PARALLEL DATA LABORATORY
CARNEGIE MELLON UNIVERSITY

THE
PDL Packet
THE NEWSLETTER ON PDL ACTIVITIES AND EVENTS • FALL 2003
http://www.pdl.cmu.edu/

AN INFORMAL PUBLICATION FROM ACADEMIA'S PREMIERE STORAGE SYSTEMS RESEARCH CENTER DEVOTED TO ADVANCING THE STATE OF THE ART IN STORAGE SYSTEMS AND INFORMATION INFRASTRUCTURES.

CONTENTS

PDL CONSORTIUM MEMBERS

EMC Corporation
Hewlett-Packard Labs
Hitachi, Ltd.
IBM Corporation
Intel Corporation
Microsoft Corporation
Network Appliance
Panasas, Inc.
Oracle Corporation
Seagate Technology
Sun Microsystems
Veritas Software Corporation

Self-* Storage

Greg Ganger & Joan Digney

Human administration of storage systems is a large and growing issue in modern IT infrastructures. We are exploring new storage architectures that integrate automated management functions and simplify the human administrative task. Self-* storage systems (pronounced "self-star"— a play on the unix shell wild-card character) are self-configuring, self-organizing, self-tuning, self-healing, self-managing systems of storage bricks. Borrowing organizational ideas from corporate structure and technologies from AI and control systems, self-* storage should simplify storage administration, reduce system cost, increase system robustness, and simplify system construction.

The Self-* Storage Architecture

Dramatic simplification of storage administration requires that associated functionalities be designed into the storage system from the start and integrated throughout the design. System components must continually collect information about ongoing tasks and regular evaluation of configuration and workload partitioning must occur, all within the context of high-level administrator guidance. The self-* storage project is designing an architecture from a clean slate to explore such integration. The high-level system architecture is shown in Figure 1.

The Brick-based Storage Infrastructure. We envision self-* storage systems composed of networked "intelligent" storage bricks, each consisting of CPU(s), RAM, and a number of disks; example brick designs provide 0.5-5 TB of storage with moderate levels of reliability, availability, and performance. Each storage brick (a "worker") self-tunes and adapts its operation to its observed workload and assigned goals. Data redundancy across and within storage bricks provides fault tolerance and creates opportunities for automated reconfiguration to handle many problems. Out-of-band supervisory processes assign datasets and goals to workers, track

Continued on page 12

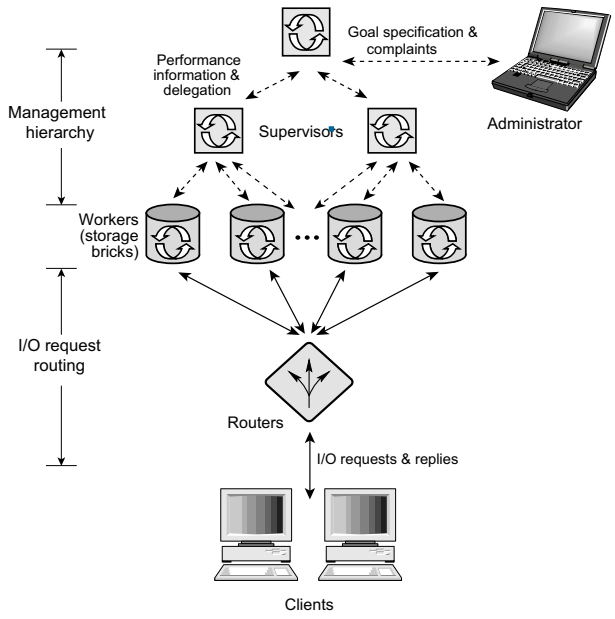


Figure 1: Architecture of self-* storage. At the top of the diagram is the management hierarchy, concerned with the distribution of goals and the delegation of storage responsibilities from the system administrator down to the individual worker devices. The path of I/O requests from clients, through routing nodes, to the workers for service is shown at the bottom. Note that the management infrastructure is logically independent of the I/O request path, and that the routing is logically independent of the clients and the workers.

FROM THE DIRECTOR'S CHAIR

Greg Ganger

Hello from fabulous Pittsburgh!

2003 has been an exciting year in the Parallel Data Lab, with a number of projects maturing and contributing to a broad, new research initiative in automated storage administration. Along the way, two faculty and several new students joined the Lab, several students spent summers with PDL Consortium companies, and several students won new Fellowships (one each from Intel, NSF, and DoD).

The PDL continues to pursue a broad array of storage systems research, from the underlying devices to the applications that rely on storage. The past year brought excellent progress in existing projects and the initiation of exciting new ones. Let me highlight a few things.

The biggest development has been the initiation of a major new project called Self-* Storage, which explores the design and implementation of self-organizing, self-configuring, self-tuning, self-healing, self-managing systems of storage bricks. For years, PDL Retreat attendees have been pushing us to attack "storage management of large installations," and this project is our response. With generous equipment donations from the PDL Consortium companies, we hope to put together and maintain 100s of terabytes of storage to be used by ourselves and other CMU researchers (e.g., in data mining, astronomy, and scientific visualization). Of course, the research challenge is to make the system almost entirely self-*, so as to avoid the traditional costs of storage administration —deploying a real system will allow us to test our ideas in practice. Towards this end, we are designing the Self-* Storage architecture with a clean-slate, integrating management functions throughout. In realizing the design, we are combining a number of recent and ongoing PDL projects (e.g., freeblock scheduling, PASIS, and self-securing storage) and ramping up efforts on new challenges such as block-box device and workload modeling, automated decision making, and automated diagnosis.

PDL's push into database storage management has also produced an exciting new project: Fates. A Fates-based database storage manager transparently exploits select knowledge of the underlying storage infrastructure to automatically achieve robust, tuned performance. As in Greek mythology, there are three Fates: Atropos, Clotho, and Lachesis. The Atropos volume manager stripes data across disks based on track boundaries and exposes aspects of the resulting parallelism. The Lachesis storage manager utilizes track boundary and parallelism information to match database structures and access patterns to the underlying storage. The Clotho dynamic page layout allows retrieval of just the desired table attributes, eliminating unnecessary I/O and wasted main memory. Altogether, the three Fates components simplify database administration, increase performance, and avoid performance fluctuations due to query interference. Other database systems projects are exploring benchmarking techniques, automatic physical database design, and new software architectures for robust load management.

The self-securing devices project has made great strides. Highlighted in 2002 by several news organizations, this project adapts medieval warfare notions to the defense of networked computing infrastructures. In a nutshell, devices are augmented with relevant security functionality and made intrusion-independent from client OSes and other devices. This architecture makes systems more intrusion-tolerant and more manageable when under attack. The self-securing devices vision has brought with it many interesting challenges and a healthy

THE PDL PACKET

The Parallel Data Laboratory
School of Computer Science
Department of ECE
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
VOICE 412•268•6716
FAX 412•268•3010

PUBLISHER
Greg Ganger

EDITOR
Joan Digney

The PDL Packet is published once per year and provided to members of the PDL Consortium. Copies are given to other researchers in industry and academia as well. A pdf version resides in the Publications section of the PDL Web pages and may be freely distributed. Contributions are welcome.

COVER ILLUSTRATION

Skibo Castle and the lands that com-prise its estate are located in the Kyle of Sutherland in the northeastern part of Scotland. Both 'Skibo' and 'Sutherland' are names whose roots are from Old Norse, the language spoken by the Vikings who began washing ashore reg-ularly in the late ninth century. The word '*Skibo*' fascinates etymologists, who are unable to agree on its original meaning. All agree that '*bo*' is the Old Norse for '*land*' or '*place*.' But they argue whether '*ski*' means '*ships*' or '*peace*' or '*fairy hill*.'

Although the earliest version of Skibo seems to be lost in the mists of time, it was most likely some kind of fortified building erected by the Norsemen. The present-day castle was built by a bishop of the Roman Catholic Church. Andrew Carnegie, after making his fortune, bought it in 1898 to serve as his sum-mer home. In 1980, his daughter, Mar-garet, donated Skibo to a trust that later sold the estate. It is presently being run as a luxury hotel.

CONTACT US

WEB PAGES

PDL Home: http://www.pdl.cmu.edu/

Please see our web pages at
http://www.pdl.cmu.edu/PEOPLE/
for further contact information.

FACULTY

Greg Ganger (director)
412•268•1297
ganger@ece.cmu.edu
Anastassia Ailamaki
natassa@cs.cmu.edu
Anthony Brockwell
abrock@stat.cmu.edu
Christos Faloutsos
christos@cs.cmu.edu
Garth Gibson
garth@cs.cmu.edu
Seth Goldstein
seth@cs.cmu.edu
Mor Harchol-Balter
harchol@cs.cmu.edu
Chris Long
chrisl@cs.cmu.edu
Todd Mowry
tcm@cs.cmu.edu
Adrian Perrig
adrian@ece.cmu.edu
Mike Reiter
reiter@cmu.edu
Mahadev Satyanarayanan
satya@cs.cmu.edu
Srinivasan Seshan
srini@cmu.edu
Dawn Song
dawnsong@ece.cmu.edu
Chenxi Wang
chenxi@ece.cmu.edu
Hui Zhang
hui.zhang@cs.cmu.edu

STAFF MEMBERS

Karen Lindenfelser
(pdl business administrator)
412•268•6716
karen@ece.cmu.edu
Stan Bielski
Mike Bigrigg
John Bucy
Joan Digney
Gregg Economou
Ken Tew
Linda Whipkey

STUDENTS

Michael Abd-El-Malek
Mukesh Agrawal
Kinman Au
Shimin Chen
Garth Goodson
John Linwood Griffin
Stavros Harizopoulos
James Hendricks
Andrew Klosterman
Chris Lumb
Amit Manjhi
Michael Mesnier
Jim Newsome
Spiros Papadimitriou
Stratos Papadomanolakis
Adam Pennington
Ginger Perng
David Petrou
Brandon Salmon
Raja Sambasivan
Jiri Schindler
Steve Schlosser
Minglong Shao
Vlad Shkapenyuk
Shafeeq Sinnamohideen
Craig Soules
John Strunk
Eno Thereska
Niraj Tolia
Mengzhi Wang
Ted Wong
Jay Wylie
Shuheng Zhou

FROM THE DIRECTOR'S CHAIR

source of funding. We have continued our exploration of how to build and exploit network interface software for containing compromised client systems, and explored in depth a new way of detecting intruders: storage-based intrusion detection. Storage devices are uniquely positioned to spot some common intruder actions (e.g., scrubbing audit logs and inserting backdoors), making this an exciting new concept.

Other ongoing PDL projects are also producing cool results. For example, the PASIS project has produced a family of efficient and scalable consistency protocols that support a wide range of fault models that require no change to the server infrastructure. A new project has begun exploring the use of context information for assigning attributes to files to enable effective attribute-based searches. We have built a working freeblock scheduling system and API in FreeBSD, and are using it for research activities such as continuous disk reorganization; it will also be a core component of each self-* storage brick. Decentralized caching is being explored in several contexts, including wide-area systems that opportunistically utilize CAS overlays and NFS clusters that dynamically shed load by replicating read-only files. This newsletter and the PDL website offer more details and additional research highlights.

On the education front: This Spring, for the third time, we offered our new storage systems course to undergraduates and masters students at Carnegie Mellon. Topics span the design, implementation, and use of storage systems, from the characteristics and operation of individual storage devices to the OS, database, and networking techniques involved in tying them together and making them useful. The base lectures were complemented by real-world expertise generously shared by 8 guest speakers from industry, including several members of the SNIA Technical Council. We continue to work on the book, and several other schools have already picked up and started teaching similar storage systems courses. We view providing storage systems education as critical to the field's future; stay tuned.

I'm always overwhelmed by the accomplishments of the PDL students and staff, and it's a pleasure to work with them. As always, their accomplishments point at great things to come.

NEW PDL FACULTY

Anthony Brockwell

Dr. Anthony Brockwell, Assistant Professor in the Dept. of Statistics at Carnegie Mellon, received his Ph.D. in Statistics and Electrical Engineering from the University of Melbourne in 1998. He joined the PDL in June to collaborate on our Self-* Storage project. Dr. Brockwell's research interests are primarily in the areas of dynamical systems and Bayesian computational methods. He is particularly interested in the analysis and control of nonlinear and non-Gaussian systems, and in associated computational methods such as particle filtering and Markov chain Monte Carlo. Dr. Brockwell has published articles in such journals as SIAM Journal on Control and Optimiza-

Continued on page 4

YEAR IN REVIEW

October 2003

- 11th Annual PDL Retreat and Workshop.

September 2003

- Jiri Schindler presented two papers at VLDB in Berlin: "Lachesis: Robust Database Storage Management Based on Device-specific Performance Characteristics," and "Matching Database Access Patterns to Storage Characteristics," which won the Ph.D. Workshop's Best Paper Award.
- Christos Faloutsos gave the keynote talk "Next Generation Data Mining Tools: Power laws and self-similarity for graphs, streams and traditional data" at ECML, Dubrovnik, Croatia.
- Spiros Papadimitriou presented "Adaptive, Hands-Off Stream Mining" at VLDB.

August 2003

- Greg visited Intel in Portland, OR, to present Self-* Storage.
- Adam Pennington presented "Storage-based Intrusion Detection" at USENIX Security '03 in Washington, DC.
- Chris Long organized a Birds of a Feather session on "Security and Usability" at the USENIX Security Symposium in Washington, DC.
- Christos Faloutsos gave a tutorial on "Mining Time Series Data" at ICML 2003, Washington DC.
- Jiri Schindler successfully defended his PhD dissertation titled "Matching Application Access Patterns to Storage Device Characteristics."

July 2003

- Greg visited HP, Microsoft and Veritas in CA and presented Self-* Storage.

June 2003

- Brandon Salmon presented "A Two-Tiered Software Architecture for Automated Tuning of Disk Layouts" at the AASMS Workshop in San Diego, CA. Greg also attended.
- Christos Faloutsos presented a tutorial on "Internet Research meets Data Mining: Current Knowledge and New Tools" at SIGMETRICS 2003, San Diego.
- Niraj Tolia attended the Usenix Annual Technical Conference in San Antonio, TX and presented the paper "Opportunistic Use of Content Addressable Storage for Distributed File Systems."

May 2003

- Craig Soules spoke on "Why Can't I Find My Files? New Methods for Automating Attribute Assignment" at HotOS in Lihue, HI. Greg also attended.
- Greg attended the AFSOR program review in Colorado and presented Self-Securing Devices
- Adam Pennington spent the summer interning at Seagate in Pittsburgh; Brandon Salmon interned at Microsoft Research, and Craig Soules interned with HP Labs.

April 2003

- Craig Soules presented "Metadata Efficiency in Versioning File Systems" at FAST 03; Greg and many other PDL students also attended.
- Fifth annual PDL Industry Visit Day.
- Chris Long co-organized the Workshop on Human-Computer

Continued on page 19

NEW PDL FACULTY

Continued from page 3

tion, Journal of Time Series Analysis and Journal of Computational and Graphical Statistics.

Mahadev Satyanarayanan

At long last, Satya has joined the PDL! Long renowned as a leading researcher in distributed file systems, Satya's research has given us many of the principles on which today's and tomorrow's files systems are based.

Key ideas from the Coda file system, which supports disconnected and bandwidth-adaptive operation, have been incorporated by Microsoft into the IntelliMirror component of Windows. Another outcome of his research is Odyssey, a set of open-source operating system extensions for enabling mobile applications to adapt to variation in critical resources such as bandwidth and energy. Coda and Odyssey are building blocks in Project Aura, a research initiative at Carnegie Mellon to build a distraction-free ubiquitous computing environment. Earlier, Satyanarayanan was a principal architect and implementor of the Andrew File System (AFS), which has been commercialized by IBM.

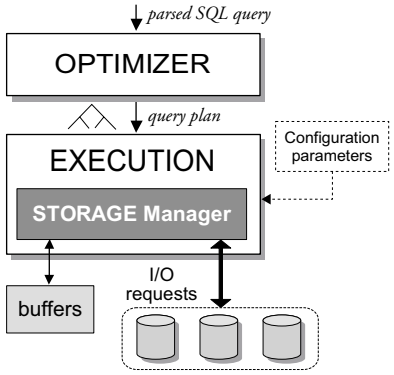
Dr. Satyanarayanan is the Carnegie Group Professor of Computer Science at Carnegie Mellon University. He is currently serving as the founding director of Intel Research Pittsburgh, which focuses on software systems for distributed data storage. He is the founding Editor-in-Chief of IEEE Pervasive Computing. He received his Ph.D. in Computer Science from Carnegie Mellon, after completing Bachelor's and Master's degrees from the Indian Institute of Technology, Madras. He is also a Fellow of the ACM and the IEEE.

Lachesis: Robust Database Storage Management Based on Device-specific Performance Characteristics

Schindler, Ailamaki & Ganger

VLDB 03, Berlin, Germany, Sept 9-12, 2003.

Database systems work hard to tune I/O performance, but do not always achieve the full performance potential of modern disk systems. Their abstracted view of storage components hides useful device-specific characteristics, such as disk track boundaries and advanced built-in firmware algorithms. This paper presents a new storage manager architecture, called Lachesis, that exploits and adapts to observable device-specific characteristics in order to achieve and sustain high performance. For DSS queries, Lachesis achieves I/O efficiency nearly equivalent to sequential streaming even in the presence of competing random I/O traffic. In addition, Lachesis simplifies manual configuration and restores the optimizer's assumptions about the relative costs of different access patterns expressed in query plans. Experiments using IBM DB2 I/O traces as well as a prototype implementation show that Lachesis improves standalone DSS performance by 10% on average. More importantly, when running concurrently with an on-line transaction processing (OLTP) workload, Lachesis improves DSS performance by up to 3X, while OLTP also exhibits a 7% speedup.



Query optimization and execution in a typical DBMS.

A Two-Tiered Software Architecture for Automated Tuning of Disk Layouts

Salmon, Thereska, Soules & Ganger

First Workshop on Algorithms and Architectures for Self-Managing Systems. In conjunction with Federated Computing Research Conference (FCRC). San Diego, CA. June 11, 2003.

Many heuristics have been developed for adapting on-disk data layouts to expected and observed workload characteristics. This paper describes a two-tiered software architecture for cleanly and extensibly combining such heuristics. In this architecture, each heuristic is implemented independently and an adaptive combiner merges their suggestions based on how well they work in the given environment. The result is a simpler and more robust system for automated tuning of disk layouts, and a useful blueprint for other complex tuning problems such as cache management, scheduling, data migration, and so forth.

Efficient Consistency for Erasure-coded Data via Versioning Servers

Goodson, Wylie, Ganger & Reiter

Carnegie Mellon University Technical Report CMU-CS-03-127, April 2003.

This paper describes the design, implementation and performance of a family of protocols for survivable, decentralized data storage. These protocols exploit storage-node versioning to efficiently achieve strong consistency semantics. These protocols allow erasure-codes to be used that achieve network and storage efficiency (and optionally data confidentiality in the face of server compromise). The protocol family is general in that its parameters accommodate a wide range of fault and timing assumptions, up to asynchrony and Byzantine faults of both storage-nodes and clients, with no changes to server implementation or client-server interface. Measurements of a prototype storage system using these protocols show that the protocol performs well under various system model assumptions, numbers of failures tolerated, and degrees of reader-writer concurrency.

A Human Organization Analogy for Self-* Systems

Strunk & Ganger

First Workshop on Algorithms and Architectures for Self-Managing Systems. In conjunction with Federated Computing Research Conference (FCRC). San Diego, CA. June 11, 2003.

The structure and operation of human organizations, such as corporations, offer useful insights to designers of self-* systems (a.k.a. self-managing or autonomic). Examples include worker/supervisor hierarchies, avoidance of micro-management, and complaint-based tuning. This paper explores the analogy, and describes the design of a self-* storage system that borrows from it.

Exposing and Exploiting Internal Parallelism in MEMS-based Storage

Schlosser, Schindler, Ailamaki & Ganger

Carnegie Mellon University Technical Report CMU-CS-03-125, March 2003.

MEMS-based storage has interesting access parallelism features. Specifically, subsets of a MEMStore's thousands of tips can be used in parallel, and the particular subset can be dynamically chosen. This paper describes how such access parallelism can be exposed to system software, with minimal changes to system in-

Continued on page 6

Continued from page 5

terfaces, and utilized cleanly for two classes of applications. First, background tasks can utilize unused parallelism to access media locations with no impact on foreground activity. Second, two-dimensional data structures, such as dense matrices and relational database tables, can be accessed in both row order and column order with maximum efficiency. With proper table layout, unwanted portions of a table can be skipped while scanning at full speed. Using simulation, we explore performance features of using this device parallelism for an example application from each class.

Data Page Layouts for Relational Databases on Deep Memory Hierarchies

Ailamaki, DeWitt & Hill

The VLDB Journal 11(3), 2002.

Relational database systems have traditionally optimized for I/O performance and organized records sequentially on disk pages using the N-ary Storage Model (NSM) (a.k.a., slotted pages). Recent research, however, indicates that cache utilization and performance is becoming increasingly important on modern platforms. In this paper, we first demonstrate that in-page data placement is the key to high cache performance and that NSM exhibits low cache utilization on modern platforms. Next, we propose a new data organization model called PAX (Partition Attributes Across), that significantly improves cache performance by grouping together all values of each attribute within each page. Because PAX only affects layout inside the pages, it incurs no storage penalty and does not affect I/O behavior. According to our experimental results (which were obtained without using any indices on the participating relations), when compared to NSM (a) PAX exhibits superior cache and memory bandwidth utilization, saving at least 75% of NSM's stall time due to data cache accesses, (b) range selection queries and updates on memory-resident relations execute 17-25% faster, and (c) TPC-H queries involving I/O execute 11-48% faster. Finally, we show that PAX performs well across different memory system designs.

Self-* Storage: Brick-based Storage with Automated Administration

Ganger, Strunk & Klosterman

Carnegie Mellon University Technical Report, CMU-CS-03-178, August 2003.

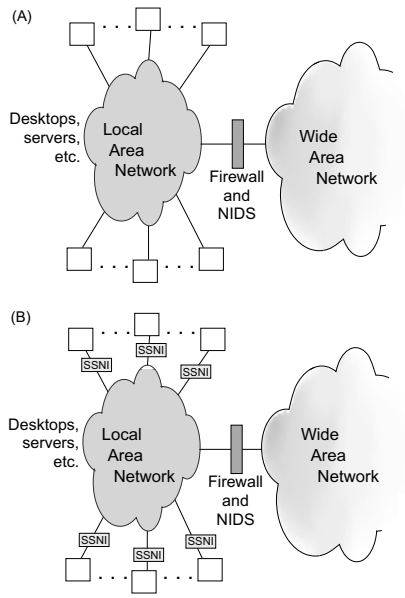
This white paper describes a new project exploring the design and implementation of "self-* storage systems:" self-organizing, self-configuring, self-tuning, self-healing, self-managing systems of storage bricks. Borrowing organizational ideas from corporate structure and automation technologies from AI and control systems, we hope to dramatically reduce the administrative burden currently faced by data center administrators. Further, compositions of lower cost components can be utilized, with available resources collectively used to achieve high levels of reliability, availability, and performance.

Finding and Containing Enemies Within the Walls with Self-securing Network Interfaces

Ganger, Economou & Bielski

Carnegie Mellon University Technical Report CMU-CS-03-109, January 2003.

Self-securing network interfaces (NIs) examine the packets that they move between network links and host software, looking for and potentially blocking malicious network activity. This paper describes how self-securing network interfaces can help administrators to identify and contain compromised machines within their intranet. By shadowing host state, self-securing NIs can better identify suspicious traffic originating from that host, including many explicitly designed to defeat network intrusion detection systems. With normalization and detection-triggered throttling, self-securing NIs can reduce the ability of compromised hosts to launch attacks on other systems inside (or outside) the intranet. We describe a prototype self-securing NI and example scanners for detecting such things as TTL abuse, fragmentation abuse, "SYN bomb" attacks, and random-propagation worms like Code-Red.



Self-securing network interfaces. (A) shows the conventional network security configuration, wherein a firewall and a NIDS protect LAN systems from some WAN attacks. (B) shows the addition of self-securing NIs, one for each LAN system.

Object-Based Storage

Mesnier, Ganger & Riedel

IEEE Communications Magazine, v.41 n.8 pp 84-90, August 2003.

Storage technology has enjoyed considerable growth since the first disk drive was introduced nearly 50 years ago, in part facilitated by the slow and steady evolution of storage interfaces

Continued on page 7

Continued from page 6

(SCSI and ATA/IDE). The stability of these interfaces has allowed continual advances in both storage devices and applications, without frequent changes to the standards. However, the interface ultimately determines the functionality supported by the devices, and current interfaces are holding system designers back. Storage technology has progressed to the point that a change in the device interface is needed. Object-based storage is an emerging standard designed to address this problem. In this article we describe object-based storage, stressing how it improves data sharing, security, and device intelligence. We also discuss some industry applications of object-based storage and academic research using objects as a foundation for building even more intelligent storage systems.

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

Soules & Ganger

Proceedings of the Ninth Workshop on Hot Topics in Operating systems, USENIX Association, May 2003.

This paper analyzes various algorithms for scheduling low priority disk drive tasks. The derived closed form solution is applicable to a class of greedy algorithms that includes a variety of background disk scanning applications. By paying close attention to many characteristics of modern disk drives, the analytical solutions achieve very high accuracy — the difference between the predicted response times and the measurements on two different disks is only 3% for all but one examined workload. This paper also proves a theorem which shows that background tasks implemented by greedy algorithms can be accomplished with very little seek penalty. Using greedy algorithm gives a 10% shorter response time for the foreground application requests and up to a 20% decrease in total background task run time compared to results from previously published techniques.

Storage-based Intrusion Detection: Watching Storage Activity For Suspicious Behavior

Pennington, Strunk, Griffin, Soules, Goodson & Ganger

12th USENIX Security Symposium, Washington, D.C., Aug 4-8, 2003. An early version is available as Carnegie Mellon University Technical Report CMU-CS-02-179, September 2002.

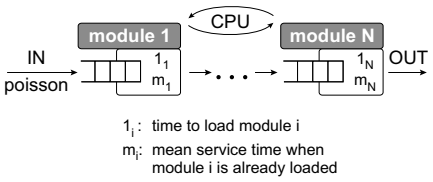
Storage-based intrusion detection allows storage systems to watch for data modifications characteristic of system in-trusions. This enables storage systems to spot several common intruder actions, such as adding backdoors, inserting Trojan horses, and tampering with audit logs. Further, an intrusion detection system (IDS) embedded in a storage device continues to operate even after client systems are compromised. This paper describes a number of specific warning signs visible at the storage interface. Examination of 18 real intrusion tools reveals that most (15) can be detected based on their changes to stored files. We describe and evaluate a prototype storage IDS, embedded in an NFS server, to demonstrate both feasibility and efficiency of storage-based intrusion detection. In particular, both the performance overhead and memory required (152 KB for 4730 rules) are minimal.

A Case for Staged Database Systems

Harizopoulos & Ailamaki

In proceedings of the First International Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, January 2003.

Traditional database system architectures face a rapidly evolving operating environment, where millions of users store and access terabytes of data. In order to cope with increasing demands for performance, high-end DBMS employ parallel processing techniques coupled with a plethora of sophisticated features. However, the widely adopted, work-centric, thread-parallel execution model entails several shortcomings that limit server performance when executing workloads with changing requirements. Moreover, the monolithic approach in DBMS software has lead to complex anddifficultto extend designs. This paper introduces a staged design for high-performance, evolvable DBMS that are easy to tune and maintain. We propose to break the database system into modules and to encapsulate them into self-contained stages connected to each other through queues. The staged, data-centric design remedies the weaknesses of modern DBMS by providing solutions at both a hardware and a software engineering level.



A production-line model for staged servers.

Adaptive, Hands-Off Stream Mining

Papadimitriou, Brockwell & Faloutsos

Carnegie Mellon University SCS Technical Report CMU-CS-02-205. Also published in Proceedings VLDB 03, Berlin, Germany, Sept 9-12, 2003.

Sensor devices and embedded processors are becoming ubiquitous, especially in measurement and monitoring applications. Automatic discovery of patterns and trends in the large volumes of such data is of paramount importance. The combination of rela-

Continued on page 16

DATABASE QUERY EXECUTION WITH FATES

Jiri Schindler, Minglong Shao, Steve Schlosser, Anastassia Ailamaki & Greg Ganger

Current database systems use data layouts that can exploit unique features of only one level of the memory hierarchy (cache/main memory or on-line storage). Such layouts optimize for the predominant access pattern of one workload (e.g., DSS), while trading off performance of another workload type (e.g., OLTP). Achieving efficient execution of different workloads without this trade-off or the need to manually re-tune the system for each workload type is still an unsolved problem. The "Fates" database system project answers this challenge.

The primary goal of the project is to achieve efficient execution of compound database workloads at all levels of a database system memory hierarchy. By leveraging unique characteristics of devices at each level, the Fates database system can automatically match query access patterns to the respective device characteristics, which eliminates the difficult and error-prone task of manual performance tuning.

Borrowing from the Greek mythology of The Three Fates–Clotho, Lachesis, and Atropos–who spin, measure, and cut the thread of life, the three components of our database system (bearing the Fates' respective names) establish proper abstractions in the database query execution engine. These abstractions cleanly separate the functionality of each component (described below) while allowing efficient query execution along the entire path through the database system.

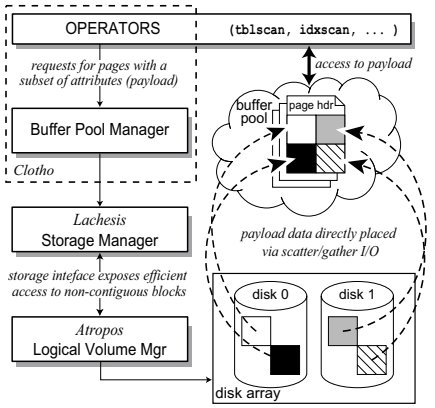


Figure 1: The Fates database system architecture. Efficient transformation of on-disk layout to in-memory page layout is achieved by DMA and scatter/gather I/O.

The main feature of the Fates database system is the decoupling of the in-memory data layout from the on-disk storage layout. Different data layouts at each memory hierarchy level can be tailored to leverage specific device characteristics at that level. An in-memory page layout can leverage L1/L2 cache characteristics, while another layout can leverage characteristics of storage devices. Additionally, the storage-device-specific layout also yields efficient I/O accesses to records for different query access patterns. Finally, this decoupling also provides flexibility in determining what data to request and keep around in main memory.

Traditional database systems are forced to fetch and store unnecessary data as an artifact of a chosen data layout. The Fates database system, on the other hand, can request, retrieve, and store just the needed data, catering to the needs of a specific query. This conserves storage device bandwidth, memory capacity, and avoids cache pollution—all of which improves query execution time.

Scatter/gather I/O facilitates efficient transformation from one layout into another and creates an organization amenable to the individual query needs on-the-fly. In addition to eliminating expensive data copies, these I/Os match explicit storage device characteristics, ensuring efficient execution at the storage device. Finally, thanks to proper abstractions established by each Fate, other database system components (e.g., the query optimizer or the locking manager) remain unchanged.

CLOTHO

Clotho ensures efficient query execution at the cache/main-memory level and figures at the inception of a request for particular data. When the data desired by a query is not found in the buffer pool, Clotho creates a skeleton of a page in a memory frame describing what data is needed and how to lay it out.

The page layout builds upon a cache-friendly layout, called PAX [1], which groups data into minipages, where each minipage contains the data of only a single attribute (table column). Aligning records on cache line boundaries within each minipage and taking advantage of cache prefetching logic improves the performance of scan operations without sacrificing full-record access.

Clotho adjusts the position and size of each minipage within the frame. It matches the minipage size to the (multiple of) block size of the storage device, provided by Atropos, and decides which minipages to store within a single page based on the needs of a given query. Hence, minipages within a single memory frame contain only the attributes needed by that query. The remaining attributes constituting the full record, but not needed by the query, are never requested.

The skeleton page inside the memory frame includes a header that lists the attributes and the range of records to be retrieved from the storage device and

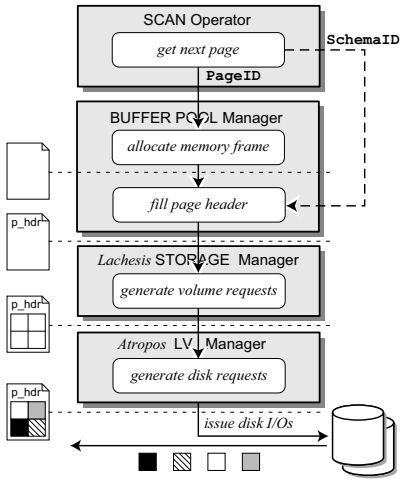


Figure 2: Operations performed by each component of the database system, showing the content of a single memory frame at each stage of a data request during query execution.

Continued on page 9

Continued from page 8

stored in that frame. Clotho marks the set of attributes needed by the query in the page header (i.e., the payload) and identifies the range of records to be put into the page. Hence, the page header serves as a request from Clotho to Lachesis to retrieve the desired data.

LACHESIS

The Lachesis database storage manager handles the mapping and access to minipages located within the LBNs of on-line storage devices. Utilizing storage device-provided performance characteristics and matching query access patterns (e.g., sequential scan) to these hints, Lachesis constructs efficient I/Os. It also sets scatter/gather I/O vectors that allow direct placement of individual minipages to proper memory frames without unnecessary memory copies.

Explicit relationships between individual logical blocks (LBNs), established by Atropos, allow Lachesis to devise a layout that groups together a set of related minipages. This grouping ensures that all attributes belonging to the same set of records can be accessed in parallel, while a particular attribute can be accessed with efficient sequential I/Os. The relationships between LBNs serve as hints that let Lachesis construct I/Os that Atropos can execute efficiently.

The layout and content of each minipage is transparent to both Lachesis and Atropos. Lachesis merely decides how to map each minipage to LBNs to be able to construct a batch of efficient I/Os. Atropos in turn, cuts these batches into individual disk I/Os comprising the exported logical volume. It does not care where the data will be placed in memory; this is decided by the scatter/gather I/O vectors set up by Lachesis.

ATROPOS

Atropos is a disk array logical volume manager that offers efficient access in both row- and column-major orders. For database systems, this translates into efficient access to complete records as well as scans of an arbitrary number of

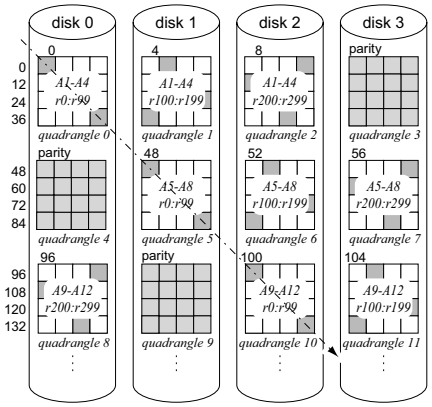


Figure 3: Mapping of database table with 12 attributes onto Atropos logical volume with quadrangles. Each quadrangle holds 4 attributes with 100 records each. The dashed arrow line shows efficient semi-sequential access for retrieving complete records.

table attributes. By utilizing features built into disk firmware and a new data layout, Atropos delivers the aggregate bandwidth of all disks for accesses in both majors, without penalizing small random I/O accesses.

The basic allocation unit in Atropos is the quadrangle, which is a collection of logical volume LBNs. A quadrangle spans the entire track of a single disk along one dimension and a small number of adjacent tracks along the other dimension. Each successive quadrangle is mapped to a different disk, much like a stripe unit of an ordinary RAID group. Hence, the RAID 1 or RAID 5 data protection schemes fit the quadrangle layout naturally.

Atropos stripes contiguous LBNs across quadrangles mapped to all disks. This provides aggregate streaming bandwidth of all disks for table accesses in column-major order (e.g., for single attribute scans). With quadrangle "width" matching disk track size, sequential accesses exploits the high efficiency of track-based access [2].

Accesses in the other major order (i.e. row-major order), called semi-sequential, proceed to LBNs mapped diagonally across a quadrangle, with each LBN on a different track. Issuing all requests together to these LBNs allows the disk's internal scheduler to service the request with the smallest positioning cost first, given the current disk head position. Servicing the remaining requests does not incur any other positioning overhead thanks to the diagonal layout. Hence, this semi-sequential access pattern is much more efficient than reading some randomly chosen LBNs spread across the set of adjacent tracks of a single quadrangle.

Semi-sequential quadrangle access is used for retrieving minipages with all attributes comprising a full record. If the number of minipages/attributes does fit into a single quadrangle, the remaining minipages are mapped to a quadrangle on a different disk. Using this mapping method, several disks can be accessed in parallel to retrieve full records efficiently.

SUMMARY

Fates is the first database system that leverages the unique characteristics of each level in the memory hierarchy. The decoupling of data layouts at the cache/main-memory and on-line storage levels is possible thanks to carefully orchestrated interactions between each Fate. Properly designed abstractions that hide specifics, yet allow the other components to take advantage of their unique strengths achieve efficient query execution at all levels of the memory hierarchy.

REFERENCES

[1] A. Ailamaki, D. J. DeWitt, M. D. Hill, M. Skounakis: Weaving Relations for Cache Performance. Proc. of VLDB, 169-180. Morgan Kaufmann, 2001.

[2] J. Schindler, J. L. Griffin, C. R. Lumb, G.R. Ganger. Track-aligned Extents: Matching Access Patterns to Disk Drive Characteristics. Conf. on File and Storage Technologies, 259-274. Usenix Association, 2002.

AWARDS & OTHER PDL NEWS

September 2003

Ganger & the PDL Awarded Equipment Grants from IBM and Intel

IBM Corporation and Intel Corporation have each generously donated over $80K in equipment to provide an early testbed for PDL's new Self-* Storage project. The article on page 1 describes Self-* Storage.

September 2003

NSF Grant to Fund Self-* Storage Research

PDL researchers have received a $1.5 million NSF grant to pursue the Self-* Storage project, which seeks to create large-scale self-managing, self-organizing, self-tuning storage systems from generic servers. The project PI is Greg Ganger (ECE and CS; Director of PDL), and the co-PIs are Natassa Ailamaki (CS), Anthony Brockwell (Statistics), Garth Gibson (CS), and Mike Reiter (ECE and CS).

September 2003

Congratulations Natassa and Babak!

Natassa Ailamaki and Babak Falsafi are thrilled to announce the arrival of their daughter Niki Falsafi, who was born at Magee Women's Hospital at 7:44 a.m. on September 27.

September 2003

5 CMU Professors Receive NSF Grant to Study Drinking Water Quality and Security*

Faculty members Jeanne VanBriesen, Civil and Environmental Engineering (CEE), Christos Faloutsos, Computer Science, Anastassia Ailamaki, Computer Science, Mitch Small, CEE and Engineering and Public Policy, and Paul Fischbeck, Social and Decision Sciences, have received a National Science Foundation grant of $1.5 million for a new project called "SENSORS: Placement and Operation of an Environmental Sensor Network to Facilitate Decision Making Regarding Drinking Water Quality and Security."

*From CMU newspaper *The Tartan*, Sept. 8, 2003.

August 2003

Ted and Addie Marry!

Ted Wong and Addie Tyler were married on Aug 9, 2003 in the Sage Chapel at Cornell University in Ithaca, New York. Ted will be defending his dissertation on October 24, and then he and Addie are moving to the Palo Alto area in California where Ted will be working at IBM.

June 2003

Congratulations to Jiri and Katrina!

Jiri Schindler and Katrina Van Dellen were married at The Wiley Inn in Peru, Vermont on June 7, 2003. Jiri successfully defended his PhD dissertation on August 22 and will be moving to the Boston area soon to work at EMC.

July 2003

Jiri Schindler Receives Best Paper Award at VLDB Workshop

Jiri Schindler's paper "Matching Database Access Patterns to Storage Characteristics," co-authored with Anastassia Ailamaki and Greg Ganger, has received the award for Best Paper in the VLDB 2003 PhD Workshop from among 34 submissions. Jiri will present his paper at the VLDB PhD Workshop, co-located with VLDB 2003 (29th Conference on Very Large Databases) in Berlin in September. The VLDB 2003 PhD Workshop brings together PhD students working on topics related to the VLDB Conference series, to present and discuss their research in a constructive and international atmosphere. This paper is available on our publications page.

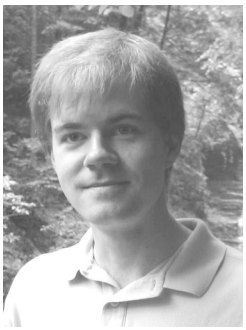
May 2003

John Linwood Griffin Receives Intel Fellowship

Our congratulations to John Linwood Griffin, who has been selected as a recipient of a 2003-04 Intel Foundation PhD Fellowship Award. The fellowship will cover John's full tuition, fees, and stipend for the year. Addi-

Continued on page 11

Continued from page 10



tionally, the fellowship provides John with an Intel-based laptop and a mentor who will act as a link between the student and those people pursuing relevant research at Intel. The fellowship does not involve an internship; rather, it is targeted at Ph.D. candidates within 18 months of degree completion. Approximately 35 candidates are selected annually for the award from a very competitive field.

May 2003

Brandon Salmon Awarded NSF Graduate Research Fellowship*

The winners of this year's National Science Foundation (NSF) Graduate Research Fellowships include Electrical and Computer Engineering (ECE) students Jennifer Morris and Brandon Salmon. The NSF's Graduate Research Fellowship funds three years of graduate study, including a $27,500 stipend for the first 12 months and an annual tuition allowance of $10,500, paid to the university. This year's contest was the most competitive in recent history: 7,788 applicants vied for 900 fellowships.

*CMU 8 1/2 x 11 News, May 1, 2003.

May 2003

James Hendricks Awarded National Defense Fellowship

Congratulations to James Hendricks, who has been awarded a National Defense Science and Engineering Graduate (NDSEG) Fellowship. The prevailing goal of this highly competitive program is "to provide the United States with talented, doctorally trained American men and women who will lead state-of-the-art research projects in disciplines having the greatest payoff to national security requirements." Since the program's inception 14 years ago, approximately 1,800 fellowships have been awarded from about 28,500 applications received. James' fellowship is supported by the Air Force Office of Scientific Research (AFOSR) and covers his full tuition and required fees during that term. Fellows have no military or other service obligations, and must be working towards a PhD.

February 2003

Welcome Michelle!

Michelle Liu was born to Mengzhi Wang and Honliang Liu on February 2, 2003 at 7 lbs. 9 oz. and 17.2 inches. It looks like she is already following her Mom's footsteps into computer related research.

January 2003

Chris Long and Greg Ganger Receive Funding from C3S

Chris Long and Greg Ganger have been awarded seed funding from the Center for Computer Security (C3S) at Carnegie Mellon for their project "Access Control for the Masses." The project will fall within a new PDL research area dealing with Better User Interfaces.

February 2003

Congrats to Chris & Alexis Long!

Robert Nicholas Long joined his parents Chris and Alexis on Feb. 16, 2003! What a bright and happy looking little fellow.

October 2002

Welcome to the Newest Seshan!

Srini Seshan and his wife Asha welcomed their first baby on October 1, 2002. Sanjay Seshan was 7lbs. 2oz. and 20 3/4 inches long at birth. In the photo at 11 months of age, it looks like he has grown quite a bit since then!

Bruce Worthington of Microsoft discussing research with Steve Schlosser, Jiri Schindler, Brandon Salmon & Craig Soules.

Continued from page 1

their performance and reliability status, and exchange information with human administrators. Dataset assignments and redundancy schemes are dynamically adjusted based on observed and projected performance and reliability. We refer to self-* collections of storage bricks as storage constellations.

Administration and Organization. At the top level, a self-* storage system will still require human administrators to provide guidance, approve procurement requests, physically install and repair equipment, and provide high-level goals for the system. A self-* storage system will need an administrative interface to provide information and offer solutions to the human administrator when problems arise or trade-offs (e.g., between performance and reliability) are faced. A self-* administrative interface should also help administrators decide when to acquire new components, which would then be automatically integrated into the self-* constellation.

The supervisors, processes playing an organizational role in the infrastructure, form a management hierarchy. They dynamically tune dataset-to-worker assignments, redundancy schemes for given datasets, and router policies. The hierarchy of supervisor nodes controls data partitioning and request distribution among workers, with the objective of partitioning data and goals among its subordinates (workers or lower-level supervisors) such that, if its children meet their assigned goals, the goals for the entire subtree will be met. By communicating goals down the tree, a supervisor gives its subordinates the ability to assess their own performance relative to goals as they internally tune; the supervisors need not concern themselves with details of how subordinates meet their goals. The top of the hierarchy interacts with the system administrator, receiving high-level goals for datasets and providing status and procurement requests. Additional internal services, referred to as administrative assistants, are also needed for a self-* constellation to function. Examples include event logging for problem diagnosis, directory services to help translate component/service names to their locations in the network, and security services.

Data Access and Storage. Workers store data and routers ensure that I/O requests are delivered to the appropriate workers for service. Thus, self-* clients interact with a self-* router to access data. We envision two types of self-* clients. Trusted clients are considered a part of the system, and may be physically co-located with other components (e.g., router instances); examples are file servers or databases that use the self-* constellation as back-end storage. Untrusted clients are modified to support the self-* constellation's internal protocols; they interact directly with self-* workers, via self-* routers, with access privileges verified on each request.

An important part of a self-* router's job is correctly handling accesses to data stored redundantly across storage nodes. Doing so requires a protocol to maintain data consistency and liveness in the presence of failures and concurrency. Since they will have flexibility in deciding which servers should handle certain requests, self-* routers will also have a role in dynamic load balancing as they deliver client requests to the appropriate workers, particularly when new data are created and when READ requests access redundant data. Doing so requires metadata for tracking current storage assignments, consistency protocols for accessing redundant data, and choices for routing requests.

Self-* workers will service requests for and store assigned data. We expect them to have the computation and memory resources needed to internally adapt to their observed workloads by, for example, reorganizing on-disk placements and specializing cache policies. Workers will also handle storage allocation internally, both to decouple external naming from internal placements and to allow support for internal versioning, since they will keep historical versions (e.g., snapshots) of all data to assist with recovery from dataset corruption. Although the self-* storage architecture would work with workers as block stores (like SCSI or IDE/ATA disks), we believe they will work better with a higher-level abstraction (e.g., objects or files), which will provide more information for adaptive specializations. Self-* workers must also provide support for a variety of maintenance functions, including crash recovery, integrity checking, and data migration.

Continued on page 20

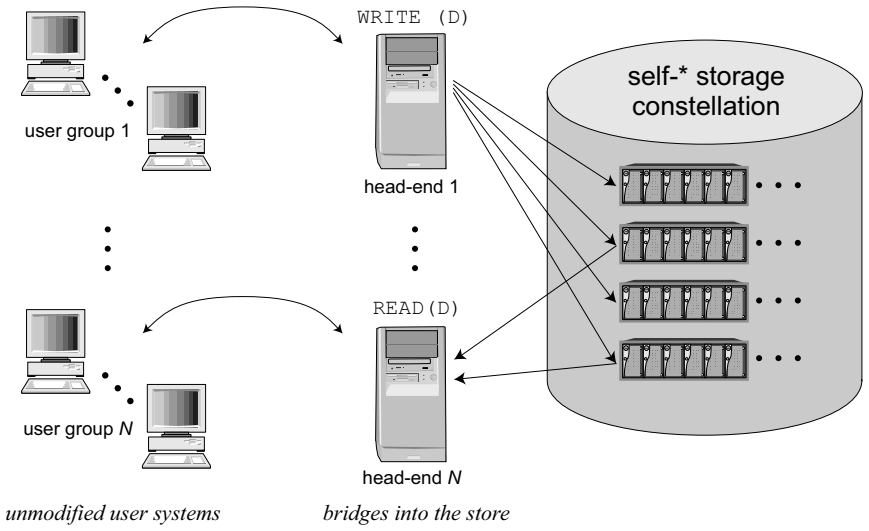


Figure 2: "Head-end" servers bridge external clients into the self-* storage constellation.

PH.D. DISSERTATION

Matching Application Access Patterns to Storage Device Characteristics

Jiri Schindler, ECE
August 22, 2003

Thesis statement: "With sufficient information, a storage manager can exploit unique storage device characteristics to achieve better, more robust I/O performance. This information can be abstract from device specifics, device-independent, and yet expressive enough to allow a storage manager to tune its access patterns to a given device."

This dissertation contends that storage device resources are not utilized to their full potential because too much is hidden behind their high-level storage interfaces. Current storage interfaces do not convey sufficient information to the storage manager to enable it to make informed decisions leading to the most efficient use of the storage device. To bridge the information gap between hosts and storage devices, the storage device should explicitly state its performance characteristics. Using this static information, a storage manager can take advantage of the device's unique strengths and avoid inefficient access patterns.

MS THESIS

A Framework for Implementing Background Storage Applications using Freeblock Scheduling

Eno Thereska, ECE
August 2003

There are many disk maintenance tasks that are required for robust system operation but have loose time constraints. Such "background" tasks need to complete within a reasonable amount of time, but are generally intended to occur during otherwise idle time. Examples include cache write-back, defragmentation, backup, integrity checking, virus scanning, report generation, tamper detection, and index generation. Developers of such applications have had no clean way of designing these applications. The main reason for that is the traditional lack of "trust" applications have had on storage devices to do what is best for the application, with consequences reflected in the narrow interfaces between the two.

We introduce a framework for implementing background storage applications by adding a new asynchronous interface to the storage device. Applications register background tasks through the interface and the storage device notifies them of their completion. The storage device uses freeblock scheduling together with idle time detectors to guarantee that the background applications will make good progress independent on the load of the system and without impacting the foreground workload. This framework is described and evaluated in the context of two real applications, a snapshot-based backup and a cache cleaner.

MS THESIS

Storage-based Intrusion Detection: Watching Storage Activity For Suspicious Behavior

Adam Pennington, ECE
August 2003

Please see the abstract of the paper of the same name on pg. 7 for an outline of this thesis.

MS THESIS

Opportunistic Use of Content Addressable Storage for Distributed File Systems

Niraj Tolia, ECE
May 2003

Please see the abstract of the paper of the same name on pg. 17 for an outline of this thesis.

THESIS PROPOSAL

Efficient, Flexible Consistency for Highly Fault Tolerant Storage

Garth Goodson, ECE
August 18, 2003

Fault-tolerant storage systems spread data redundantly across a set of storage-nodes in an effort to preserve and provide access to data despite failures. One difficulty created by this architecture is the need for a consistent view, across storage-nodes, of the most recent update. Such consistency is made difficult by concurrent updates, partial updates made by clients that fail, and failures of storage-nodes. This thesis will demonstrate how to achieve scalable, highly fault-tolerant storage systems by leveraging an efficient and flexible family of strong consistency protocols enabled by server versioning. In particular, the design of block-based storage systems and file systems will be evaluated. The storage protocol is made space-efficient through the use of erasure codes and made scalable by offloading work from the storage-nodes to the clients. The protocol family is flexible in that it covers a broad range of system model assumptions with no changes to the client-server interface, server implementations, or system structure. Each protocol scales with its requirements—it only does work necessitated by the system and fault models.

THESIS PROPOSAL

Staged Database Systems

Stavros Harizopoulos, SCS
April 24, 2003

Thesis Statement: "By organizing and assigning system components into self-contained stages, database systems can exploit instruction and data commonality across concurrent requests thereby increasing throughput. Furthermore, staged database systems are more scalable, easier to extend,

Continued on page 18

INFERRING ATTRIBUTES FROM CONTEXT

Craig Soules & Greg Ganger

As storage capacity continues to increase, users find it increasingly difficult to manage their files using traditional directory hierarchies. Attribute-based naming enables powerful search and organization tools for ever-increasing user data sets. However, such tools are only useful in combination with accurate attribute assignment. Existing systems rely on user input and content analysis, but they have enjoyed minimal success. We propose several new approaches to automatically assigning attributes to files through context analysis, a technique that has been successful in the Google web search engine. With extensions like application hints (e.g., web links for downloaded files) and inter-file relationships, it should be possible to infer useful attributes for many files, making attribute-based search tools more effective.

Existing Organizational Tools

As storage capacity increases, the amount of data belonging to an individual user increases accordingly. Soon, storage capacity will reach a point where there will be no reason for a user to ever delete old content — in fact, the time required to do so would be wasted. The challenge has shifted from deciding what to keep to finding particular information when it is desired. To meet this challenge, we need to improve our approach to personal data organization.

Today, most systems provide a tree-like directory hierarchy to organize files. Although this is easy for most users to understand, it does not provide the flexibility required to scale to large numbers of files. In particular, the strict hierarchy provides only a single categorization with no cross-referenced information.

Alternatives to the standard directory hierarchy systems generally assign attributes to files, providing the ability to cluster and search for files by their attributes. An attribute can be any metadata that describes the file, although most systems use keywords or <category, value> pairs. The key challenge is assigning useful, meaningful attributes to files.

Unfortunately, the two most prevalent methods of attribute assignment, user input and content analysis, have been largely unsuccessful. Although users often have a good understanding of the files they create, it can be time-consuming and unpleasant to distill that information into the right set of keywords. As a result, users are understandably reluctant to do so. On the other hand, content analysis takes none of the user's time, and can be performed entirely in the background to eliminate any potential performance penalty. However, the complexity of language parsing, combined with the large number of proprietary file formats and non-textual data types, restricts the effectiveness of content analysis.

Context-based Attributes

Early web search-engines, (e.g. Lycos), relied upon user input (user submitted web pages) and content analysis (word counts, word proximity, etc.). Although valuable, the success of these systems has been eclipsed by the success of Google.

To provide better search results, Google utilizes two forms of context analysis. First, it uses the text associated with a link to determine attributes for the linked site. This text gives the context of both the creator of the linking site and the user who clicks on the link at that site. The more times that a particular word links to a site, the higher that word is ranked for that site. Second, Google uses the actions of a user after a search to decide what the user wanted from that search. For example, if a user clicks on the first four links of a given search, and then does not return, it is likely that the fourth link was the best match, providing the user's context for those search terms.

Unfortunately, Google's approach to indexing does not translate directly into the realm of file systems. Much of the information that Google relies on does not exist within a file system. Also, Google's query feedback mechanism relies on two properties: users are normally looking for the most popular sites when they perform a query, and they have a large user base that will repeat the same query many times. Conversely, in file systems, users usually search for files that have not been accessed in a long time, because they usually remember where recently accessed files reside, and there is generally only a single user for each set of files, making it unlikely that frequent queries will be generated for any given file.

Context-based Attributes in File Systems

We are investigating four approaches to automatically gathering context information for use in file systems. The first two focus on gathering attributes when a file is created or accessed. The second two focus on propagating attributes among related files to increase the coverage of attribute assignment. Together, these techniques should categorize a much broader set of files than creation-based attribute assignment alone.

Application assistance: Although computers provide a vast array of functionality, most people use their computer for a limited set of tasks using a small set of applications that, in turn, access and create most of the user's files. Modifying these applications to provide hints about the user's context could provide invaluable attribute information.

Existing user input: Although most users are not willing to input additional information, they are willing to choose a directory and name for their files. Each of the sub-directories along the path and the file name itself probably contain context information that can be used to assign attributes. For example, if the user stores a file in "/home/papers/FS/Attribute-based/Semantic91.ps," then it

Continued on page 15

Continued from page 14

is likely that they believe the file is a "paper" having to do with "FS," "attribute-based," and "semantic."

User access patterns: As users access their files, the pattern of their accesses provides a set of temporal relationships between files. A possible use of this information is to help propagate information between related files. For example, accessing "SemanticFS.ps" and "Gopal.ps" followed by updating "related.tex" may indicate a relationship between the three files. Subsequently, accessing "related.tex" and creating "FindingFiles.ps" may indicate a transitive relationship.

Inter-file content analysis: Content analysis will continue to be an important part of automatically assigning attributes. In addition to existing per-file analysis techniques, our focus on creating context-based connections between files suggests another source of attributes: content-based relationships. For example, some current file systems use hashing to eliminate duplicate blocks within a file system, or even locate similarities on non-block aligned boundaries. Such content overlap could also be used to identify related files, by treating files with large matching data sets as related. Similarly, users (or the system) will often keep several slightly different versions of a file. Although these files generally contain differences, often the inherent information contained within does not change (e.g., a user may keep three instances of their resume, each focused for a different type of job application). This gives the system two opportunities for content analysis. First, content comparison can identify related files. Second, by performing content analysis solely on the differences between versions, it may be possible to determine version-specific attributes, making it easier for users to locate individual version instances.

Prototype Evaluation System

Figure 1 shows an overview of a prototype system for evaluating context-based attribute assignment schemes. The system is composed of four main parts: the tracer, the application interface, the analyzer, and the database.

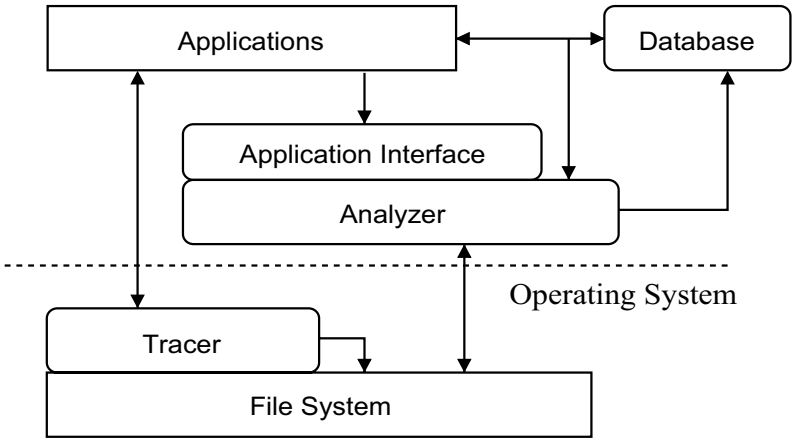


Figure 1: A prototype system for evaluating context-based attribute assignment schemes.

The tracer keeps a trace of all file system activity in the system. Any file system calls made by applications are tracked and stored in a file for later offline analysis. This allows a single system to employ a variety of different analysis techniques. The application interface allows applications to pass context information into the system, such as email header information or link information from a web browser. This information is used by the analyzer to generate attributes for files. The analyzer combines application information, and offline trace analysis to generate attributes for files. All updated attribute information is passed to the database, which provides the search interface to the application. It allows applications to locate files using the file attributes assigned by the analyzer. Feedback from the search results is pushed to the analyzer for further attribute refinement.

This design could include multiple databases. In order to compare the results of different trace analysis algorithms, the analyzer could maintain a database for each, and users could compare the results of the different approaches. For more information on this project see Soules [1] or the PDL project page at www.pdl.cmu.edu/AttributeNaming/.

References

[1] Craig A.N. Soules, Greg Ganger. Why Can't I Find My Files? New methods for automating attribute assignment. Proceedings of the Ninth HotOS Workshop, USENIX Association, May 2003.

Young Professor Tim Ganger teaching the ECE 18-746 storage systems class.

Continued from page 7

tively limited resources (CPU, memory and/or communication bandwidth and power) poses some interesting challenges. We need both powerful and concise "languages" to represent the important features of the data, which can (a) adapt and handle arbitrary periodic components, including bursts, and (b) require little memory and a single pass over the data.

This allows sensors to automatically (a) discover interesting patterns and trends in the data, and (b) perform outlier detection to alert users. We need a way so that a sensor can discover something like "the hourly phone call volume so far follows a daily and a weekly periodicity, with bursts roughly every year," which a human might recognize as, e.g., the Mother's day surge. When possible and if desired, the user can then issue explicit queries to further investigate the reported patterns.

In this work we propose AWSOM (Arbitrary Window Stream mOdeling Method), which allows sensors operating in remote or hostile environments to discover patterns efficiently and effectively, with practically no user interventions. Our algorithms require limited resources and thus can be incorporated in individual sensors, possibly alongside a distributed query processing engine. Updates are performed in constant time, using sublinear (in fact, logarithmic) space. Existing, state of the art forecasting methods (AR, SARIMA, GARCH, etc) fall short on one or more of these requirements. To the best of our knowledge, AWSOM is the first method that has all the above characteristics.

Experiments on real and synthetic datasets demonstrate that AWSOM discovers meaningful patterns over long time periods. Thus, the patterns can also be used to make long-range forecasts, which are notoriously difficult to perform automatically and efficiently. In fact, AWSOM outperforms manually set up auto-regressive models, both in terms of long-term pattern detection and modeling, as well as by at least 10x in resource consumption.

Location-based Node IDs: Enabling Explicit Locatlity in DHTs

Zhou, Ganger & Steenkiste

Carnegie Mellon University School of Computer Science Technical Report CMU-CS-03-171, August 2003.

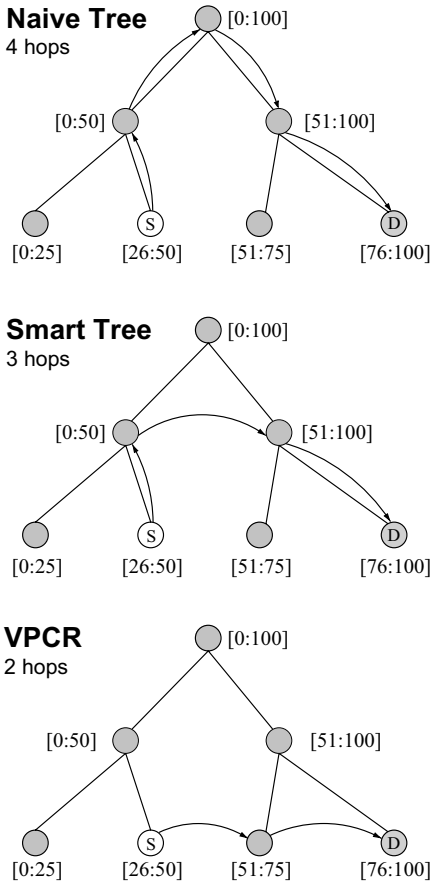
Current peer-to-peer systems based on DHTs struggle with routing locality and content locality because of random node ID assignment. To address these issues, we promote the use of location-based node IDs to encode physical topology and improve routing. This gives applications explicit knowledge about and control over data locality at a coarse-grain. Applications can place content in particular regions or route towards a close replica. Schemes to address the difficulties that ensue, particularly load imbalance, are discussed.

GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information

Newsome & Song

Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003). November 5-7, 2003, Redwood, CA.

The widespread deployment of sensor networks is on the horizon. One of the main challenges in sensor networks is to process and aggregate data in the network rather than wasting energy by sending large amounts of raw data to reply to a query. Some efficient data dissemination methods, particularly data-centric storage and information aggregation, rely on efficient routing from one node to another. In this paper we introduce GEM (Graph EMbedding for sensor networks), an infrastructure for node-to-node routing and data-centric storage and information processing in sensor networks. Unlike previous approaches, it does not depend on geographic information, and it works well even in the face of physical obstacles. In GEM, we construct a labeled graph that can be embedded in the original network topology in an efficient and distributed fashion. In that graph, each node is given a label that encodes its position in the original network topology. This allows messages to be efficiently routed through the network, while each node only needs to know the labels of its neighbors. To demonstrate how GEM can be applied, we have developed a concrete graph em-



Virtual polar coordinate routing for VPCS. A packet is routed from S to D using the three routing algorithms. Smart Tree and VPCR use a 1-hop neighborhood.

Continued on page 17

Continued from page 16

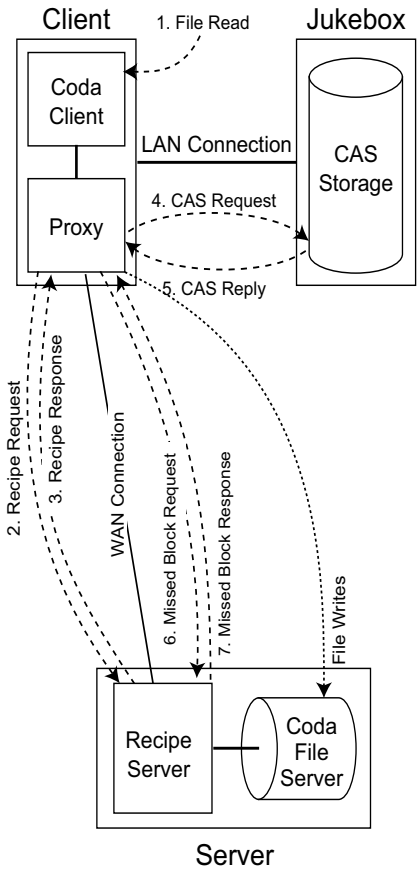
bedding method, VPCS (Virtual Polar Coordinate Space). In VPCS, we embed a ringed tree into the network topology, and label the nodes in such a manner as to create a virtual polar coordinate space. We have also developed VPCR, an efficient routing algorithm that uses VPCS. VPCR is the first algorithm for node-to-node routing that guarantees reachability, requires each node to keep state only about its immediate neighbors, and requires no geographic information. Our simulation results show that VPCR is robust on dynamic networks, works well in the face of voids and obstacles, and scales well with network size and density.

Opportunistic Use of Content Addressable Storage for Distributed File Systems

Tolia, Kozuch, Satyanarayanan, Karp, Bressoud & Perrig

Proceedings USENIX Annual Technical Conference, General Track 2003: 127-140, June 9-14, San Antonio, TX.

Motivated by the prospect of readily available Content Addressable Storage (CAS), we introduce the concept of file recipes. A file's recipe is a first-class file system object listing content hashes that describe the data blocks composing the file. File recipes provide applications with instructions for reconstructing the original file from available CAS data blocks. We describe one such application of recipes, the CASPER distributed file system. A CASPER client opportunistically fetches blocks from nearby CAS providers to improve its performance when the connection to a file server traverses a low-bandwidth path. We use measurements of our prototype to evaluate its performance under varying network conditions. Our results demonstrate significant improvements in execution times of applications that use a network file system. We conclude by describing fuzzy



System diagram.

block matching, a promising technique for using approximately matching blocks on CAS providers to reconstitute the exact desired contents of a file at a client.

Metadata Efficiency in a Comprehensive Versioning File System

Soules, Goodson, Strunk & Ganger

2nd USENIX Conference on File and Storage Technologies, San Francisco, CA, Mar 31- Apr 2, 2003.

A comprehensive versioning file system creates and retains a new file version for every WRITE or other modification request. The resulting history of file modifications provides a detailed view to tools and administrators seeking to investigate a suspect system state. Conventional versioning systems do not efficiently record the many prior versions that result. In particular, the versioned metadata they keep consumes almost as much space as the versioned data. This paper examines two space-efficient metadata structures for versioning file systems and describes their integration into the Comprehensive Versioning File System (CVFS). Journal-based metadata encodes each metadata version into a single journal entry; CVFS uses this structure for inodes and indirect blocks, reducing the associated space requirements by 80%. Multiversion b-trees extend the per-entry key with a timestamp and keep current and historical entries in a single tree; CVFS uses this structure for directories, reducing the associated space requirements by 99%. Experiments with CVFS verify that its current-version performance is similar to that of non-versioning file systems. Although access to historical versions is slower than conventional versioning systems, checkpointing is shown to mitigate this effect.

Byzantine-tolerant Erasure-coded Storage

Goodson, Wylie, Ganger & Reiter

Carnegie Mellon University SCS Technical Report CMU-CS-03-187, September, 2003.

This paper describes a decentralized consistency protocol for survivable storage that exploits data versioning within storage-nodes. Versioning enables the protocol to efficiently provide linearizability and wait-freedom of read and write operations to erasure-coded data in asynchronous environments with Byzantine failures of clients and servers. Exploiting versioning storage-nodes, the protocol shifts most work to clients. Reads occur in a single round-trip unless clients observe concurrency or write failures. Measurements of a storage system using this protocol show that

Continued on page 19

Continued from page 13

and more readily fine-tuned than traditional database systems."

Database system architectures face a rapidly evolving operating environment where millions of users store and access terabytes of data. To cope with increasing demands for performance high- end DBMS employ parallel processing techniques coupled with a plethora of sophisticated features. However, the widely adopted work-centric thread-parallel execution model entails several shortcomings that limit server performance, the most important being failure to exploit instruction and data commonality across concurrent requests. Moreover, the monolithic approach in DBMS software has lead to complex designs which are difficult to extend.

This thesis introduces a staged design for high-performance, evolvable DBMS that are easy to fine-tune and maintain. I propose to break the database system into modules and encapsulate them into self-contained stages connected to each other through queues. The staged, data-centric design remedies the weaknesses of modern DBMS by providing solutions at (a) the hardware level: it optimally exploits the underlying memory hierarchy, and (b) at a software engineering level: it is more scalable, easier to extend, and more readily fine-tuned than traditional database systems.

THESIS PROPOSAL

Using MEMS-based Storage Devices in Computer Systems

Steve Schlosser, ECE
June 5, 2003

MEMS-based storage is an interesting new technology that promises to bring fast, non-volatile, mass data storage to computer systems. MEMS-based storage devices (MEMStores) themselves consist of several thousand read/write tips, analogous to the read/write heads of a disk drive, which read and write data in a recording medium. This medium is coated on a moving rectangular surface that is positioned by a set of MEMS actuators. Access times are expected to be less than a millisecond with power consumption 10–100X less than a low-power disk drive, while streaming bandwidth and volumetric density are expected to be around those of disk drives.

We are starting to exploring how MEMStores would best be used in computer systems and how those systems should adapt to their differences as compared to disks. For example, existing operating system policies are tuned for disks, including request scheduling, data layout, and power conservation. Also, given the performance, capacity, and non-volatility of MEMStores, they represent a new, intermediate member of the memory hierarchy. My thesis is that most of these aspects can conform, with little penalty, to disk-like policies and usages.

Because MEMStores perform basically like fast disks, with only a few exceptions, they can be treated by systems as such. My dissertation will show that for most workloads, the same linear logical block abstraction that is used for disk drives is appropriate for MEMStores. The benefit of using the same abstraction is that MEMStores can be easily integrated into computer systems with little or no change.

THESIS PROPOSAL

D-SPTF: Decentralized Scheduling for Storage Bricks

Christopher Lumb, ECE
August 19, 2003

Distributed Shortest-Positioning Time First (D-SPTF) is a request distribution protocol for decentralized systems of storage servers.

D-SPTF exploits high-speed interconnects to dynamically select which server, among those with a replica, should service each read request. In doing so, it simultaneously balances load, exploits the aggregate cache capacity, and reduces positioning times for cache misses. For network latencies of up to 0.5ms, D-SPTF performs as well as would a hypothetical centralized system with the same collection of CPU, cache, and disk resources. Compared to existing decentralized approaches, such as hash-based request distribution, D-SPTF achieves up to 50% higher throughput and adapts more cleanly to heterogenous server capabilities.

THESIS PROPOSAL

Autonomous Spatio-Temporal Data Mining

Spiros Papadimitriou, SCS
May 5, 2003

The goal of data mining is to facilitate the extraction of useful information from large collections of data. Thus, eliminating the *requirement* of user intervention is essential. We propose to develop fast tools for spatio-temporal data mining towards that goal. We have completed work in the area of spatio-temporal data mining and, in particular, outlier detection and time series modeling. These provide sufficient evidence that we can improve upon previous techniques.

THESIS PROPOSAL

Prefetching and Locality Optimizations for Database Memory Hierarchy Performance

Shimin Chen, SCS
May 2, 2003

Database performance studies have been traditionally focused on I/O performance. Recently, researchers have shown that, on traditional disk-oriented databases, roughly 50% or more of the execution time in memory is wasted due to cache misses. Therefore, to exploit the full power of modern computer systems requires optimizing both cache and disk performance in the memory hierarchy, which together

Continued on page 20

Continued from page 17

the protocol scales well with the number of failures tolerated, and that it outperforms a highly-tuned instance of Byzantine-tolerant state machine replication.

Robustness Hinting for Improving End-to-End Dependability

Bigrigg

Second Workshop on Evaluating and Architecting System Dependability (EASY). In conjunction with ASPLOS-X. Sunday, 6 October 2002, San Jose, California, U.S.A.

File systems make unreasonable attempts to provide data to the point that they will block an application instead of passing the error on to the application to handle. Transient problems such as network congestion or outages and heavily loaded systems or denial of service attacks can lead to failure-like situations. Alternative mechanisms have been developed for the file system to trade performance for robustness in an attempt to always guarantee full availability of data. These mechanisms may not be necessary, as the application programmer may have already accounted for such situations. By hinting to the file system the application's ability to handle errors it is possible for the file system to make better resource allocation decisions and improve end-to-end dependability.

Data Staging on Untrusted Surrogates

Flinn, Sinnamohideen, Tolia & Satyanarayanan

Proceedings 2nd USENIX Conference on File and Storage Technologies (FAST03), Mar 31-Apr 2, 2003, San Francisco, CA.

We show how untrusted computers can be used to facilitate secure mobile data access. We discuss a novel architecture, data staging, that improves the performance of distributed file systems running on small, storage-limited pervasive computing devices. Data staging opportunistically prefetches files and caches them on nearby surrogate machines. Surrogates are untrusted and unmanaged: we use end-to-end encryption and secure hashes to provide privacy and authenticity of data and have designed our system so that surrogates are as reliable and easy to manage as possible. Our results show that data staging reduces average file operation latency for interactive applications running on the Compaq iPAQ hand-held by up to 54%.

Give PDL storage systems researchers snow and a plastic chair and see what happens!

YEAR IN REVIEW

Continued from page 4

Interaction and Security Systems in Fort Lauderdale, FL.

March 2003

- Christos Faloutsos presented a tutorial on "Data Mining the Internet" at INFOCOM, San Francisco, CA.

January 2003

- Over the past term, several visitors have contributed to our Storage Systems course, including: Dave Anderson, Seagate; Steve Kleiman, NetApp; Ric Wheeler, EMC; Harald Skardal, NetApp; Jim Hughes, StorageTek; Richie Lary, Independent Consultant; and Mark Carlson, Sun.
- Stavros Harizopoulos presented "A Case for Staged Database Systems" at the First International Conference on Innovative Data Systems Research (CIDR), in Asilomar, CA.

December 2002

- Greg Ganger chaired the session on Decentralized Storage Systems at OSDI in Boston, MA.
- Timmy Ganger gave a guest lecture in 15-712 (Advanced OS and Distributed Systems).
- Ted Wong presented "Verifiable Secret Redistribution for Archive Systems" at the First International IEEE Security in Storage Workshop.

November 2002

- SDI Speaker: Richard Golding, then of Panasas, Inc., spoke on "The Palladio Project: A Survivable, Scalable Distributed Storage System."
- DB Seminar Speaker: C. Mohan of IBM on "Future directions in Data Mining: Streams, Networks, Self-similarity and Power Laws."
- Christos Faloutsos gave the keynote talk at CIKM in McLean, VA and was also an invited speaker at the NSF NGDM Workshop in Baltimore, MD and the N.A.S. Workshop in Washington, DC.

October 2002

- SDI Speaker: John Wilkes of HP Labs on "Travelling to Rome—QoS Specifications for Automated Storage System Management."

PROPOSALS & DEFENSES

Continued from page 18

have not been well studied for database systems before. My thesis is that prefetching and locality optimizations can effectively improve both cache and disk performance of database systems and that differences between the cache-to-memory and the memory-to-disk gap play a significant role in the design and choice of specific prefetching and locality optimization techniques.

To validate my thesis, I revisit two important classes of database algorithms, B+tree index algorithms and join algorithms. In my preliminary work, I have exploited cache prefetching to improve search and range scan operations of main memory B+trees. I have studied fractal prefetching B+trees, which are a new type of B+trees that optimize both cache and disk performance. In fractal prefetching B+trees, smaller cache-optimized trees are embedded in disk pages to improve data locality for index search. Both cache prefetching and I/O prefetching are used to improve performance. I have worked on improving hash join cache performance by exploiting inter-tuple parallelism through prefetching. For my proposed future work, I will be focusing on utilizing history information to improve join performance. Since updates are relatively infrequent compared to joins in DSS and OLAP environments, it is possible to use history information about matching tuples to guide and improve future join performance. In contrast to previous studies with join indices, I want to analyze the history information to identify data locality in the joining relations and then improve join performance by exploiting the data locality and using prefetching.

THESIS PROPOSAL

Prototyping Without Prototyping: Evaluating hypothetical storage components in real systems

John Linwood Griffin, ECE
August 1, 2003

For my dissertation research I am continuing our investigation into the utility and capabilities of timing-accurate storage emulation (TASE).

Our previous work demonstrates that TASE is feasible for evaluating both evolutionary changes (faster platter speeds; modified firmware algorithms) and revolutionary changes (MEMS-based technology) to storage devices.

Several interesting questions remain before this new storage evaluation technique can be fully utilized by researchers and developers. For example, an emulator may have to measure and deal with variable externally-induced timing errors during an experiment. How can an evaluator rest assured that the emulator is correctly compensating for these errors? As another example, an emulator may need to keep data in RAM in order to provide per-request data before each request completes. How can an emulator meet such deadlines when the experimental working set is larger than the emulator's RAM?

SELF-* STORAGE

Continued from page 12

Ursa Minor and Ursa Major

In order to effectively explore how our ideas will simplify storage administration, it is essential that operational systems be built and deployed. The Parallel Data Lab has been developing technologies relevant to the self-* storage architecture for several years, allowing us to build a prototype relatively quickly. Our initial focus will be on implementing the data protection aspects, embedding instrumentation, and enabling experimentation with performance and diagnosis.

Our first prototype, named Ursa Minor, will be approximately 15 TB spread over 45 small-scale storage bricks. The main goal of this first prototype is rapid (internal) deployment to learn lessons for the design of a second, larger-scale instantiation of self-* storage. Ursa Major will be a large-scale (~1 PB) storage constellation, called Ursa Major. Its data storage capacity will be available to research groups around Carnegie Mellon (e.g. groups involved data mining and scientific visualization) who rely on large quantities of storage for their work. We are convinced that such deployment and maintenance is necessary to evaluate self-* storage's ability to simplify administration for system scales and workload mixes that traditionally present difficulties. As well, our use of low-cost hardware and immature software will push the boundaries of fault-tolerance and automated recovery mechanisms, which are critical for storage infrastructures. From the perspective of their users, our early self-* constellations will look like really big, really fast, really reliable file servers (initially NFS version 3). The decision to hide behind a standard file server interface was made to reduce software version complexities and user-visible changes-user machines can be unmodified, and all bug fixes and experiments can happen transparently behind a standard file server interface. The resulting architecture is illustrated in Figure 2. Direct, untrusted client access will be added over time.

For more information, please see the Self-* Storage project page at http://www.pdl.cmu.edu/SelfStar/.