalized load metric. The evaluation of BASIL showed around 25% improvement in IOPS over random migration and 53% improvement in IOPS over random initial placement. It also showed promising results in studies that involved experts who were asked to perform the same operations as BASIL using the same information that was available to BASIL.

Kaladhar Voruganti (NetApp) asked if BASIL would work against the block-level migration policies of storage vendors. Gulati felt that this wouldn't be an issue as long as the granularity of these operations were vastly different—block-level migration tends to be much shorter, on the order of minutes, but BASIL works in the granularity of days or weeks. If the granularity is very similar, then turning off one of these would be better. Someone asked about write flushes in practice and how BASIL models would react to write flushes. Gulati mentioned that they haven't seen too many in practice. However, frequent write flushes should get reflected in the device model. Also, as was stated before, the model beaks down for write-heavy workloads. It was part of their future work to extend their model to work better with write workloads.

- ***Discovery of Application Workloads from Network File Traces***
  Neeraja J. Yadwadkar, Chiranjib Bhattacharyya, and K. Gopinath, Indian Institute of Science; Thirumale Niranjan and Sai Susarla, NetApp Advanced Technology Group

The paper described the work on identifying application workloads from NFS traces. Knowledge of the application can be useful in provenance mining, anomaly detection, enabling autonomous systems, etc. The goals of this work include (1) identifying workloads, (2) identifying transition workloads in a trace sequence, (3) identifying incomplete or partial traces, (4) identifying concurrent applications at the same client, and (5) identifying variability of operations—the same cp command with a minor variation in the options can produce a vastly different trace. This means that a straightforward comparison of traces won't suffice.

Yadwadkar went on to explain the significance of global and local alignment and how they pertain to the goals of this work. Global alignment is the process of arranging the sequences so as to maximize the similarity between the sequences. Local alignment allows sub-sequence matches, thereby making the transition study possible. The key insight here is that there exists a similarity between the computational and the biological problem of matching gene sequences, and profile HMMs (hidden Markov models) have been successfully employed to achieve good results. The model was evaluated using traces from both commonly used UNIX commands (cp, tar, grep, etc.) and larger workloads such as TPC/C and Postmark. It was also shown that just about 20% of the trace is required to make a reasonable match.

Margo Seltzer, Harvard University, asked how the server could retain more information, thus providing vital information typically not available in the trace. Irfan Ahmed (VMware) wondered about extending the work to identify the queries that make up the workload in TPC/C and also to use a similar technique to block traces. Yadwadkar said both suggestions would be looked into as part of future work.

- ***Provenance for the Cloud***
  Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer, Harvard School of Engineering and Applied Sciences

Provenance is the metadata that describes the operations that were performed on the data. They are typically described as a DAG that shows the dependency relationship between various entities that process that piece of data. Muniswamy-Reddy presented protocols for storing provenance for cloud-based applications. Cloud stores have gained significant traction since the introduction of Amazon S3 and Microsoft Azure. Cloud stores are used in a wide variety of domains: for backup, to share scientific data and Web application data, and also for Web pages themselves for hosted environments. The nature of cloud stores makes the process of storing provenance non-trivial, and the latency and cost constraints make the trade-offs very challenging. Provenance is important as it allows users to validate data sets, identify how data spread through the system, and improve the quality of data search. Cloud storage providers can also trade off between storage and computation by generating rarely used data on demand as opposed to storing it all the time if they had the provenance available to them. The work builds upon the Provenance Aware Storage System (PASS), which could originally handle local and network file systems.

The key properties that make provenance truly useful are (1) provenance data coupling—provenance should accurately describe the data; (2) multi-object ordering, which extends the coupling property all through the provenance chain; (3) data-independent persistence—provenance should be retained even after the data it describes has been deleted; and (4) efficient query—provenance needs to be accessed efficiently. Muniswamy-Reddy went on to describe three protocols of increasing strength and complexity that guarantee varying levels of these properties described above. These protocols ensure persistence and causal ordering, but data coupling and efficient queries are not available in all of them. These protocols are (1) stand-alone cloud store, (2) cloud store and cloud database (to allow efficient queries), and (3) cloud store with cloud database and messaging to ensure data coupling and efficient queries. The evaluation of the system compared the baseline time of Amazon S3fs with these systems using the Blast microbenchmark, nightly backup benchmark, and provenance challenge benchmark. The most interesting result was that Protocol 3 delivered the best performance. Thus good performance and strong provenance are not always opposing functions.

Session chair Kaladhar Voruganti (NetApp) asked how a clean slate approach would re-architect the provenance system. Muniswamy-Reddy thought that a new approach would require better APIs from the cloud vendor as well as transactional support. He also felt that storing DAGs in cloud-db may not be ideal, and so a storage system that is better suited to storing graphs would be worth exploring.

CONCENTRATION: THE DEDUPLICATION GAME

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

- ***I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance***
  Ricardo Koller and Raju Rangaswami, Florida International University

Ricardo Koller introduced I/O Deduplication, a way to improve storage efficiency by using content similarity to improve I/O performance. It comprises three techniques: content-based caching, dynamic replica retrieval, and selective duplication. This approach was shown to improve performance 28–47% across a set of three workloads.

Each of their techniques was built around the observation that it is possible, and in some cases common, to have data at different sector addresses that share the same content. While this is the same observation that leads to traditional deduplication efforts, I/O Deduplication differs in that it focuses on optimizing the cache and I/O performance when accessing duplicate data.

In content-based caching, the authors observe that identical data may be included multiple times in the buffer cache. To address this, they created a secondary cache underneath the page cache that services blocks of data based on content. In dynamic replica retrieval, the authors note that when data is requested for a read, it could be served from any location that holds identical data. By predicting the location of the disk head based on previous I/O operations, the authors argue, one can select alternate addresses that minimize read I/O latency. Finally, in selective duplication, data is intentionally replicated across the disk to improve locality.

In closing, Koller detailed their impressive performance results and pointed out areas of future work. He proposed integrating I/O deduplication with the page cache and the I/O scheduler. He also proposed extending the technique to work with multiple disks and variable-sized blocks.

Mark Lillibridge from HP Labs noted that the whisker plots showing performance improvement had overlapping confidence intervals and asked if any conclusions could really be drawn about performance. Koller pointed to the average performance, which was improved in every instance, though not by more than the confidence interval. Margo Seltzer of Harvard University wondered how much memory would be required to achieve the same performance benefit by simply expanding the size of the buffer cache. Finally, Bill Bolosky of Microsoft Research stressed that since MD5 is insecure, one likely needs a more costly algorithm to ensure that users receive the correct data in the presence of an attacker. He asked how SHA-1 would change the performance of the system. Koller wasn't certain, but acknowledged that using the more costly SHA-1 algorithm would have a performance impact.

- ***HydraFS: A High-Throughput File System for the HYDRAstor Content-Addressable Storage System***
  Cristian Ungureanu, NEC Laboratories America; Benjamin Atkin, Google; Akshat Aranya, Salil Gokhale, and Stephen Rago, NEC Laboratories America; Grzegorz Całkowski, VMware; Cezary Dubnicki, 9LivesData, LLC; Aniruddha Bohra, Akamai

Cristian Ungureanu detailed NEC's file system, which addresses the need for scale-out failure-resistant storage that features global deduplication, high throughput, and ease of management. Their system, HydraFS, is built as a layer on top of HYDRAstor, the content-addressable (CA) store presented at FAST '09. Content-addressing schemes, while popular, do not expose an API that is compatible with conventional applications and file systems. HydraFS addresses this issue by providing a traditional filesystem interface that can sit atop a CA data store, while maintaining filesystem consistency and optimizing throughput.

When designing the system, Ungureanu and his team faced several challenges. In a content-addressed system, modifying a block changes both the data and its address. This necessitates updating on-disk references to the modified block, and doing so in a way that leaves the file system in a consistent state. Content addressing also implies a higher degree of request latency, because chunking, hashing, erasure coding, and compression each add delay to request processing. The presentation touched on many interesting aspects of the system, starting with the overall architecture and continuing into the details of the I/O path and the admissions policies that ensure that enough resources are available for each request.

As in other file systems, the write path begins with a buffer that accumulates write requests and is flushed on sync requests. These requests are passed to a chunker, which decides where block boundaries should be placed. Like many other systems, these boundaries are based on data content, so inserting into the middle of a file will not change chunking decisions far from that location. The file server places the file system's metadata updates in a log of modification records. The log is read by the commit server, which periodically translates log entries into the appropriate filesystem tree updates. Each time it does this, a new on-disk consistency point is created. At each new consistency point, the file server has the opportunity to clean its in-memory cache by removing entries that have since been updated. This process is carefully constructed to avoid large-scale or long-held locks, and need not consult the filesystem tree to perform its function. On the read path, prefetching and a buffer cache are used to mitigate the performance impacts of the deep filesystem metadata tree. Both prefetching and eviction policy are designed to favor filesystem metadata, as metadata cache misses are particularly costly.

In the future, the authors plan to enhance HydraFS2 to operate as a cluster-wide distributed file system. They also plan to incorporate solid state disks into their designs.

- ***Bimodal Content Defined Chunking for Backup Streams***
  Erik Kruus and Cristian Ungureanu, NEC Laboratories America; Cezary Dubnicki, 9LivesData, LLC

Erik Kruus of NEC began his presentation by asking that the audience join him in considering new ways to improve on his simple but effective technique for content-defined chunk selection in backup streams. In his talk he posed the question, "What other approaches are out there, just waiting for brilliant minds to figure them out?"

Chunk selection refers to the process of taking a large file and breaking it up into smaller units of storage. These boundaries are chosen with the hope that chunks with the same data can be found in other files and duplicate data can be eliminated. To improve the deduplication rate, chunk selection is often done by evaluating a sliding window over the data. This produces variably sized chunks, where the boundaries are determined by the data itself. This content-defined chunking process increases the chances that identical sub-sequences will be deduplicated, even if they appear at different offsets or in a different file.

Selecting large chunk sizes offers lower storage overheads and better I/O performance, but smaller chunk sizes offer higher deduplication rates. Kruus's observation is that between two backups many small changes are likely to occur. Around the areas that have been modified, small chunk sizes can best capture the differences. The rest of the data is unchanged, so is better represented with large chunk size. To facilitate this approach an interface was added that allows one to check for the existence of a particular chunk in the data store. When larger chunks are not already present, smaller chunks may be considered. Kruus showed the effects of incorporating compression and the merging and splitting of chunks in an attempt to reach the highest levels of deduplication possible. He also warned that as features are introduced they increase metadata overheads, which degrades the overall performance of the system. He cited Occam's razor in explaining that even though he had an impulse to try complicated algorithms, simpler approaches usually prevailed.

In the question period, Kaushik Veeraraghavan from the University of Michigan suggested taking quick initial action to speed requests along the I/O path, then revisiting simple chunking decisions later with more complicated algorithms. Randal Burns from Johns Hopkins University drew a parallel between this work and the differencing seen in network literature, and wondered why, in the former, some workloads seemed to perform poorly. Kruus replied that without a reasonably high duplication rate (cutting the original data down to a third or more) the approach didn't have enough opportunities to provide benefit.

THE POWER BUTTON

Summarized by Sriram Subramanian (srirams@cs.wisc.edu)

- ***Evaluating Performance and Energy in File System Server Workloads***
  Priya Sehgal, Vasily Tarasov, and Erez Zadok, Stony Brook University

Power consumption is one of the biggest operating expenses of a datacenter. For every $1 spent on hardware, 50 cents are spent on just power. This clearly shows the importance of power management, and the first step towards better understanding this is to benchmark file systems' power consumption and performance. To make such benchmarking systematic is the most important contribution of this paper. Two of the commonly used techniques to reduce power are (1) right sizing—reducing the number of active components results in significant power reductions (CPU DVFS is a popular technique); (b) work reduction—in essence, reducing the workload but doing the same amount of work.

The experimental methodology involved varying the workload, file system, and hardware. The workloads employed were Web server, database, file server, and mail server (all through FileBench). These workloads have different I/O sizes, directory depth, average total number of files, read-write ratio, etc. The file systems studied (ext2, ext3, XFS, and ReiserFS) also had varied allocation block sizes, journaling modes, extents, indexing structures, etc. Finally, two widely different hardware configurations were also studied and their performance-power consumption analyzed. One significant observation was that performance closely followed energy consumption in almost all the configurations. In the first machine configuration and under the mail server workload, ReiserFS performed the best; ext2 suffered from fsync bottleneck, and XFS from a lookup-related bottleneck. Another interesting observation was that ReiserFS with no tail-packing produces a 29% improvement in performance due to avoidance of tree rebalancing.

Sudhanva Gurumurthy, University of Virginia, asked what were the other parts of the system stack that can be reconfigured. Priya suggested DVFS as a probable option. Also the system admin can provide useful hints that can make this process better. Why is fsync not a bottleneck in machine 2? Machine 2 has a faster disk and a much larger disk cache. Jason Flinn, University of Michigan, asked whether performance is a good indicator of power consumption; can performance be used as a metric to reduce energy costs? Priya felt that the current experiments were carried out at peak loads, so reduced loads might give different results. Irfan Ahmed, VMware, asked if this research work can be used to create a comprehensive power model. Priya felt that it would require more in-depth study and would be part of their future work.

- ***SRCMap: Energy Proportional Storage Using Dynamic Consolidation***
  Akshat Verma, IBM Research, India; Ricardo Koller, Luis Useche, and Raju Rangaswami, Florida International University

Luis Useche presented SRCMap, which attempts to make storage energy proportional through the use of dynamic consolidation in order to deal with datacenters' growing energy requirements, which are increasing at 15–20% every year. Storage forms a significant portion of the energy requiremen,t at almost 10% to 25%. The problem with storage devices is that even at low load, they consume peak power. Other parts such as the CPU can be powered down, making them energy-proportional. Thus the solution proposed in this work is to consolidate workloads in a few storage devices to make them energy-proportional. The most important challenge here is to keep the cost of migrating logical volumes to a minimum while still achieving the benefits of consolidation.

The three key observations that drive their solutions are: (1) active data sets only form a small portion of the total storage, (2) there is significant variability in the I/O load, and (3) over 99% of the working set consists of either popular or recently accessed data. This implies that there is significant opportunity to offload volumes due to load variability, and the stability of working sets means that the operation of synchronizing replicas of a logical volume is relatively rare.

For each logical volume, SRCMap evaluates the working sets and replicates them on scratch space available in all physical volumes. Once this is done, the short-term workload is characterized and SRCMap figures out which subset of physical volumes and replicas needs to be active for the next few hours. The other disks are spun down, thus saving power. The characterization and replication are done only at initialization, while the consolidation phase is repeated every few hours so as to change the active replicas in response to variations in the observed workload.

SRCMap also provides fine granularity for replication of volumes, a relatively low space overhead in terms of the replicas, workload adaptation, and reliability. The replica placement policy is based on the stability of the working set, the average load, power efficiency of the primary physical volume, and the working set size. Experiments were run on top of eight independent volumes and the time interval for disk spin up/down was set to two hours. The evaluation clearly showed a reduction of power requirement of about 35.5%. Also, SRCMap was shown to be an energy-proportional system.

Someone asked how writes are handled reliably during volume replication. Useche said that the reliable replication is the responsibility of the virtualization manager, and writes to both the content of the working set being replicated and to ones not in the working set were handled while the volume was offloaded. Sankaran Sivathanu (Georgia Institute of Technology) asked about the differences between this work and PARAID. Useche suggested that PARAID was a solution for disk level, and SRCMap targeted volumes that were being managed by the volume manager, thus managing more than just the physical volumes. Also, PARAID and SRCMap are complementary and can be used together. Vidhya Bhusan from Virginia Tech wanted to know about the mechanism for determining the working sets. SRCMap has block-level traces and thus can easily maintain the working set for each of the logical volumes based on the access patterns. Ajay Gulati, VMware, wanted to know if they performed any analysis on SRCMap's ability to handle bursts. Useche acknowledged that I/O bursts could pose a problem and that it was a good direction for future work.

- ***Membrane: Operating System Support for Restartable File Systems***
  Swaminathan Sundararaman, Sriram Subramanian, Abhishek Rajimwale, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Michael M. Swift, University of Wisconsin—Madison

  Awarded Best Paper!

Sundararaman presented Membrane, which describes operating system support to make file systems restartable. Recent research has uncovered a lot of bugs in filesystem code, and this can not only cause significant problems to end users but also prevent adoption of newer file systems. A key insight into this problem is that filesystem developers are paranoid about failures, which is evident from the presence of lots of asserts in the code. But the absence of a generic and correct mechanism of recovery results in all these asserts triggering a kernel panic. Recovery of a crashed file system is hard, as there is a lot of state spread all through the kernel and tracking; cleaning and restoring state is a challenging problem.

The three main components of Membrane are fault anticipation, fault detection, and recovery. Fault anticipation refers to the mechanisms added to prepare the kernel for a crash. In Membrane, the anticipation machinery involves generic copy-on-write-based checkpointing and maintaining operation logs to aid recovery. The fault detection components harden the kernel interfaces by adding additional parameter checks. The recovery machinery involves a cleanup of state on crash, un-mount and remount of the file system, and replay of completed operation. A key aspect of this recovery is handling operations that were in flight at the time of the crash, using the skip-trust unwind protocol. The evaluation of Membrane showed the applicability of these mechanisms to three different file systems and the limited overhead imposed on the kernel by Membrane.

Geoff Kuenning of Harvey Mudd College felt that the whole infrastructure of Membrane was hacky and wanted to know why one would trust the Membrane machinery to handle the reliability of a file system. Sundararaman replied that Membrane should be considered as the last line of defense and that filesystem developers must continue improving the systems. Rick Spillane from Stony Brook University wanted to know how Membrane interacts with a journaling file system. Each journal transaction acts as the checkpoint boundary to ensure through modifications done at the jbd layer. The second question had to do with COW pages creating memory pressure, but Sundararaman felt that pages that are marked COW are copied only when they are touched by other processes, so it is copy on demand and not copy always. Erez Zadok from Stony Brook University asked how Membrane ensures that only filesystem code pages are made non-executable. The file system is compiled as a loadable kernel module, thus letting Membrane mark its code pages easily. Sundararaman said that some compile time modifications could help solve this issue. Jason Flinn, University of Michigan, wanted to know about the frequency of bugs that were both fail-stop and transient. The bug reporting and fixing methodology of Linux kernel is ad hoc and done primarily through mailing lists, which makes the process of identifying real bugs harder.