;login:
OCTOBER 2013 VOL. 38, NO. 5
ELECTRONIC SUPPLEMENT

Conference Reports

HotStorage '13: 5th USENIX Workshop on Hot Topics in File and Storage Technologies

San Jose, CA
June 27-28, 2013

Filesystems Everywhere

Summarized by Lanyue Lu (ll@cs.wisc.edu)

Fault Isolation and Quick Recovery in Isolation File Systems

Lanyue Lu, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin—Madison

Lanyue Lu presented isolation file systems, providing fault isolation and quick recovery within a single file system. Because file systems are important data access interfaces in many environments, high availability is critical; however, a single fault can trigger a large-scale impact for the whole file system, such as remounting as read-only and a system crash. Lanyue explained how a metadata corruption of a virtual machine disk image can affect multiple virtual machines that share the same hypervisor file system. Lanyue argued that modern file systems do not provide fine-grained fault isolation, thus all the files within a file system share a single fault domain.

A study of global failures for three modern file systems (ext3, ext4, and Btrfs) showed that remounting as read-only and crash are common in file systems; all these global failures are caused by metadata and systems states. Lanyue presented a new abstraction, the file pod, to represent an independent fault domain for a group of files and their related metadata. Based on this abstraction, a file system is broken into multiple file pods with their own failure policies. Faulty pods will not affect healthy pods; thus, global failures can be converted to local failures. Failure recovery is speeded up since only faulty pods need to be checked instead of the whole file system. A prototype of the isolation file system on ext3 was also described in aspects of file system layout, data structures, allocation algorithms, and journaling mechanisms.

When asked about how parallel commits are coordinated in isolation file systems, Lanyue responded that each commit thread is only responsible for its own pod, and no coordinators are needed. Ajay Gulati (VMware) asked about the difference between a file pod and a partition. Lanyue replied that a partition is not disk-space efficient; second, there is lots of overhead to manage multiple partitions; finally, a single file system on a partition can crash the operating system, which will affect all partitions. Binny Gill (Nutanix) asked whether pod-related structures will cause global failures. Lanyue responded that pod structures are only local, and there is no sharing or dependency across multiple pods. If any failure is related to a pod structure, it will only affect itself. Haryadi Gunawi (University of Chicago) asked why not improve current file system recovery code to avoid these global failures. Lanyue responded that modern file systems handle failures in an ad hoc manner. Recovering from a transaction failure in the journaling layer is hard because it requires tracking many related states in various places. Instead, isolation file systems do this in a systematic way by isolating the whole I/O path from a system call to the journaling layer; thus, any low-level failure can be isolated, related, and recovered in a cleaner way.

**-Box: Towards Reliability and Consistency in Dropbox-like File Synchronization Services*

Yupu Zhang, Chris Dragga, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau, University of Wisconsin—Madison

Cloud file synchronization services such as Dropbox are popular nowadays. Although users enjoy the benefits of automatic file updates and recovery on multiple devices, data corruption and inconsistent data from local file systems may also propagate to the cloud; thus, all copies of the data will be polluted. Yupu Zhang argued that the main reason why this can happen is a lack of coordination between the local file system and the file synchronization service. With *-Box, Yupu presented several techniques to integrate these two components for better data reliability and consistency.

A series of fault injection experiments were conducted to show that corrupted data gets propagated to the server when the Dropbox client detects changes in the file content or its modification time. This is because the Dropbox client cannot differentiate legitimate changes from corruption. Yupu proposed a possible solution in which data checksumming is applied in local file systems to detect potential data corruption. The local file systems are then able to recover from any detected corruption by using the redundant copies in the cloud. Additionally, crash consistency is also critical for both local file systems and cloud file services; however, if local file systems do not handle crash recovery correctly, Dropbox can synchronize inconsistent data on disk to the cloud after a crash. In some cases, Dropbox even fails to synchronize consistent data on disk. Yupu proposed a way to make Dropbox synchronize files from an in-memory snapshot of local file systems. With this technique, data in the cloud is always consistent with data on disk, so file systems and Dropbox clients can perform correct recovery after a system crash.

Nitin Agrawal (NEC Labs America) asked why should the sync service care about local file system dumping garbage data? Yupu responded that it would be great if local file systems could handle corruption. But because Dropbox files may be inconsistent upon a crash, extra care is required to make files in the cloud actually match the files on disk. When asked about an alternative to using a thin client file system provided by Dropbox or other sync service, Yupu responded that if the sync services used a user-level file system, it would definitely affect performance. Otherwise, loading an extra kernel module would be required. Binny Gill (Nutanix) asked about using checksums provided in some hard disks. Yupu replied that most users of Dropbox still use commodity hard drives.

Mobile Data Sync in a Blink

Nitin Agrawal, Akshat Aranya, and Cristian Ungureanu, NEC Labs America

Nitin Agrawal presented Simba, a datacentric platform for developers to build and deploy mobile services with little effort. Modern mobile applications rely on both local storage and remote cloud services for data store and access. Under this datacentric model, developers must handle the data synchronization, network connections, and resource usage explicitly. Moreover, developers have to manage structured and unstructured data with two different models (tables and objects), and carefully handle the dependency between them. Without opportunistic data coalescing and compression, applications built on this model may not use network resources efficiently.

Simba provides a unified table and object API for mobile applications. This API supports simultaneous updates of object data and associated table data; thus, it provides a single namespace for all data. The Simba Client Data Store is responsible for storing both tables and objects; it splits objects into smaller chunks and detects sub-object changes for fine-grained synchronization. The minimum unit of atomic synchronization in Simba is a single row of a table, which is a stronger guarantee than other existing alternatives. Applications may also specify different sync policies for different requirements. The Network Manager in Simba handles network connectivity transparently and consumes network resources in a frugal manner. With the help of Simba, mobile applications will be greatly simplified.

When asked about the performance and reliability of the single process of Simba, Nitin responded that they did not observe much overhead for tables; but for large objects and frequent updates, there will be some penalty. Someone asked why not just export the Simba API. Nitin replied that running Simba as a service can exploit the optimization across multiple apps on the same device, such as consolidation of data transfer, better compression, and even sharing a single TCP connection. Binny Gill (Nutanix) asked about the difference between Simba and iCloud/Dropbox services. Nitin responded that Simba was not developed for mobile apps only. Instead, Simba provides a unified data model for both table and objects, and it also exploits benefits across multiple apps.

Everything About NAND

Summarized by Ilari Shafer (ishafer@cs.cmu.edu)

Improving NAND Endurance by Dynamic Program and Erase Scaling

Jaeyong Jeong, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim, Seoul National University

Jaeyong Jeong opened the talk with the observation that, although NAND flash capacity doubles every two years, and its cost-per-bit is continually decreasing, device lifetime is not keeping pace. Lifetime was defined as the quotient of total number of writes the device can sustain and the amount of data written to it. Whereas traditional approaches look at reducing the amount of data written, the authors examine device physics and instead focus on increasing endurance.

The gist of the damage model was that the program/erase cycle for NAND requires increased voltage applied to the device that causes more rapid breakdown, particularly during the much longer erase cycle. The goal of the proposed scheme is therefore to reduce the erase voltage. This cannot be done independently, though. The erase voltage is proportional to the width of the threshold voltage distributions used when programming, and shrinking the width of these distributions entails increasing the time used for programming as more voltage increments are required.

This tradeoff between programming time and damage is the basis for DPES (dynamic program and erase scaling), the technique developed by Jaeyong's group. DPES separates flash blocks into different modes: ones that allow a short programming time but require high erase voltage and high damage, and ones with progressively longer programming time that cause less damage. Exposing this tradeoff to the flash translation layer gives it the opportunity to reduce wear when write throughput requirements are lower. Jaeyong then presented an evaluation of such an FTL, autoFTL, which uses utilization of the write buffer as a proxy for throughput requirements. The FlashBench-based evaluation on four I/O traces and two NAND environments (mobile and enterprise) demonstrated an average of 45% improvement in endurance with negligible impact on write throughput.

After the talk, Hangyoung Chun (UT Dallas) asked what would happen if a power failure or breakdown occurred during the now-longer programming step; Jaeyong replied that failure questions are not within the scope of this work. Nitin Agarwal (NEC Labs) inquired about the trace characteristics: the factor of 1,000 difference in buffer size between mobile and enterprise environments was an assumption, and one of the four traces ("mobile'') was a mobile torrent trace. Steven Swanson (UCSD) asked how the programming voltage was varied for the NAND characterization. Jaeyong explained it can be done with an internal test mode for the device.

Dynamic Interval Polling and Pipelined Post I/O Processing for Low-Latency Storage Class Memory

Dong In Shin, Taejin Infotech; Young Jin Yu and Hyeong S. Kim, Seoul National University; Jae Woo Choi and Do Yung Jung, Taejin Infotech; Heon Y. Yeom, Seoul National University

Dong In Shin began with some background: his company, Taejin Infotech, builds memory-based storage servers and is integrating system firmware and software design. To that end, the talk focused on optimizing the performance of storage class memory by modifying the I/O path to achieve lower latency, complementing the group's previous work on improving throughput by merging I/Os within a short time window.

The first solution for latency reduction targeted polling overhead. Although polling is lower-overhead than using interrupts, a too-long poll interval increases response time, and (as substantiated by an experiment) too short an interval can impose a burden on the storage channel. Instead, Dong introduced dynamic interval polling, which adjusts the polling interval based on measured device response time (exactness, Dong explained, is not critical).

And although their previous work improved throughput through I/O batching design, it came at a cost of latency when memory operations that needed to happen after an I/O were delayed until the entire batch completed. To mitigate this problem, Dong's group proposed a new interface—a page-level completion acknowledgment message that would be sent from the controller.

Dong presented an evaluation of the two proposed solutions using a DRAM device modified to have higher latency to emulate PCM. Under four microbenchmark I/O patterns, performance was high for read benchmarks, and performance of an OLTP-style macrobenchmark also increased, particularly for more parallel workloads.

Peter Desnoyers (Northeastern University) proposed DMAing the status of the device into memory to avoid polling. An attendee from Fujitsu asked which solution had more effect; Dong replied there was no analysis to separate the two. The next questioner suggested dynamically changing the speculative latency for polling, given that there is latency variation at the device level. Dong replied this could be done in future work on a FPGA board with real PCM, and there is a plan to do so. Lastly, Michael Condict (NetApp) wondered why PCM was not simply placed on the memory bus. Dong replied it was an eventual goal but that it is difficult to make a memory controller for PCM.

What Systems Researchers Need to Know About NAND Flash

Peter Desnoyers, Northeastern University

Peter Desnoyers' talk began by underscoring the need to cover some common misconceptions about NAND flash, referencing the popular "why most published research findings are false" and observing that it is easy to make mistakes about a piece of hardware that is designed somewhat nonintuitively.

First was the fact that NAND flash no longer comes in just two page sizes: there are now 4 KB, 8 KB, and 16 KB page sizes to increase programming throughput, and 64, 128, 192, 256, and perhaps even 512-page erase blocks to amortize per-erase- block overhead. Flash pages are also not uniform, and come in "fast" and "slow" flavors due to the way MLC flash is read and some MLC devices that treat half the pages as SLC. The final observation from the hardware side was that flash devices don't "just break." Instead, there is a gradual degradation process as cells hold their value with less fidelity, a decline exacerbated by program/erase cycles.

Peter then described some characteristics of flash and SSD controllers. Modern devices actually have a fairly substantial amount of onboard DRAM, and have much more internal parallelism than, say, a traditional RAID array. Among other implications, this means that while large reads can be striped across multiple channels, page-sized reads will only go to one channel and be limited to the throughput of a single chip.

Peter explored the common wisdom that flash is poor for random writes, and showed that this may not be the case if writes are kept within a small region. An experiment showed that when writes were kept within 1/32 of the addressable space, throughput only degraded by about a factor of two (whereas writing over the entire LBA space resulted in poor performance). In addition to the importance of this finding in evaluations, it implied that applications and file systems should maximize locality—for example, by keeping a free block stack instead of a queue.

There was no time for online questions afterwards.

RAID Parade

Summarized by Sangwhan Moon (sangwhan@tamu.edu)

Don't Let RAID Raid the Lifetime of Your SSD Array

Sangwhan Moon and A. L. Narasimha Reddy, Texas A&M University

Sangwhan Moon presented his recent research on the reliability of SSD-based RAID. He pointed out that parity protection is not guaranteed to improve the lifetime of SSD array for two reasons: (1) parity should be updated whenever data is updated, thus increasing the total number of writes done to the SSD array; (2) parity consumes space and this increases space utilization, which results in more write amplification from less efficient garbage collection. Sangwhan explained in the presentation how he estimated the lifetime of SSD arrays considering parity protection and write amplifications. He showed how much parity protection (RAID5) can improve the lifetime of an SSD array compared to striping (RAID0).

Sangwhan and his collaborator introduced Markov models to estimate the lifetime of SSD arrays. They started modeling from a Markov model for each page, and extended this model to cover an SSD array considering parity protection and write amplification. From steady state analysis of the model, they estimated the probability of data loss and the lifetime of SSD arrays.

Sangwhan showed that in a single array the lifetime of parity protection (RAID5) is possibly more than that of striping (RAID0) only when the number of SSDs is more than or equal to eight. He showed many conditions when parity protection becomes competitive to striping in terms of lifetime—for example, when space utilization of the SSD is low, when many factors to reduce writes are considered, and when device failure rate of the SSD is high. In large-scale systems, however, he showed that parity protection is more scalable than striping.

Sangwhan concluded his talk with his finding that parity protection is possibly worse than striping with a small number of SSDs and that this deserves further study. His team has plans to do more evaluations, to validate models, to compare monetary costs of different versions of RAID, and to suggest a scheme to reduce write amplifications.

After the talk, Sangwhan was asked whether there is a result comparing the lifetime of parity protection to the lifetime of single SSD. He answered that, because the evaluation results are the ratio of the lifetime of a target SSD array to the lifetime of single SSD, the ratio shows the comparison.

A Solution to the Network Challenges of Data Recovery in Erasure-Coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster

K. V. Rashmi and Nihar B. Shah, University of California, Berkeley; Dikang Gu, Hairong Kuang, and Dhruba Borthakur, Facebook; Kannan Ramchandran, University of California, Berkeley

K. V. Rashmi gave a talk about her study on erasure coding in the Facebook warehouse cluster (in production). Replication provides reliability and availability in datacenters, but requires a high storage overhead. For data that is accessed less frequently, the use of erasure codes can save a significant amount of storage. Facebook employs a (10,4) Reed Solomon (RS) code on data not accessed for more than three months, which can protect 10 data blocks and 4 parity blocks from up to any four block failures.

As Rashmi and her collaborators observed, the RS code requires considerably more network traffic and disk I/O for recovery as compared to replication. This increases the burden on the network infrastructure. As a solution, they proposed an alternative code that is obtained by modifying the RS code by adding "piggybacks." The "Piggybacked-RS" code reduces the network traffic and disk I/O for recovery, while having the same storage overhead and fault-tolerance capability as RS codes. In the talk, she illustrated the idea of piggybacking via a (2,2) toy example. The Piggybacked-RS code for the (10,4) case can reduce the download and I/O from 20 to 13. They are implementing their solution in the Hadoop distributed file system (HDFS), and plan to provide an empirical evaluation in the future.

There were two queries regarding the measurements: the reasons for the failures and the percentage of network traffic for recovery over total network traffic. Rashmi said she did not have that data. Another questioner asked her to compare the recovery gain between recovery of one block to lazy recovery of multiple losses. Rashmi said it can be done but is orthogonal to the proposed solution. She had showed in the presentation that 98% of recovery operations performed were single-block failures. Two questioners worried about CPU utilization and computation overhead of the Piggybacked-RS code. Rashmi said that the computation overhead, as compared to the RS code, is only in adding and subtracting the piggybacks. She also said that the computation overhead was not really an issue since the datacenter under consideration was limited by resources such as the network bandwidth and the disk I/O, and not by computation.

RAIDq: A Software-Friendly, Multiple-Parity RAID

Ming-Shing Chen, National Taiwan University; Bo-Yin Yang, Academia Sinica; Chen-Mou Cheng, National Taiwan University

Bo-Yin Yang presented the design and implementation of RAIDq, a software-friendly multiple-parity RAID. As storage systems scale out, triple-parity RAID (RAID7) and quadruple-parity RAID (RAID8) are demanding. RAIDq uses linear code and it is backwards-compatible such that RAID5 and RAID6 are special cases of RAIDq. RAIDq exploits hardware-accelerated instructions, which are already supported by CPUs for RAID5 and RAID6. This can boost the encoding and decoding speed of RAIDq. RAIDq has a limit on the number of protected disks. Bo-Yin showed why and provided the numbers in the presentation.

Bo-Yin continued by showing why current RAID cannot be easily extended to triple-parity or quadruple-parity RAID using finite field representation. When RAID is extended beyond RAID7 over a certain finite field, there is a condition that the RAID7 should satisfy: to make a generator matrix invertible, three determinants should be nonzero. The number of protected disks, for example, cannot be more than 21 if RAID8 is over finite field F (256); otherwise, the condition may be unsatisfied. If compatibility with existing RAID6 is disregarded, the number of disks can be up to 27.

RAIDq addresses the general case of RAID with an arbitrary amount of checksum data up to certain limits. Bo-Yin explained how he split a scalar multiplication to hardware-supported table lookups and XOR operations. RAIDq7 improves its performance by using those hardware accelerated instructions. He showed how he split a multiplication in extended finite fields to the arithmetic over F(256) as well. The encoding and decoding throughput of many candidates for RAIDq 8 with different finite fields were evaluated. RAIDq is implemented on SSSE3 (Supplemental Streaming SIMD Extensions 3). Bo-Yin compared RAID6, RAIDq 7, and the proposed candidates for RAIDq 8 to Reed Solomon (RS) code. The result showed that most of RAIDq have acceptable maximum number of protected disks, up to 92 for RAIDq 8, whereas encoding and decoding throughputs are higher than RS code.

Virtual Machine Data

Summarized by Varun Prakash (vsprakash@uh.edu)

Efficiently Storing Virtual Machine Backup

Stephen Smaldone, Grant Wallace, and Windsor Hsu, EMC Corporation

The authors had a chance to present the summary of their work on the previous day during the poster sessions. The talk started with an intro about the current backup scenarios and logical vs physical types of backup. With a focus on virtual machine backup, the question being addressed in the paper is, "Can physical backup be as efficient as logical backup?" The testbed being used for experiments is a typical industrial testbed and a clear-cut methodology to perform experiments was provided. The experiments included a comparison of compressed and noncompressed data and the performance effects of deduplication. The authors also included the effects of file systems and metadata on the virtual machine archiving process.

The results are attributed to some effects within the file system, such as file-system churn. Some of the counterintuitive results from the analysis provided a lot of insight and a deeper understanding of some of the inner workings of the system.

After a note on the related work, the authors concluded that physical backup is easier to manage and can be more space-efficient than logical backup.

Improving I/O performance Using Virtual Disk Introspection

Vasily Tarasov and Deepak Jain, Stony Brook University; Dean Hildebrand and Renu Tewari, IBM Research—Almaden; Geoff Kuenning, Harvey Mudd College; Erez Zadok, Stony Brook University

The use of virtual machines can sometimes be a compromise between performance and loss of file system-related data that do not trickle down to the hypervisor. This serves as a motivation to the researchers, who try to answer the question, "How can semantics of the virtual machine and metadata be preserved in the file system?" After giving a short background on some of the ways the hypervisor and storage architectures access data, Vasily Tarasov noted the ways in which the guest OS, mainly isolated in previous black-box approaches, provides assistance. There are also various levels of file system awareness of the guest OS by the lower layers, which in many approaches has no or only incomplete information about the guest file system.

The internal working of the introspection code is said to be based on a process of reverse translation to find a sweet spot within the system. Some of the areas of focus include how to perform mapping management, the types of optimizations that are commonly found in the market, and the effects of tiered storage on unused-block detection, compression, and optimization.

Some of the results obtained by the group include increased performance in terms of runtime reduction. The system is said to be efficient while performing some types of operations, such as delete and read-intensive tasks.

Low-Cost Dedup for Virtual Machine Backup

Wei Zhang, Tao Yang, and Gautham Narayanasamy, University of California, Santa Barbara; Hong Tang, Alibaba Inc.

The talk started with an introduction to deduplication technologies and the presentation of a low cost method of deduplication. The motivation of the project is derived from the fact that there is a need to continuously improve the service reliability. Some of the solutions include looking out for inexpensive storage and architecture considerations. In this regard, a note on some of the requirements is also made in terms of storage and architectures.

The key idea to improve performance is to separate the process of deduplication from data backup. Background information about virtual machine snapshots are provided and the system that has been implemented is discussed in stages.

Some of the core operations performed on virtual machine snapshots, such as inclusion and deletion of the snapshot and its effects on the overall performance of the deduplication, were discussed. After a brief note on the testbed on which the experiments were conducted and the parameters used, some of the results of the virtual machine backup performance, such as backup time and the effects of multiple partition sizes, was discussed.

Storage Performance and Energy

Summarized by Kai Ren (kair@cs.cmu.edu)

Challenges in Getting Flash Drives Closer to CPU

Myoungsoo Jung and Mahmut Kandemir, The Pennsylvania State University

Myoungsoo Jung started his presentation by showing that the traditional I/O controller hub (i.e., south bridge) is not suitable for today's high speed solid state disks (SSDs), since the old interface can only provide limited bandwidth (600 MB/s). This leads to an alternative approach to get flash memory (SSDs) closer to the CPU by using the PCIe interface.

Myoungsoo presented two representative architectures designed for PCIe SSDs: from-scratch SSD (FSSD) and bridge-based SSD (BSSD). FSSD employs FPGA or ASIC-based native PCIe controllers, and runs flash management software in the host OS. In contrast, BSSD employs an on-board PCIe-to-SAS (or SATA) bridge controller, and runs flash management software inside the on-board controller.

Myoungsoo discussed the tradeoffs of the two architectures and provided useful performance analysis. He showed that FSSD outperforms BSSD. Specifically, FSSD is 40% better on latency, and proved 21% and 80% better for read and write throughput, respectively; however, he also found that FSSD requires more memory in the host OS than BSSD (about 9 GB more), and consumes four times more CPU cycles to complete I/O requests.

Lastly, he compared the two architectures with multi-worker benchmarks. With more workers, both architectures suffer higher latency, while the IOPS of FSSD increases; however, when stressing FSSD with many more workers, it continues to consume more memory and CPU, and therefore its advantage decreases.

Someone asked whether there is a middle ground to take advantage of both architectures, by moving a part of flash software (i.e., FTL) into host-OS while still keeping the SAS interface. Myoungsoo said such middle ground is difficult to find because SATA or SAS cannot provide any detailed information about low-level operations required to perform FTL.

Runtime I/O Re-Routing + Throttling on HPC storage

Qing Liu, Norbert Podhorszki, Jeremy Logan, and Scott Klasky, Oak Ridge National Laboratory

Qing Liu was trying to solve an important problem he faced when managing an HPC cluster with 100,000 cores for many concurrent data-intensive applications in Oak Ridge National Laboratory: concurrent execution of many applications causes contention in shared storage systems and, therefore, a reduction in I/O throughput. Many applications have synchronization points, and the slow down of one thread will cause other hundreds of thousands of cores to wait, leading to lots of wasted CPU cycles. Qing described some previous efforts to solve similar problems under other contexts such as explicit QoS support, LFS, and chain logging; however, these previous works may introduce other synchronization points that may not be suitable for HPC.

Qing proposed a network-based solution by rerouting work to nodes with lower loads, and throttling throughput to achieve a higher degree of balance. The proposed approach comes from the observation that the load on shared storage is bursty and unstable, so an offline approach is not likely to mitigate hotspot problems effectively; and second, the imbalance of the storage system is often caused by a few outliers. Qing described their online I/O rerouting mechanism inspired by these observations. Their solution implements a virtual messaging layer (VML) to redirect I/O traffic to less loaded storage locations. VML uses a group-based two-level control framework to coordinate the I/O traffic in a scalable fashion. I/O redirecting happens during write phases, and can mitigate the I/O variability; however, it may cause an imbalance of data placement and affect later read phases. To address this problem, they use a threshold to limit the amount of write traffic that can be rerouted. Some results on synthetic benchmark were presented to demonstrate the mitigation of I/O variability in the write phase (about 2x speed up), and the slow down in the read phase (about 2.3x slow down). Finding the balance between read and write phases is left for future work.

An audience member asked about the source of imbalance in storage workloads. Qing answered that the imbalance comes from both concurrent execution of different HPC applications and the intrinsic property of these applications. Someone else asked why previous QoS techniques could not be applied to solve the problem. Qing answered that QoS techniques may introduce a centralized scheduler that might not scale to 100,000 cores. Another person wondered whether taking information from applications can help with solving this problem. Qing replied that there is a wide range of applications running in the cluster, and it is difficult to optimize the storage system for just one portion of these applications.

Specialized Storage for Big Numeric Time Series

Ilari Shafer, Raja R. Sambasivan, Anthony Rowe, and Gregory R. Ganger, Carnegie Mellon University

Current databases are not optimized for large-scale time series data: they don't consider properties of time series data, which are mostly append-only and have a lower durability requirement, and they do not optimize queries needed by time series analysis such as range queries, multi-resolution access, and even multiple streams analysis. Ilari Shafer's talk focused on a specialized storage system that uses lossless compression for efficiency and optimized for range and multi-resolution queries.

His proposed approach considers the append-only properties and uses a memory buffer to keep most recently inserted data. By using a column-based schema, the storage layout stores timestamps and value fields separately. And the storage engine can also use run-length encoding and delta compression to compress data columns. A preliminary experiment shows that the storage schema specialized for time series data can reduce the data size by 50% of the general approach. Future work will focus on finding the right tradeoffs of space efficiency and query performance, as well as seeking appropriate API for time series queries.

Someone asked if the range queries are on the value field instead of the timestamp field, whether the append-only property goes away. Ilari answered that the storage system can use the append-only property to optimize range queries on the value field, and needs to build a secondary index. Another question was whether using SSD will change the system design. Ilari answered that SSD adds differences and is a good direction for future work.

FDIO: A Feedback Driven Controller for Minimizing Energy in I/O-Intensive Applications

Ioannis Manousakis, Manolis Marazakis, and Angelos Bilas, Foundation for Research and Technology - Hellas (FORTH)

Ioannis started by highlighting the opportunity to save energy in datacenters by using a holistic approach to track overall energy consumption of all hardware components. He pointed out that previous work, which focuses on the CPU utilization-based method, is not effective enough, especially for I/O-intensive workloads.

To demonstrate this idea, he presented a testbed system with detailed per-component hardware instrumentations. He compared standard Linux CPU governors with the best possible operating settings (by enumerating static DVFS setting points to find the optimal point) under three different I/O workloads: TPC-W, nsort, and TPC-H. For all cases, the optimal setting can double the system energy efficiency (reducing EDP by 50%) over existing Linux governors.

In order to find the optimal DVFS setting point, the authors proposed a feedback-driven controller called FDIO, which minimizes system-level energy-related metrics (EDP) instead of just CPU utilization level. FDIO tracks the energy consumption of all hardware components from the sensors, and enumerates the possible DVFS setting to determinate the best setting for current workload.

In the nsort benchmark, they demonstrated that FDIO can distinguish the I/O-intensive phase and the CPU-intensive phase to find out the optimal setting for both. Future work is to explore ways to track multithread applications using control theory to optimize the schema and other directions.

An attendee asked whether using other existing hardware instead of specialized hardware to track energy consumptions is possible. Ioannis answered that the current Intel CPU provides many fine-grained performance counters for CPU and memory, and one can also use an analytic model to estimate the energy consumptions of other peripherals. Someone asked whether there is a way to increase energy efficiency for running memory-bound applications. Ioannis answered that there are no good solutions currently for these types of applications.