;login:

THE MAGAZINE OF USENIX & SAGE

June 2003 • volume 28 • number 3

inside:

CONFERENCE REPORTS

2nd USENIX Conference on File and Storage Technologies (FAST '03)

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

2nd USENIX Conference on File and Storage Technologies (FAST '03)

SAN FRANCISCO, CALIFORNIA MARCH 31–APRIL 2, 2003

KEYNOTE ADDRESS

DATA SERVICES – FROM DATA TO CONTAINERS

John Wilkes, Hewlett-Packard Labs

Summarized by Scott Banachowski

After a few opening remarks from conference chair Jeff Chase, John Wilkes of HP Labs kicked off the conference with the keynote address. As he introduced the talk, Wilkes clued any bored listeners to look for themes in the photographs scattered throughout the presentation slides. Wilkes's keynote foreshadowed many of the themes that would appear in the following conference sessions: how to deal with the rising complexity and ability of storage systems and meet our expectations for new capabilities.

The solution to storage problems lies in using Quality of Service (QoS), because it encompasses everything we'd like to say about our storage problems. Therefore the path to a solution includes defining data QoS needs, using storage QoS abilities, and automating storage and data management.

The enterprise IT plan is a complex, large-scale system; Wilkes demonstrated this assertion with a convoluted enterprise IT architecture schematic. Some of the requirements of these systems resemble those of existing large-scale scientific applications, so we can look to these applications as predictors of future trends: they access huge quantities of data using specialized access patterns. As examples, Wilkes reviewed the storage systems required by the human genome sequencing engine and the CERN electron collider.

It is important for storage researchers to distinguish between data and storage, because we often confuse the data's attributes with those of the containers. Wilkes outlined many data metrics, pointing out that it is easy to measure amounts, rates of growth, or access patterns, but difficult to get a handle on resilience or security. It is important to consider that not all data are created equal (some have little value, some never change, and some can be regenerated) and that data have differing lifetime and security requirements. All of these attributes can be captured by QoS, which provides storage systems with both a set of objectives and a contract for service. In a brief overview of storage containers, Wilkes covered new technologies such as MEMS, MRAM, and smart disks (bricks). He proposes using the SNIA shared storage model to get a handle on how to set up large storage systems.

Recognizing that storage systems are just a part of larger computing systems, it is important to grapple with the management challenges. The main challenge is keeping administration costs low by automating tasks. We have already generated many techniques for managing systems, so Wilkes recommends that we learn to use existing techniques before creating new ones.

The key to develop future storage systems is to embrace complexity. In this era, we must use a data-centric viewpoint for putting everything together, and this must be driven by QoS. Wilkes summarized the problem as moving from "some of the parts to sum of the parts" as the required step to accessing data anywhere and anytime. During the Q&A, someone noted the agricultural theme running through Wilkes's slides, and asked what the equivalent to fertilizing and weeding is in the storage field. Wilkes chuckled but supplied no answer.

SESSION: INTERNET SCALE STORAGE

Summarized by Preethy Vaidyanathan

POND: THE OCEANSTORE PROTOTYPE

Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz, University of California, Berkeley

Peter Honeyman from the University of Michigan chaired this first session of the conference. Sean Rhea presented OceanStore, a global-scale storage system, and the prototype Pond, a self-organizing, self-maintaining, secure Internet-scale file system among untrustworthy hosts. (This paper received the Best Student Paper award.)

The main challenges in a global-scale storage system are availability and manageability. All the resources in the system are virtual, and replication is used to provides fault tolerance and reliability. Tapestry, a decentralized scalable object location and routing system, is used in this prototype to identify the resources.

The prototype uses erasure codes and a modified Byzantine agreement protocol to provide fault tolerance and consistency among the replicas. Erasure codes are more durable than data mirroring for the same space. The Byzantine protocol was modified to decrease the number of messages passed to make replica copies consistent. Each object is assigned a primary replica by an inner ring server. The updates are in-place among the inner-ring servers.

The Pond prototype was tested using two experimental testbeds at Berkeley and PlanetLab (*http://www.planet-lab.org*). Andrew benchmark test results compared to NFS show improvements in read access and performance degradation on writes. Rhea explained the write cost and limitations as due to erasure code. This cost would be alleviated when servicing large writes. Pond has good performance in other benchmark experiments.

Pond source code is available at *http://oceanstore.cs.berkeley.edu.*

DATA STAGING ON UNTRUSTED SURROGATES

Jason Flinn, Intel Research Pittsburgh and University of Michigan; Shafeeq Sinnamohideen, Niraj Tolia, and M. Satyanaryanan, Intel Research Pittsburgh and Carnegie Mellon University

Mobile computers are increasing in popularity, and Jason Flinn presented a mechanism by which surrogates close to the mobile device can be used as data staging stations, reducing transfer latency. This architecture provides less latency for the client when accessing data, as the data is now retrieved from the surrogate and not the file server.

The data staging architecture consists of a client proxy that observes file system traffic and initiates a surrogates help if need be. A data pump near the file server acts as an intermediate point between the client and the server. When a client accesses data, the pump authenticates the message, reads from the file system, encrypts it, and sends the cryptographic hash to the client and the data to the surrogate. All this is done through a secure channel. The client reads the data from the surrogate, decrypts it and checks validity using the hash.

The architecture design is independent of the underlying file system. For experimentation, this design is implemented on the Coda file system. The data staging design assumes client, file server, and data dump file system to be trustworthy and entrusts the surrogate and the network.

Flinn presented two experiments to test the data staging architecture. The first tested the performance of aggressive perfecting in this architecture. The results shows that surrogate data miss affects the performance more than the surrogate having data that are never accessed. The second set of experiments tested what factors affect the performance of the system. The results showed that latency hurt the performance more than bandwidth, which iterates the assumption for this work.

The source code of this project can be obtained at *http://info.pittsburgh.intel-research.net.*

PLUTUS: SCALABLE SECURE FILE SHARING ON UNTRUSTED STORAGE

Mahesh Kallahalla, Hewlett-Packard Labs; Erik Riedel, Seagate Research; Ram Swaminathan, Hewlett-Packard Labs; Qian Wang, Pennsylvania State University; and Kevin Fu, Massachusetts Institute of Technology

Mahesh Kallahalla presented a cryptographic storage system for secure file sharing. Data security is different from network security, and Kallahalla described Plutus, a decentralized key management system where all keys are handled by the client with minimum trust on the server. Plutus architecture is scalable and secure, as there is no single point of failure, because the client does the encryption and decryption work and the server acts as a data store.

Blocks of the file are encrypted with symmetric key called file-block key. There is a file-lockbox key for the file. To minimize the number of keys generated, files with similar attributes are grouped into file groups. All the files in the file group share the same file-lockbox key. The architecture differentiates between readers and writers by using file-verify and file-sign public-private key pairs.

File updates might result in file group fragmentation. This is handled in the Plutus architecture by key rotation. Each update results in the creation of a key version, not a new key. The owner of the data can only generate the next version. The readers keep track of the newest version of the key, and previous versions can be rolled back from the current version. A prototype has been designed using OpenAFS. The Plutus system is tested using UNIX traces and synthetic benchmarks. Kallahalla concluded that in spite of the overhead of encryption and decryption, Plutus performance is favorable with key points such as file grouping, key rotation, and lazy re-encryption.

SESSION: FILE STORAGE

Summarized by Scott Banachowski

METADATA EFFICIENCY IN VERSIONING FILE SYSTEMS

Craig A. N. Soules, Garth R. Goodson, John D. Strunk, and Gregory R. Ganger, Carnegie Mellon University

The first talk of the Storage Session, chaired by Margo Seltzer of Harvard, came from Craig Soules of CMU. Soules presented the Comprehensive Versioning File System (CVFS), a log-based file system that keeps old versions of data using structures that reduce the storage overhead of metadata, essentially trading back-in-time performance for space.

A versioning file system keeps multiple versions of data for backing out mistakes, failure recovery, and history analysis. Current versioning systems write a new copy of the metadata for each version of a file, leading to high metadata overhead. The goal of CVFS is reduction of storage overhead, which is accomplished by combining journaling and b-trees.

The system maintains the most current metadata version and differences between previous versions, stored in a journal. The journal approach saves space because it only records incremental changes, but retrieving previous versions is not efficient because all previous versions must be unrolled in sequence to recreate the desired version. To reduce the roll-back time, CVFS occasionally store an entire version, i.e., a checkpoint. Multiversion b-trees provide an efficient structure for storing multiple versions of data using keys comprised of a name/time pair. B-trees are more efficient for single-lookup operations than journals, so CVFS uses b-trees to maintain directories, where lookup is the most common operation and modifications are infrequent.

The performance of CVFS was evaluated by playing back 1 month of NFS traces that contained 164 GB of data traffic from 30 users. Compared to conventional versioning system, CVFS saved 53% in file metadata and directory space. The performance relative to other systems diminishes when keeping less comprehensive data, for example storing only on-close versions or periodic snapshots. During the Q&A, someone questioned Soules about the usefulness of such comprehensive versioning, considering that many writes may never be seen by the file system due to caching, to which Soules replied that it depends on the application.

yFS: A Journaling File System Design for Handling Large Data Sets with Reduced Seeking

Zhihui Zhang and Kanad Ghose, State University of New York, Binghamton

Recognizing that most file system designs are based on file-size assumptions from years ago, Zhihui Zhang presented yFS, with the goal of handling large and small files with equal ease. yFS was implemented in FreeBSD.

The features of yFS include extent-based allocations, multiple inode formats, b*-tree structures for managing inode data, support for large directories, and lightweight logging. The file system divides the disk into allocation groups, each containing its own metadata. Space is allocated to files in both fragments and blocks, although, unlike other segment-based systems, there is no restriction concerning which segment a file begins, and fragments may have variable size. For large files, there are competing goals of contiguity and locality; yFS scatters large files across allocation groups.

yFS was compared with FFS enhanced with Soft Updates using four benchmarks: a kernel build, the extraction of an archive file, the PostMark benchmark, and a file system aging test. Without Soft Updates, the synchronous metadata updates of FFS lead to performance that is not comparable to yFS. However, even with Soft Updates, yFS outperformed FFS in all measurements except for one phase of the compilation benchmark.

Semantically-Smart Disk Systems

Muthian Sivathanu, Vijayan Prabhakaran, Florentina I. Popovici, Timothy E. Denehy, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

Remzi Arpaci-Dusseau explained the concept of “semantically-smart disk systems.” Storage systems are currently layered into the file system and the disk or RAID system; the origin of this separation is the hardware/software boundary, but now each layer is becoming increasingly complex. Because layers are separated by a bus or protocol, the semantics of file system operations are lost in the disk layer, so semantically-smart disks aim to reacquire this information in the disk layer using both offline and online techniques. The approach allows RAID systems to exploit their processing and memory capability without changing their interface.

Semantically-smart disks understand file system operations, and discover the layout of on-disk structures and operations by reverse engineering the block stream. Static knowledge of file system layout is determined with the aid of a gray-box tool called Extraction of Filesystems (EOF). EOF creates a disk traffic pattern that, when observed by the disk, provides hints that allow the smart disk to determine the types of blocks and their layout. With this information, the smart disk may then take steps to improve performance, for example by automatically caching inodes and directory blocks in NVRAM.

The paper includes several case studies for using semantically-smart disks (SDS), but during the talk Arpaci-Dusseau focused on adding its secure deletion feature. By detecting when files are deleted, the disk can automatically remove its dead-blocks so that they cannot be reread using magnetic microscopy techniques. The file system cannot reliably do this, because it may absorb writes in cache or may leave stray blocks on the disk. Using SDS it is possible to detect deletes, which traditional RAID systems cannot do, and overwrite the file’s data blocks with patterns multiple times, so they cannot be reread. In the Q&A session, someone asked if it is possible to also learn semantics through the interfaces of object-based storage. The approach may be similar in philosophy but it is still an open issue.

INVITED SESSION: PETABYTES AND BEYOND

Reagan Moore, San Diego Supercomputer Center; Thomas M. Ruwart, University of Minnesota Digital Technology Center, Intelligent Storage Consortium; and Clod Barrera, Director of System Strategy, IBM Systems Group

Summarized by Preethy Vaidyanathan

The “Petabytes and Beyond” panel was hosted by Jeff Chase of Duke University.

Reagan Moore started out by listing the challenges that will be important in the future for applications dealing with large amounts of data. He pointed out that, based on the current data growth trend, handling large amounts of data would require organizing data into collections. Some major challenges would be to tag these large amounts of data to generate information, organizing information into collections, querying collections for relationships (data mining) and organizing relationships in concept spaces.

Moore presented some of the projects dealing with petabytes of data: NASA Earth Observing Satellite, Large Hadron Collider, and NSF Teragrid. Teragrid is a project that handles large volumes of data in a distributed environment. It aims to provide a collective set of resources greater than any one site can provide. The participating groups in this project are the San Diego SuperComputer Center (SDSC), the National Center for Supercomputing Applications (NCSA), the Argonne National Laboratory (ANL), and CalTech. Some of the challenges include managing the resources over WAN, discovery of data, and naming conventions.

Other works include Digital Library, data grids like Grid Bricks that build cheap storage systems (bricks) using common disks, and IDE drives and tape archives. A data grid accesses distributed resources and should manage namespace, user authentication, and user access control across bricks.

Moore concluded by presenting specific directions for some of the challenges. Discovery of data in this environment is essential. This can be provided by an infrastructure-independent naming convention. Data discovery is implemented by cataloging a database that manages the logical name and abstracts the physical location. Because of WAN latency management is another challenge. The SDSC Storage Resource Broker focuses on this problem. One solution is to enable the application at the destinations to pull data from remote resources by sending an aggregate message that eliminates the large overhead of a number of individual messages.

Thomas Ruwart presented "Storage on the Lunatic Fringe." He introduced the DoE Accelerated Strategic Computing Initiative (ASCI), High Energy Physics (HEP), NASA Earth Observing System Data Information Systems (EOSDIS), DoD NSA and DoD Army High Performance Computing Centers, and the Naval Research Center as the "lunatics" who are dealing with petabytes of data. The problem that these projects are looking at is what the industry will face in 5–10 years, so a lunatic in this scenario corresponds to a visionary.

Ruwart gave a brief historic outline of large-scale computing resources, starting with supercomputer centers in the '90s to the current ASCI Q and the ASCI Red Storm, Purple, and NASA RDS.

He then went on to present some specific problems. In HEP, thousands of scientists look at large datasets with different access pattern considerations. The Data Grid project is a distributed architecture and the main issue is managing data for long periods of time. How to handle a trillion files is the challenge for the NSA project. He said that considering 256 bytes of metadata per file, for a trillion files this itself would result in 256 TB. If this number was not enough to overwhelm, Ruwart raised the question of backups in this scenario. Other problems raised included searches for content in these datasets, security, and availability.

With data on such a scale, legacy block-based file systems will not work. A vision for the future is more intelligent storage devices whose functionality would migrate from the operating system to the storage device. The storage device apart from storing data, should handle managing and administering data.

Some of the technologies addressed by the Lunatic Fringe include object based storage devices, intelligent storage, and data grids.

Clod Barrera presented "Size and Shape of Things to Come." He stated that the main concern in the future apart from performance and scalability, would be data management. He stressed that the ease and cost of management should be taken seriously.

He presented life science research as an audience who will need to deal with petabytes of storage requirements. At this requirement level, data access consistency, error recovery, high performance, and a scalable file system would be essential. One such project dealing with a scalable file system is the Storage Tank project. A storage network of petabyte scale might start with a fiber channel but could be other technology such as IP over SCSI. Holographic storage is an emerging technology that simplifies content searching. Barrera concluded by pointing out that an intelligent system with knowledge of the different storage technologies can map the application to the right storage technology for optimal performance.

SESSION: STORAGE SYSTEMS

Summarized by Preethy Vaidyanathan

Using MEMS-Based Storage in Disk Arrays

Mustafa Uysal and Arif Merchant, Hewlett-Packard Labs; Guillermo A. Alvarez, IBM Almaden Research Center

Mustafa Uysal presented the first talk (given Best Paper award). Two widely used storage technologies are NVRAMs and disks. NVRAMs are faster, more expensive, and less reliable, whereas disks are slower and cheaper devices. Current I/O performance is still bounded by disk access.

MEMS is a new technology being developed with access characteristics between NVRAM and disk. They provide persistent and nonvolatile storage like disks but have different characteristics. They have no rotational delay, with slow moving parts making acceleration easy and high density storage providing a short seek distance. Uysal expanded upon the various architecture alternatives for this new device, assuming it was cost-effective and useful.

Uysal presented five array architectures. MEMS replacing disk (MEMSdisk), MEMS replacing NVRAM (MEMScache), and three hybrid architectures: MEMSmirror, Logdisk, and DualStripe. In the hybrid architectures, MEMS does not totally replace any device. MEMSmirror has a MEMS device as mirror for the disk. No access or layout change is needed here: reads of data not in cache are handled by MEMS and writes are propagated from NVRAM.

Logdisk and DualStripe have data from MEMS mirrored in disk for redundancy.

In the Logdisk architecture, updates to MEMS are propagated to disk in a log-structured manner. Reads are handled by MEMS and sequential reads can be effectively handled by disk. DualStripe is a dynamic architecture. Reads if cache miss are handled by MEMS for a short queue length and handled by disk for queues greater than a threshold or if it's a large sequential read. Sequential access detection is implemented in firmware.

The different array architectures were tested for synthetic and trace workloads. The overall conclusion: having a MEMS device in your storage array architecture will be cost-effective and efficient. The hybrid architecture provide a good cost/performance benefit and Logdisk provides the most cost-effective architecture.

In the question period, Uysal clarified that this work studied the placement of MEMS in disk arrays so the scheduling policy was the same as disk. Another question led to the conclusion that a possible 3-level hierarchy of NVRAM, MEMS, and disk would be an extension to this work when the exact characteristics of the MEMS device is known.

OPTIMIZING PROBE-BASED STORAGE

Ivan Dramaliev and Tara Madhyastha, University of California, Santa Cruz

Probe-based storage or MEMS is a new technology with characteristics such as low power consumption, high density, high parallel tip movement producing high throughput, and no rotational movement. These devices can be modeled to different design points each resulting in different performance measures. This would answer the question of which workload would best suit probe-based architecture. Madhyastha outlined a parameterized analytical model to compute average request latency for MEMS devices.

The Probe-based storage is characterized by X and Y movements, with no rotational movement such as disks use. The data layout goal is to minimize movement for consecutive requests, similar to traditional disks. There is a mobile part which moves when servicing a read or write request. The repositioning time comprises seek time and time taken while moving with a constant velocity in Y direction to access data (transfer time).

Madhyastha illustrated how these two times can be calculated for this device. In the model, the seek time depends on bit per tip, distance moved in the X and Y directions, bit width, acceleration, and settle time. The transfer time is proportional to the data per tip. The service time model gives a formula into which values can be plugged to compute the service time of the request.

This model was tested by comparing it with simulation results for a wide range of parameters. Block level traces were used to test the performance of the model. The error computed was small (up to 15%) when compared to simulation.

In the Q/A section, Madhyastha agreed with the observation that the reliability of these devices should be considered in future research.

ARC: A SELF-TUNING, LOW OVERHEAD REPLACEMENT CACHE

Nimrod Megiddo and Dharmendra S. Modha, IBM Almaden Research Center

Dharmendra S. Modha discussed how to manage cache or what page to replace to maximize hit-ratio. The cache replacement strategy presented was with respect to demand paging.

Two popular techniques, Least Recently Used (LRU), and Least Frequently Used(LFU), are algorithms that have long been used for cache replacements. LRU captures locality of reference; LFU, the frequency of reference. Modha presented a new scheme, ARC, that captures both these characteristics by maintaining two self-consistent lists. The first lists the pages seen only once and the second lists pages seen at least twice.

In ARC a sliding window of the size of the cache is used to determine what page to replace. The sliding window overlaps the two lists and the percentage of overlap dynamically varies depending on the workload. This implementation has low computational overhead and is tested with a wide range of trace data. ARC consistently outperforms LRU and has similar performance to an offline replacement algorithm that is optimally tuned for the workload.

In the Q&A Modha clarified that the sliding window starts initially with the midpoints in the two lists and the sliding movement is sensitive to the request.

The source code is available at *http://almaden.ibm.com/cs/people/dmodha*.

SESSION: SHARING BLOCK STORAGE

Summarized by Nate Edel

FAÇADE: VIRTUAL STORAGE DEVICES WITH PERFORMANCE GUARANTEES

Christopher R. Lumb, Carnegie Mellon University; Arif Merchant, Hewlett-Packard Labs; and Guillermo A. Alvarez, IBM Almaden Research Center

Christopher Lumb described work at HP Labs on the Façade system, a storage system that provides service level guarantees. Unlike existing solutions, which don't differentiate between workloads, the Façade system is able to adapt to changing workloads to attempt to meet each separate workload's service level objective (SLO).

Façade works by intercepting storage requests and prioritizing them based on their SLOs. SLOs are latency bounds at a given rate of I/O operations; a workload may have separate SLOs for read and write operations, and multiple workloads/SLOs may share one RAID.

The system has three components: the I/O scheduler, the controller and monitor, and a target queue. The I/O scheduler receives all requests and then timestamps and queues them. The scheduler then watches the wait times and dispatches the IO requests based on earliest deadline to the target queues. The adaptive controller and monitor monitors the response times and workloads; if requests aren't meeting deadlines, it will makes changes to the target queue behavior.

The target queue maintains latency requirements by shrinking in depth when latency targets are not met to decrease throughput, and growing in depth when latency targets are met to increase throughput. The target queue growth rate is conservative, because shrinking queue depth is harder, as I/O operations have to first drain the queue to the desired depths, and further I/Os have to finish to allow new operations to enter the queue.

Lumb presented benchmark numbers for a sample set of workloads with three different SLOs, and showed that without Façade, the different workloads would have roughly equal latency and would not meet their targets. With Façade, by reducing the I/O rate for a continuous process, the other two burst workloads met their latency targets almost all the time. There were spikes during transition from high throughput to SLO compliance; he noted that these were optimizable but that the best way to do so was not clear.

Finally, Lumb compared two workloads on separate arrays against Façade on a higher resource single array; with Façade, the combined array had essentially the same latency for both processes and the same throughput for the burst workload. However, with Façade, the continuous workload could take advantage of bandwidth unused by the bursty workload when it was not active, allowing some degree of overprovisioning to be avoided.

In the question-and-answer period, someone asked if Façade works well when all workloads compete equally, or what happens when workloads compete for different resources? This was noted as not clear, but it would be an interesting experiment, which may be able to account for some aspects, and they could increase the complexity of the model by taking into account other aspects, but Lumb was not sure if it would make a difference. Someone else asked if the project had tested other metrics, such as bandwidth? The project did not, but it would be simple to use bandwidth rather than request rate if that was preferred. Another question was how Façade prevented starvation? It doesn't handle admission control; they assume the system could eventually service all requests.

Design and Implementation of Semi-preemptible IO

Zoran Dimitrijevic, Raju Rangaswami, and Edward Chang, University of California, Santa Barbara

Zoran Dmitrijevic described work done on developing a system for preemptible disk-access. The key benefit of preemptibility is decreasing the initial latency of high priority IO requests. The size of other, lower priority I/O requests will not have as great an impact on these requests; and premptibility may be able to improve other scheduling.

Overall, the time to execute a read request depends on several factors – wait for seek, rotational delay, and the maximum IO size and maximum disk IO size. These are typically selected to balance throughput and latency with all tasks being equal; without normal, non-preemptible IO, the total response time is the waiting time plus the service time. For an average command, the expected wait time is one-half of the service time for an average IO. Preemption allows elimination of waiting time, and its key metric is the reduction of expected waiting time.

Without preemption implemented on the disk itself, the proposed implemention of semi-premptible IO splits lower priority IO into several commands. This allows the controller or OS to interpose higher priority IO requests into the stream of smaller requests, a technique called chunking; because of disk read prefetching and buffering, the overhead of IO bus traffic and kernel CPU activity remains close to constant.

Along with chunking, Dmitrijevic discussed two other techniques. The first, just-in-time-seek, attempts to calculate pre-seek slack using the rotational delay to make that time preemptible – pre-seek slack can be used for "free" perfecting and seek-splitting. The second, seek-splitting, takes long seeks and splits them into multiple shorter seeks. This allows preemption of seeks and takes advantage of rotational slack. The down side is that multiple short seeks take slightly longer than a single direct seek.

There were several implementation issues addressed: disk block mappings needed to be implemented, the optiminal chunk size had to be determined, and rotational factors and seek curves were analyzed. For the optimal chunk size, Dmitrijevic noted there were both lower and upper bounds: a minimum size, below which throughput suffered, and a maximum size above which it seemed that prefetching might not continue to be a factor. SCSI and IDE drives showed similar curves, although SCSI was more efficient for a range of smaller transactions.

The experimental implementation was a user-mode driver running on the SCSI generic driver on Linux, and tested using traces from a specially instrumented Linux kernel. It was tested with a simulated workload of random IO using FIFO and elevator scheduling, as well as with TPC and multimedia streaming traces. Expected waiting time is much lower with some workloads, and only very slightly worse throughput with random accesses.

The talk closed by noting that the contributions were measuring the preemptibility of disk accesses and showing that preemptibility can cut down waiting time. Noted future directions were the use of semi-preemptible IO in scheduling algorithms, and a QOS disk scheduler for Linux.

John Wilkes asked if it would be possible to implement this on the on-disk controller. The response was that while it may be possible, existing drives would need more onboard computing power on the drive; while Dmitrijević was unsure how much more would be needed, he indicated that it should be possible.

Block-Level Security for Network-Attached Disks

Marcos K. Aguilera, Minwen Ji, Mark Lillibridge, John MacCormick, and Erwin Oertli, Hewlett-Packard Labs; Dave Andersen, Massachusetts Institute of Technology; Mike Burrows, Microsoft Research; Timothy Mann, VMware; and Chandramohan A. Thekkath, Microsoft Research

Marcos Aguilera described the Snapdragon file system prototype, which was created to test a security model for network attached disks (NAD) storage. This was contrasted with standard distributed file systems, with disks on a server and all accesses via that server; in that case, the server is the main performance and reliability bottleneck.

NAD is like a storage area network (SAN), but is simpler because there is no separate network for storage; in either case, a server is used only for metadata, and file access is direct to disks over the (shared or separate) network. Because the server is out of the data access path, this offers better scalability and better reliability on server failover. The problem is that the server no longer guarantees security; a bad client can overwrite data for other clients, or hackers or malicious/viral code can access whole disks.

Eliminating the security problem is nontrivial: Ddisks are dumb, low-level devices with no idea of permissions or even files. The solution is to make disks smarter. One proposed way of doing so is to use higher-level objects rather than block I/O, but block I/O has important advantages. It is simple and well understood, so people would like to keep it.

Achieving security with block I/O is possible. The naïve way is to store with each block the owner, group, and mode. However, that list can grow quite large and is tied to particular OSes. Capability-based security is the better alternative: the server provides a capability, and then the client passes the capability to the disk with a write request. The Snapdragon system uses capabilities that contain a block range, permissions (r or r/w), and a cryptographic signature. These are checkable by relatively simple disk hardware.

The cryptographic system used is Message authentication codes (MAC); these are like digital signatures but are short and easy to compute, and use a shared key rather than a public key. The detailed protocol was originally proposed for the NASD system in 1997. In order to allow capability revocation, a revocation list is kept on the disk, sent directly from the server. This will increase in size over time, and is bounded by garbage collection; in part, this is achieved by the expiration time of capabilities, but large numbers of revocations within a short time can fill it. It is further limited by capability groups – the server can invalidate whole groups – while new capabilities can be issued by the server if a valid client gets rejected.

The system is able to avoid replay attacks; because timestamps/counters have drawbacks, the combination of bloom filters and epoch numbers are used. Bloom filters check for duplicate messages; the epoch number is a per-drive counter. The down side is a single rejected request per client per epoch change; to avoid this, the drive keeps a window of one old epoch's filter. As long as clients stay in regular contact, no messages should be rejected.

The resulting system provides a low-level block device, secure NAD devices, and a very high degree of flexibility and portability. The Snapdragon prototype is a client and server kernel-space implementation on Linux. Each disk is implemented using a simple program. The system was benchmarked using low-end hardware (400Mhz Intel Celeron-based machines for the client, server, and disks) over gigabit Ethernet; the resulting system is approximately 16% slower in throughput and 5% slower in latency than the system without security.

The actual protocol overhead is small; capabilities are a 116-byte block, and only 128Kb are required on the disk. Similarly, the software complexity on the disk is small – it only has to be able to compute the MAC, verify the capability, and check the bloom filters for a duplicate; as a result, it is suspected, but not verified, that it will be implementable in firmware for existing disk hardware.

In the Q&A section, the system drew praise from Garth Gibson, who went on to ask what the effect of a difference between block model and object model would be. The response was that minimizing change on disk was the motivation, as it was for the bloom filter.

Someone noted that the bloom filters offered only a probabilistic guarantee, and would have some false positives, and asked how this was handled. The system adds a nonce to each request and has a false positive rate of about 0.1%; while this may allow a denial of service attack, this could also be accomplished by bombarding the drive with any sort of invalid request.

WORK-IN-PROGRESS REPORTS

Summarized by Nate Edel

Parallel Extensions to the DAFS Protocol

Peter Corbett, Netapp

Peter Corbett discussed extending the DAFS (Direct Access File System) protocol to be used by a parallel file server. This provides excellent support for high performance computing with clustered and parallel clients, with support for fencing, shared key reservations, and locking, and for parallel IO semantics such as asynchronous I/O and completion groups. This extension to DAFS was implemented as a single tier file server with a clever client; the fastest implementation was in user space and opens doors for a DAFS client that understands parallel files.

World-wide Repository for I/O Trace Collection and Analysis Tools

Arnold Jones, Storage Networking Industry Association

Arnold Jones discussed the creation by SNIA of a repository of IO traces, workloads, collection, analysis tools and snapshots, for use by industry and academia, as well as a forum for the discussion of tool and trace problem solving, data collection, and similar issues. Participants will also be able to request traces on physical media. The design of the repository is in place, and they hope to be online by 7/1/2003.

http://www.snia.org/apps/IOTTA_Survey/register.php

The Zettabyte File System

Jeff Bonwick, Sun Microsystems

The Zettabyte file system (ZFS), developed at Sun, is to be released later this year. Bonwick noted that existing file systems have problems: no defense against data corruption, and lot of limits, such as on the number of files, maximum file sizes, and so forth. As a result, existing file systems are "excruciating to manage," between tools like fsck, many configuration files, and managing partitions, volumes, and the like. ZFS hopes to "end the suffering" by offering end to end data integrity, immense (128-bit) capacity, and very simple administration. All operations are copy-on-write transactions, with the on-disk state always valid. All data is checksummed to prevent silent data corruption, with support for self-healing in mirrored or RAID configurations. It also supports pooled storage models to eliminate partition management.

Running NFS over RDMA

Brent Callaghan, Sun Microsystems

Callaghan described NFS as implemented over Remote Direct Memory Access (RDMA). This is useful at 1gb/sec, and will probably be a critical requirement at 10gb/sec. It allows direct data placement (DDP), and has been implemented as a new transport on NFS, in parallel to UDP and TCP. He showed a method of doing DDP with RPC/NFS packets and benchmark numbers: 60Mb-sec peak with NFS/TCP, and 102MB-sec peak with NFS/RDMA. Further, CPU utilization is much lower with RDMA. There is a prototype running on Solaris, over gigabit Ethernet support for Infiniband is in progress.

Using SATF Scheduling in Real-Time Systems

Lars Reuther, Dresden University

Reuter began by pointing out that disk request scheduling is best handled at disk; on the other hand, the OS loses some control, which is undesirable for real-time systems, and onboard queue sizes on disk are small. His work examines request scheduling at driver level, especially using SATF, and asks whether it can be used for QoS guarantees. In doing so, this work measured the time between two requests – including the command overhead, seek and rotational delays, and the time to actually read a sector – with instrumentation on a SCSI device driver to determine the match between the model and measured values.

Benchmarks indicate that the external scheduler can match the performance of the disk internal scheduler on a slower disk – catch up with the internal scheduler; on a faster disk, total effective bandwidth is 12% lower but the faster queue benefits real-time guarantees. This shows that a system can do scheduling in software with QoS guarantees – allowing the tradeoff between queue size for throughput and QoS guarantees.

Using a Vector-Based Approach to Predict Performance of Distributed Storage Systems

Alexandra Federova, Harvard

Federova discussed using a vector-based approach to analyzing the performance of n-tier distributed systems. The problem domain is as follows: in an n-tier system – for example, Web servers to application servers to db servers – the impact of adding servers at a tier to improve a perceived bottleneck at that tier is difficult to test. At the same time, determining the impact in advance through simulation is tough to get right, and either approach takes time. Modelling this is difficult because of the complexity of the problem. Vector based modeling has been used for stand-alone systems, and Federova and her colleagues are looking into whether it can be used on distributed systems. The presentation briefly touched on how vector modeling is used for simpler systems, and some proposed rules for the composition of models for distributed systems.

Dumb Storage Devices Seek Smart Cluster Storage System Software

Christian Saether, Clustor.com

Saether discussed a mechanism for improving access to shared metadata in a cluster, based on earlier work on VAX clusters. The motivation for this work was the availability of cheap and dense multi-system hardware. It uses WAN protocols for data "right next door," with a new access layer for transactional updating of shared storage. Data objects are mapped for fault tolerance and performance, and write-ahead logging is used when there is no contention for data. The system is implemented at the kernel level, above the disk driver, "like an LVM," below the buffer cache. Modifications are made via transactions with nested redo and undo operations, and using a distributed lock manager to maintain data coherency.

The Storage Transport Protocol

Pat Shuff, Texas A&M University

Shuff discussed a proposed solution to the problem of how to use excess disk on campus, without central administration, on a variety of OS platforms. Their group estimated that the campus had approximately 200 terabytes of excess storage "lying around" – as compared to 2 terabytes of online storage for students and faculty. Existing mechanisms do not offer the ability to take advantage of excess space on unrelated systems or to find existing redundancy to use as backups.

Existing partial solutions include network backup and rsync, file-system level sharing, and block level sharing, but these all have some combination of cost, management, portability, or scalability issues. As an alternative, Shuff's group is working on a new storage protocol, to be implemented under the vnode layer or under the Windows virtual disk interface, which will act as an intelligent manager to handle variable locations for data. They are working on a protocol for network access accounting for security and automation which should work with existing file systems automatically. *http://people.tamu.edu/~pshuff/*

Testing for Distributed Filesystems

Richard Hedges, Lawrence Livermore National Labs

Hedges started by discussing briefly the need for and scope of very high performance computer clusters at LLNL; one large cluster supports up to approximately 100 TFLOPS, with 100 gigabytes per second IO throughput to a single parallel app. The team at LLNL works with other projects, including the Lustre filesystem – a collaboration between the three big DOE labs and industry (see *http://www.lustre.org/*).

There were several testing methods, including both traditional serial testing with readily available tools such as fsx, iozone, bonnie++, and the posix verification suite, and cluster validation testing.

One set of this testing is done using the IOR code, which was recently rewritten, designed as peak-performance throughput test for supercomputing data patterns. It is used as heavy IO load testing for parallel file systems, and is good for modeling data patterns, acceptance testing, and development activities.

Another tool used is Simul, which is an MPI-coordinated suite of system calls and library function calls, accessing a single file up to thousands of times or thousands of different files. It tests only minimal data transfer but high instantaneous load, and is used as a race-condition finder and for testing massive serialization.

For for information, see *http://www.llnl.gov/icc/lc/siop/*

Clusterfile: A Parallel File System

Florin Isaila, University of Karlsruhe, Germany

Isaila discussed technical issues with different types of parallelism: logical parallelism consists of multiple compute nodes accessing a file system, and physical parallelism consists of striping of data across multiple disks. The main problem this leads to is a poor match between the two types of parallelism. The proposed solution is a shared data representation.

A model was found in the PARADIGM compiler, which has been extended to their system for data representation. This is implemented using the physical partition of files into subfiles, and logical partitioning into views. Directions include implementing collective IO, disk directed IO inside the file system. The system can detect matching physical and logical distributions.

A second area is cooperative caching, with the cooperation of IO nodes and compute nodes' buffer caches as a remote disk driver for Linux. Using this, a node can fetch remote blocks into the local buffer cache. The policy to do so is downloadable and highly flexible. Two possible policies have been implemented so far.

Decentralized Recovery for Survivable Storage Systems

Ted Wong, CMU

Ted Wong discussed research into the problem of putting data objects on a storage server, intending them to be retrievable 5, 10, or 20 years hence with confidence and privacy. The goals of this work are longterm availability and confidentiality; the method is to distribute data with (m,n) threshold sequence sharing techniques. These work by splitting the data into n devices, and the threshold sequence techniques mean that only m pieces must be available to recover the data: up to $n–m$ failures are OK, and to compromise the overall data object, at least m parts must be compromised. Over time servers will fail, and there is a need to be able to recover from failures or compromised servers. The proposed technique is verifiable secret redistribution for threshold shared data; the system would use a witness value to prove possible reconstruction. To do this, the original shares split into subshares, and there is a broadcast protocol for share and subshare witnesses. For more information see *http://www.cs.cmu.edu/~tmwong/research/*

Federation of Local File System Data into a Shared-Disk Cluster File System

Anjali Prakash, Johns Hopkins University

Prakash discussed a system for "hassle-free data management" in a cluster file system (IBM Storage Tank). The goal is that it be easy to set up seamlessly integrate existing data, and allow for incremental migration of existing data into the cluster. The specific requirement was to add online access to local file system data to the cluster file system. Whether to migrate that data or not is a management decision.

Federated DAFS: Scalable Cluster-based Direct Access File Servers

Murali Rangarajan, Rutgers

Rangarajan described the design and implementation of a portable user-level DAFS implementation, for use in a federation of DAFS servers using memory-to-memory communication. The DAFS client and server in user space share a virtual interface architecture for communications; DAFS calls are translated to RPC on the server, using Berkeley SEDA. The system is implemented on Linux, FreeBSD, and Solaris. Compared to Harvard kernel-based DAFS, overall performance is close, with slightly more slowdown at with higher file sizes.

Synthetic IO Workload Generation Based on RS Plots

Junkil Ryu, POSTECH Korea

Ryu discussed Synthetic IO workload generation based on RS plots, a mechanism intended to generate a synthetic workload statistically equal to real traces.

Transparent Page Cache Coherence Support for Linux-based Stackable File Systems

Manish Prasad, Stony Brook University

Prasad discussed issues with VFS stacking, giving the example of a user process accessing an encryption file system built over ext2. Because, in Linux, file read and write is purely through page cache, there is a high risk of inconsistency in a stacked VFS environment: for example, reads may occur from an upper level, writes to a lower one. The existing stackable layer was modified to be centralized-cache-manager aware, trying to support native filesystems non-intrusively. This was set up to figure out stack order, detect and intercept calls such as `write()` and `sync()`, and then resolve them by invalidating (rather than updating) stale caches. Some future directions include performance evaluation, extension to work with the dentry and inode caches, and integration with network file systems.

SSM: A Self-Tuning, Self-Protecting, Self-Healing Session State Management Layer

Benjamin Ling

Ling defined a session as a period of interaction between user and application, and state is the temporary data which has to persist during the course of the session. He gave the example of a customer signup for a brokerage account. In a typical installation, this would be a database application stored on a standard file system, such as Netapp filer or similar. To improve performance, in-memory replication could be used, but that adds state to the middle tier, and performance is then coupled with uneven distribution of load after a failure. The SSM exploits properties of session state to separate it from the application servers (stubs) and state servers (bricks, which store state in parallel in memory) using a "write to many, wait for few" technique. There is a windowing mechanism (similar to TCP) for stubs to track bricks, and self-healing to recover from errors. There is a prototype, written in Java, running on the UC Berkeley Millennium cluster.

Decoupled Storage: Free the Replicas!

Andy Huang, Stanford

Huang discussed ongoing work at Stanford intended to reduce the cost of persistent state in Internet services with the goal of making managing state very simple, much like stateless front ends and app servers. This is done using a separate state store for non-transactional data. It uses a hash table API, and uses quorums to simplify recovery and keep data available throughout. Updates are handled by broadcasting a message to the replicas and then waiting for majority reply. On a read, the system accepts the reply with the most recent timestamp, and then writes back the new data to all out-of-date replicates. An initial prototype has been implemented.

StoRM: Storage Resource Management

Sandeep Gopisetty, IBM Almaden

Gopisetty noted that the cost of storage management often exceeds the cost of physical hardware, and that a typical heterogenous environment will have various administrative tools that may or may not be interoperable. His group at IBM is developing an enterprise systems management product, distinct from the existing Tivoli products (SAN viewer and data viewer).

The product is intended to manage the complete storage life cycle in several phases: identifying data storage assets, evaluating data in terms of priorities and storage problems, controlling policy and automation, and predicting usage and growth trends. This uses what they call an "autonomic architecture" which supports self-configuration, optimization, correction, and healing. They are engaged in research on automated provisioning for optimal use of resources, modeling for dependability, reliability, and performance, and have stated a goal of developing a policy-based architecture

Scalability of NFSv4 Next Steps

Dean Hildebrandt, CITI at the University of Michigan

Hildebrandt discussed the scalability of the upcoming NFSv4 protocol, in work funded by ASCI. He noted that NFSv4 is stateful on the file server for open, lock, and delegation operations. Questions related to scalability include how to share an object among multiple NFSv4 servers, or, more generally, how to share state. Their proposal was to implement a division between a state server and a data server, and then to extend the NFSv4 redirection mechanism to handle relocating individual files. The process flow would be for a client to mount onto the state server; open requests would go there, and then read or write is relocated to the data server via load balancing, with state copied to data servers as needed.

Is Parity-Protected RAID Obsolete?

Eran Gabber, AT&T

Gabber noted that disk capacity increases at 80% per year, while access time improves much more slowly, at about 12% per year. With the bottleneck of IO rate, not capacity, parity protected RAID has undesirable performance characteristics – 4 I/Os for every write, 2 if everything is in cache, while mirroring always only uses 2 I/Os. With disks that are so large, why bother with RAID?

This doesn't apply in all cases: for read-mostly data, where there are few writes, there is little write penalty. And for cases where the absolute lowest latency is necessary, the need for high-end disk devices may mean that customers cannot afford to replicate; similarly, where the absolute lowest cost is an issue, customers may also not be able to afford to replicate. Another interesting question is, how does MEMS fit? "But for the common case, watch the trend."

INVITED SESSION: ENTERPRISE STORAGE: THE NEXT DECADE

David Black, EMC; Garth Gibson, Panasas; and Steve Kleiman, Network Appliance

Summarized by Scott Banachowski

David Black started the panel discussion by noting that in the future, people will be the scarce resource in storage systems, because any headway in management scaling is instantly consumed by increased capacity. We must address the problem by changing what must be managed. Black mentioned approaches such as content-addressed storage and fixed-content data. These approaches not only solve some of the performance problems, but also change management problems, by requiring no directories or hierarchical namespaces.

Black outlined the "mystery meat" analogy: identifying data that was stored "in the freezer" for a long time. Grid researchers have started working on the problem of tracking and locating data sets, focusing their attention onto the content of the data rather than where it's stored.

Black reviewed some emerging technologies such as iSCSI and storage bricks, noting that how they will be used is unpredictable, as the innovation of early adopter markets dictates their future use. He concluded his segment by noting that the interesting problem in enterprise storage is no longer performance, but robustness.

Garth Gibson noted that most of the great ideas generated in the '90s still show no value to customers. Now that we live in leaner times, cost-effectiveness is high priority. For the rest of his talk, Gibson described the ideas that he predicts will survive in the following decade.

A trend toward simpler administration will survive, as it leads to cost-effectiveness. The most cost-effective mainframe computer is a cluster, and Gibson believes this is true of storage systems as well, as storage clusters leverage commodity storage and connectivity products. Other notable survivors are NAS and SAN, which are converging as new systems mix and match these storage network infrastructures to improve performance and manageability. Separating the control paths from data and asymmetrically accessing data by exposing parallelism will lead to better performance due to increased data bandwidth, offloaded metadata services, and fewer bottlenecks. Gibson explained that he liked object devices because the device encapsulates metadata, and clients don't need to be trusted when servers do authentication.

Gibson described a direction for enterprise storage that includes changing policy and namespace management. Rule-based policy is the rage, but making it work requires a body of well-understood expertise on using policies correctly, otherwise administrators will be cleaning up the mess left by misbehaving AI algorithms. The direction for namespace is toward search-engine-like interfaces for data access. Gibson concluded that in this decade, enterprise storage must deliver the research of the previous decade in order to cope with increasing scale and bandwidth demands.

Steve Kleiman focused on the problems of the next few years: supporting access to petabytes of data in geographically dispersed locations with thousands of users and nodes, and running diverse applications. Further complicating the systems, data will have different availability and recovery requirements, as well as different access patterns.

Kleiman provided a review of the evolution of enterprise architectures from their beginning as proprietary networks to wide-area large-scale enterprise data infrastructures. Driving this evolution is reduced cost of fast storage links and high volume, reliable disk systems. The problem isn't the technology but the management, especially considering the amount of geographic dispersion between data centers.

In order to increase manageability, Kleiman suggests virtualization of devices, elimination or drastic simplification of existing paradigms, and unification of existing enterprise storage solutions. Only by changing the paradigms and allowing the new technologies to enable new strategies will we solve our main problem of data management.

Jeff Chase offered a quick viewpoint before opening the floor for discussion. He noted that because interfaces for storage systems exist at different levels, people have different views of the meaning of convergence. Chase warned that this is leading us down a road that repeats problems facing cluster computing research in the '90s. During the discussion period, the speakers mostly reiterated points made in their talks. The liveliest part came when someone asserted that policy-based management is "evil" and "foolish." The systems are much too complex, with many contradicting rules for different scenarios, so that automated management will lead to fiasco. The panel agreed that policies are difficult to define and will never replace administrators.

SESSION: MEASURING THE TECHNOLOGY

Summary by Nate Edel

Modeling Hard-Disk Power Consumption

John Zedlewski, Sumeet Sobti, Nitin Garg, and Fengzhou Zheng, Princeton University; Arvind Krishnamurthy, Yale University; and Randolph Wang, Princeton University

Sumeet Sobti discussed a disk simulator developed at Princeton which gives an estimate of how much energy the disk is likely to consume, given a disk IO trace and a description of the disk. The motivating application is to determine the impact of file system attributes on energy consumption. Locality, burstiness, and the type and number of requests are all key factors; as such, power consumption will be based on user workload and the file system parameters.

Data layout policies, asynchrony, data layout, and background reorganization are all possible parameters, and in total are a huge design space to explore. The original goal was comprehensiveness, which proved very time consuming – simulator speed was key.

A flaw in many existing models is that disks don't consume power at a constant rate. For example, the IBM microdrive varies by 20–25% during active periods, and by a factor of 10 from idle to active. As such, a coarse-grained simulator is not adequate.

The team developed a fast and fine-grained disk power simulator, which worked well for the two disks it was tested with – and evaluated it against coarse-grained power models. The architecture of the simulator was in two parts. The simulator itself is based on DiskSim with an added Energy Simulator model. The second part is an automatic parameter extractor, which builds on the existing performance parameters extraction.

The energy simulator calculated total energy, which is in turn composed of of active energy states – seek/rotating/reading/writing – and idle states – low power modes and transitions. These are estimated via DiskSim to gather statistics about disk stages, with seek energy for example, determined by seek distance; rotate/read/write determined by constant use differing per activity. A table is used to approximate the behavior of available low-power modes and the transitions between them. Power values are extracted by the combination of hardware and software, tested on a 30ms basis. This is too coarse-grained to get individual operations, so they are instead spread across longer traces and then statistically determined.

Storage over IP: When Does Hardware Support Help?

Prasenjit Sarkar, Sandeep Uttamchandani, and Kaladhar Voruganti, IBM Almaden Research Center

Prasenjit Sarkar began by distinguishing Storage over IP from conventional SAN systems: although in both cases, storage is a service over the IP network, in conventional SANs servers are attached to storage over a specialized SAN network, while with IP SAN, a combined gigabit IP network is used by both servers and storage systems.

IP SAN implementations are flexible, with three common approaches: a software-only implementation with a generic network adapter (HBA), an adapter which implements a TCP Offload Engine (TOE) which supports the TCP/IP network stack on the network adapter, and an intelligent-HBA approach where both the IP storage protocol and TCP are implemented on the adapter.

The down sides to a software-only approach are the TCP copy overhead, multiple interrupts per data transfer, and high communications overhead; variants such as jumbo-frames and zero-copy TCP can ameliorate these somewhat.

A TOE adapter uses DMA to stream data to the network adapter, which implements TCP/IP; this reduces the overhead on the host and results in one interrupt per data transfer, reducing communications overhead, although the TCP copy overhead remains.

An intelligent HBA approach uses a single DMA to the HBA and removes both the storage protocol (iSCSI) and TCP overheads from the host, with one irq per data transfer, and has the lowest communications overhead.

MORE THAN AN INTERFACE – SCSI VS. ATA

Dave Anderson, Jim Dykes, and Erik Riedel, Seagate Research

Eric Riedel aimed to dispel myths and confusion about hard drives. He noted that while many consumers and businesses divided up the market by interface between SCSI and ATA, the market segmentation as it was seen by the disk drive industry was quite different. The two main segments he went on to discuss are drives for personal storage (PS), used in desktop systems and low-end servers, and drives for enterprise storage (ES), used in servers, high-end workstations, and drive arrays. He also noted that there are separate market segments for mobile drives, such as those used in notebook computers, and for drives for consumer devices/appliances, but that these would be a topic for future discussion. He also noted the persistent myth that drives are built along one assembly line, tested at the end, and the "bad ones" get ATA controller boards and the better ones get SCSI.

He compared two Seagate drives, a 10k RPM Cheetah vs. a 7200 RPM Barracuda, to show the differences between an enterprise drive and a lower-end model. The Cheetah had a smaller platter (84mm vs. 95mm), a much larger actuator assembly to reduce seek times, and a more rigid case structure for durability and vibration-proofing, which he noted were only three major factors, out of many. He also briefly noted differences from a 15k RPM Cheetah, which has a 65mm platter for still lower mass and shorter seeks, but at the expense of lower capacity.

Seek times are much more aggressive on enterprise drives, with the level of separation increasing; the rate of improvement is low on personal drives, and somewhat quicker on enterprise. Seek time is very sensitive to both the mechanics and signal processing, and thus costly. Sensitivity to external vibration is also a factor; the rotation of one drive can affect neighbors, and while enterprise drives are designed to block those vibrations, personal drives are not. This can have a negative effect on performance.

The talk closed with a comparison of the direct performance impact of certain design choices, comparing two IBM drives; area density and platter size were large on the personal drive. RPM was higher on the enterprise drive, overall resulting in a slight bandwidth advantage to the personal drive. However, an enterprise drive of comparable generation has a higher bandwidth (53MB/s vs 37MB/s), while the changes (more platters, higher RPM) result in a higher cost.