PARALLEL DATA LABORATORY
CARNEGIE MELLON UNIVERSITY

THE

PDL Packet

THE NEWSLETTER ON PDL ACTIVITIES AND EVENTS • FALL 2002

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

CONSORTIUM MEMBERS

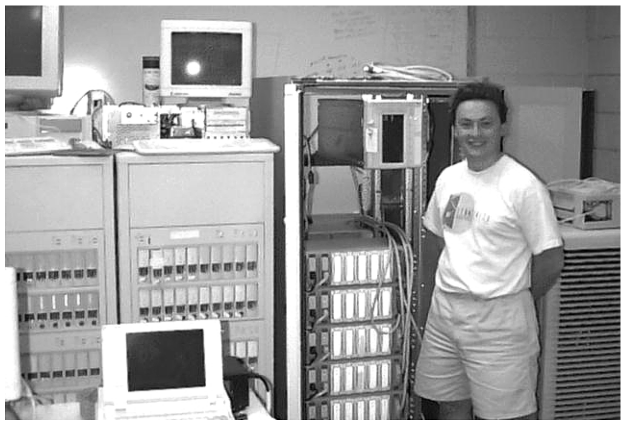
EMC Corporation
Hewlett-Packard Labs
Hitachi, Ltd.
IBM Corporation
Intel Corporation
Microsoft Corporation
Network Appliance
PANASAS, Inc.
Seagate Technology
Sun Microsystems
Veritas Software Corporation

PDL Celebrates its 10th Year!

Greg Ganger & Joan Digney

Over the past 10 years, Carnegie Mellon's Parallel Data Lab (PDL) has established itself as academia's premiere storage systems research center, consistently pushing the state-of-the-art with new storage system architectures, technologies, and design methodologies. Today, the PDL consists of over 40 active researchers and has an annual budget of over $2.5 million. As we prepare for the 10th Annual PDL Retreat, it is fun to look back at how we got here.

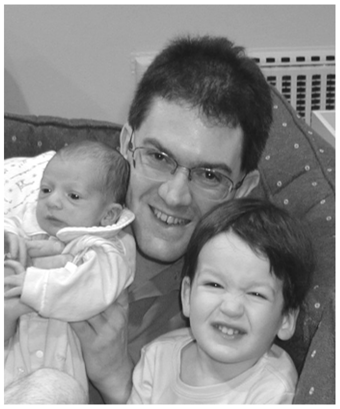
Dr. Garth Gibson founded the PDL in 1993. It started with Gibson and 7 students from CMU's CS and ECE Departments. Having recently finished his Ph.D. research, which defined the industry standard RAID terminology for redundant disk arrays, Gibson guided the PDL researchers in advanced disk array research. The name "Parallel Data Lab" comes from this initial focus on parallelism in storage systems. In the PDL's formative years, its researchers developed technologies for improving failure recovery performance (parity declustering) and maximizing performance in small-write intensive workloads (parity logging). They also developed an aggressive prefetching technology (transparent informed prefetching, or TIP) for converting serial access patterns into highly parallel workloads capable of exploiting large disk arrays.


Garth displays the Scotch I and II storage hardware used in early TIP and RAID research, two of the first research projects undertaken by the PDL as a group. (1994)

The first PDL Retreat was held in October of 1993, and was attended by 20 CMU participants and 11 industry visitors. As is still the case, the first Retreat was highly interactive, allowing the sponsors to hear about and give feedback on PDL research and offering the students a chance to develop relationships with future colleagues and potential employers.

Bill Courtright, a PDL student at the time, recalls everyone wondering if they would have enough solid content to keep the industry attendees' attention throughout the 3-day retreat — but of course it was not a problem. Every year since then, the difficult problem has been what to leave out, as the PDL researchers generate more cool ideas than will fit into

… continued on pg. 12



FROM THE DIRECTOR'S CHAIR

Greg Ganger

Hello from fabulous Pittsburgh!

2002 has been a fun year of building on the growth and solidity achieved last year, which brought us over $5 million in new government funding, 3 new faculty in key growth areas, a new storage systems class, a new storage systems conference (FAST), and several new students and staff. The result has been progress on existing projects combined with cool new research directions. This year is also noteworthy for its historical significance, as it is PDL's 10th year and includes the 10th PDL Retreat.

The PDL continues to pursue a broad array of storage systems research, ranging from the underlying devices to the applications that rely on storage. The past year saw excellent research progress, new Fellowships for PDL students (one from IBM, one from Intel, and one from Microsoft), numerous students spending summers with PDL Consortium companies, and Best Student Paper Awards at two top-tier conferences. Allow me to highlight a few things.

The self-securing devices project has made great strides. Highlighted in April 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 source of funding. In the past year, we have developed network interface software for containing compromised client systems, expanded on self-securing storage, and come up with a new way of detecting intruders: storage-based intrusion detection. Storage devices are uniquely positioned to spot some common intruder actions (such as scrubbing audit logs and inserting backdoors), making this an exciting new concept.

Also exciting has been the continuing growth in database systems research along several parallel tracks. One project is developing new data structures that simultaneously maximize CPU cache and disk performance. A second project complements the first by extending the storage-specific knowledge in database storage managers, allowing them to match their access patterns to device characteristics automatically. Another project applies data mining techniques to I/O traces in order to characterize their spatial and temporal features; Mengzhi Wang won the Best Student Paper Award at Performance 2002 for this work. Other projects are creating tools for automatically partitioning large database tables and database architectures for superior memory performance.

Building on our previous work, we have initiated several interrelated projects in automated storage management. At the lowest level, we are exploring the use of freeblock scheduling for continuous reorganization of data within storage devices. At the level of small collections of storage servers, layered clustering balances load among servers without requiring changes to clients or the client-server protocol. For large systems, at the data center and beyond, we are culling lessons from human organizations to gain traction on dynamic management of self-configuring, self-organizing components. Not to be outdone by those inventing buzzwords for the goal of automated storage management, we collectively refer to these projects by the meta-buzzword "Self-* Storage."

... continued on pg. 3

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 comprise 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 regularly 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 summer home. In 1980, his daughter, Margaret, donated Skibo to a trust that later sold the estate. It is presently being run as a luxury hotel.

CONTACTING 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

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

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

Mukesh Agrawal	David Petrou
Aditya Akella	Brandon Salmon
Shimin Chen	Asad Samar
Chris Costa	Jiri Schindler
Garth Goodson	Steve Schlosser
John Linwood Griffin	Bianca Schroeder
Stavros Harizopoulos	Minglong Shao
James Hendricks	Craig Soules
Andrew Klosterman	John Strunk
Vinod Das Krishnan	Eno Thereska
Chris Lumb	Monica Ullagaddi
Amit Manjhi	Mengzhi Wang
Michael Mesnier	Ted Wong
Spiros Papadimitriou	Jay Wylie
Stratos Papadomanolakis	Shuheng Zhou
Adam Pennington	

FROM THE DIRECTOR'S CHAIR

… continued from pg. 2

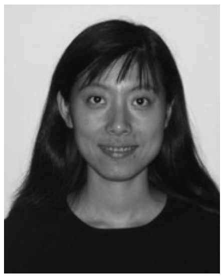
Other ongoing PDL projects are also producing exciting results. For example, the DIXtrac disk characterization tool has been used to explore the use of disk-specific knowledge in file systems, resulting in the Best Student Paper at the first File and Storage Technologies (FAST) conference. The PASIS project continues to develop tools and methodologies for exploring the complex trade-off space of survivable storage systems. The CHIPS research center continues to develop hardware and process technologies to realize MEMS-based storage devices, while PDL researchers are looking at reliability issues and system-level performance issues for MEMS-based storage. For the latter, we developed a timing-accurate storage emulator that looks to systems like a real MEMS-based storage device. We have built a working freeblock scheduler inside a FreeBSD device driver and are building demonstration applications to show off in a future code release. This newsletter and the PDL website offer more details and additional research highlights.

On the education front: this spring, for the second time, we offered our new storage systems course to undergraduates and masters students at Carnegie Mellon. Topics spanned 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 to make them useful. The base lectures were complemented by real-world expertise generously shared by 8 guest speakers from industry, including 2 CTOs and 4 of the 8 members of the SNIA Technical Council. We continue to work on the storage systems textbook, and two other schools (Johns Hopkins and NYU) have already picked up and started teaching similar storage systems courses. We view providing storage systems education as critical to the field's future, so 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

Dawn Song



Dr. Dawn Xiaodong Song joined the PDL and the Departments of ECE and CS this fall as an Assistant Professor. She received her Ph.D. in Computer Science from UC Berkeley in 2002, following the defense of her dissertation titled "Automatic Tools for Building Secure Systems." Her main research interests are in computer security and applied cryptography, including security in systems, networking, databases, electronic commerce. She has worked on a wide range of research projects in the areas of systems and networking security, creating new cryptographic protocols, and designing and developing automatic tools for building secure systems.

A. Chris Long

Dr. A. Chris Long joined the PDL in August as a Post-Doctoral Fellow in ECE. He is working with Greg Ganger on user interfaces to allow system administrators to monitor and manage self-securing network interfaces and storage devices. He is also interested in the areas where human-computer

… continued on pg. 4

YEAR IN REVIEW

October 2002

- Tenth annual PDL Retreat & Workshop

September 2002

- Mengzhi Wang awarded Best Student Paper at Perf'2002 in Rome
- Dawn Song and Adrian Perrig join the PDL

August 2002

- Srinivasan Seshan, Assistant Professor of CS and ECE, helped organize and served as the Tutorials Chair at ACM's SIGCOMM 2002 Conference in Pittsburgh
- Christos Faloutsos tutorials: at SIGCOMM 2002 on "Data Mining the Internet," and at VLDB '02 on "Sensor Data Mining: Similarity Search and Pattern Analysis"
- Chris Long joins the PDL

July 2002

- Stavros Harizopoulos spent part of the summer working with Natassa at the Technical University of Crete in Chania, Greece

June 2002

- SDI speaker: Michael Kozuch, of Intel, on "Internet Suspend/Resume"
- Thesis Proposal (ECE): Jiri Schindler on "Matching Access Patterns to Storage Device Characteristics"

May 2002

- John Strunk, Jay Wylie, Chris Lumb and Steve Schlosser interned at HP Labs in Palo Alto
- Garth Goodson spent the summer interning with IBM at Almaden.
- Thesis Proposal (ECE): David Petrou on "A System for Matching Application Resource Supply and Demand"
- SDI speaker: Winfried W. Wilcke, of IBM Almaden, on "The IceCube Project"

April 2002

- Fourth annual PDL Open House

March 2002

- Mengzhi Wang spoke on "Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic" at the 18th ICDE in San Jose

January 2002

- Jiri Schindler, John Griffin & Chris Lumb awarded Best Student Paper at FAST 2002 for "Track-Aligned Extents: Matching Access Patterns to Disk Drive Characteristics." Jiri gave the conference talk.
- Chris Lumb spoke on "Freeblock Scheduling Outside Disk Firmware" at FAST 2002.
- John Griffin spoke on "Track-aligned Extents: Matching Access Patterns to Disk Drive Characteristics" at FAST 2002
- Over the spring term 8 visitors from industry were guest lecturers for the new storage course, including: Steve Kleiman, Network Appliance; Wayne Rickard, Gadzoox; Dave Anderson, Seagate; Ric Wheeler, EMC; Jim Hughes, StorageTek; Harald Skardal, Network Appliance; John Wilkes, HP; and Roger Cummings, Veritas.

December 2001

- SDI speaker: PDL Alumni Tammo Spalink, grad student at Princeton, on "Building a Robust Software-Based Router Using Network Processors"

November 2001

- Ninth Annual PDL Retreat & Workshop

NEW PDL FACULTY

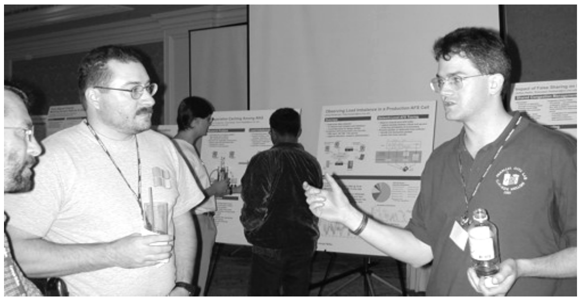
... continued from pg. 3

interaction and security intersect, such as developing interfaces to help ordinary users manage their electronic security and privacy more easily and effectively.

Chris received his Ph.D. in Computer Science from UC Berkeley in 2001. His dissertation focused on a tool for helping designers of pen-based user interfaces create and improve gestures for their interfaces. He has also worked on interfaces for editing digital video, speech interfaces, multimodal interfaces, and virtual reality.

Adrian Perrig

Dr. Adrian Perrig joined the PDL as an Assistant Professor in ECE and Engineering and Public Policy at Carnegie Mellon University. He earned his Ph.D. in Computer Science from Carnegie Mellon University, and spent three years during his Ph.D. with Doug Tygar as his advisor at UC Berkeley, writing his thesis on "Security Protocols for Broadcast Networks." He received his Bachelors degree in Computer Science from the Swiss Federal Institute of Technology in Lausanne (EPFL). Adrian's research interests revolve around building secure systems and include network security, security for sensor networks and mobile applications.

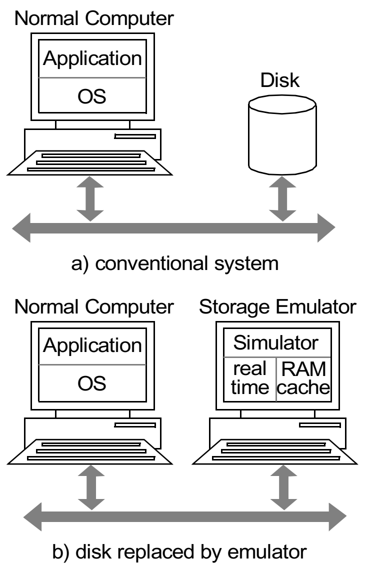


Bruce Worthington (Microsoft) and Greg discuss research during a Retreat poster session.

Timing-Accurate Storage Emulation

Griffin, Schindler, Schlosser & Ganger

Conference on File and Storage Technologies (FAST), January 28-30, 2002. Monterey, CA.



A system with (a) real storage or (b) emulated storage. The emulator transparently replaces storage devices in a real system. By reporting request completions at the correct times, the performance of different devices can be mimicked, enabling full system-level evaluations of proposed storage subsystem modifications.

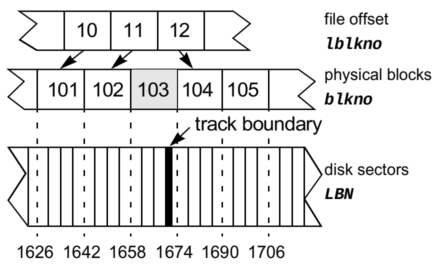
Timing-accurate storage emulation fills an important hole in the set of common performance evaluation techniques for proposed storage designs: it allows a researcher to experiment with not-yet-existing storage components in the context of real systems executing real applications. As its name suggests, a timing-accurate storage emulator appears to the system to be a real storage component with service times matching a simulation model of that component. This paper promotes timing-accurate storage emulation by describing its unique features, demonstrating its feasibility, and illustrating its value. A prototype, called the Memulator, is described and shown to produce service times within 2% of those computed by its component simulator for over 99% of requests. Two sets of measurements enabled by the Memulator illustrate its power: (1) application performance on a modern Linux system equipped with a MEMS-based storage device (no such device exists at this time), and (2) application performance on a modern Linux system equipped with a disk whose firmware has been modified (we have no access to firmware source code).

Track-Aligned Extents: Matching Access Patterns to Disk Drive Characteristics

Schindler, Griffin, Lumb & Ganger

Conference on File and Storage Technologies (FAST) January 28-30, 2002. Monterey, CA.

Track-aligned extents (*traxtents*) utilize disk-specific knowledge to match access patterns to the strengths of modern disks. By allocating and accessing related data on disk track boundaries, a system can avoid most rotational latency and track crossing overheads. Avoiding these overheads can increase disk access efficiency by up to 50% for mid-sized requests (100-500 KB). This paper describes traxtents, algorithms for detecting track boundaries, and the use of traxtents in file systems and video servers. For large file workloads, a modified version of FreeBSD's FFS implementation reduces application run times by 20% compared to the original version. A video server using traxtent-based requests can support 56% more concurrent streams at the same startup latency and buffer space. For LFS, 44% lower overall write cost for track-sized segments can be achieved.



Mapping system-level blocks to disk sectors. Physical block 101 maps directly to disk sectors 1626-1641. Block 103 is an excluded block because it spans the disk track boundary between LBNs 1669-1670.

Capturing the Spatio-Temporal Behavior of Real Traffic Data

Wang, Ailamaki & Faloutsos

Performance 2002 (IFIP Int. Symp. on Computer Performance Modeling, Measurement and Evaluation), Rome, Italy, Sept. 2002.

Traffic, like disk and memory accesses, typically exhibits burstiness, temporal locality and spatial locality. There is much recent ground-breaking work on temporal modeling (self-similarity etc.), on disk and web traffic, with several statistical models that generate realistic series of time-stamps. However, no work generates realistic traces for both time and location (e.g., block-id). In fact, except for qualitative speculations, it is not even known whether/how the time-stamps are correlated with the locations, nor how to measure this correlation, let alone how to reproduce it realistically.

These are exactly the problems we solve here: (a) We propose the 'entropy plots' to quantify the spatial/temporal correlation (or lack of it), and (b) we propose a new model, the 'PQRS' model, that captures all the characteristics of real spatio-temporal traffic. Our model can generate traffic that is bursty (or uniform) on time; bursty or uniform on space; and it can mimic the correlation

... continued on pg. 6

… continued from pg. 5

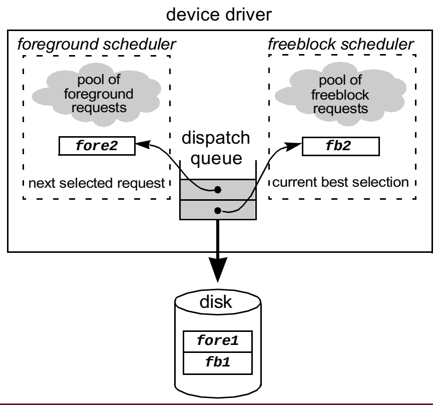
between space and time, whenever such correlation exists. Moreover, it requires very few parameters (p, q, r, and the grand total of disk/memory accesses), and it has linear scalability in computing these parameters. Experiments with multiple real data sets (disk traces from HP Labs, TPC-C memory traces), show that our model can mimic real traces very well, while the only obvious alternative, the independence assumption, leads to more than 60x worse error.

Freeblock Scheduling Outside of Disk Firmware

Lumb, Schindler & Ganger

Conference on File and Storage Technologies (FAST), January 28-30, 2002. Monterey, CA.

Freeblock scheduling replaces a disk drive's rotational latency delays with useful background media transfers, potentially allowing background disk I/O to occur with no impact on foreground service times. To do so, a freeblock scheduler must be able to very accurately predict the service time components of any given disk request – the necessary accuracy was not previously considered achievable outside of disk firmware. This paper describes the design and implementation of a working external freeblock scheduler running either as a user-level application atop Linux or inside the FreeBSD kernel. This freeblock scheduler can give 15% of a disk's potential bandwidth (over 3.1MB/s) to a background disk scanning task with almost no impact (less than 2%) on the foreground request response times. This increases disk bandwidth utilization by over 6x.



Freeblock scheduling inside a device driver.

My Cache or Yours? Making Storage More Exclusive

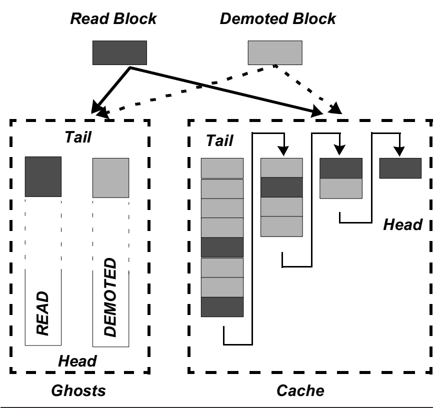
Wong & Wilkes

USENIX Annual Technical Conference (USENIX 2002), June 10-15, 2002, Monterey, CA.

Modern high-end disk arrays often have several giga-bytes of cache RAM. Unfortunately, most array caches use management policies which duplicate the same data blocks at both the client and array levels of the cache hierarchy: they are inclusive. Thus, the aggregate cache behaves as if it was only as big as the larger of the client and array caches, instead of as large as the sum of the two. Inclusiveness is wasteful: cache RAM is expensive.

We explore the benefits of a simple scheme to achieve exclusive caching, in which a data block is cached at either a client or the disk array, but not both. Exclusiveness helps to create the effect of a single, large unified cache. We introduce a DEMOTE operation to transfer data ejected from the client to the array, and explore its effectiveness with simulation studies. We quantify the benefits and overheads of demotions across both synthetic and real-life workloads. The results show that we can obtain useful, sometimes substantial, speedups.

During our investigation, we also developed some new cache-insertion algorithms that show promise for multi-client systems, and report on some of their properties.



Operation of read and demoted ghost caches in conjunction with the array cache. The array inserts the metadata of incoming read (demoted) blocks into the corresponding ghost, and the data into the cache. The cache is divided into segments of either uniform or exponentially-growing size. The array selects the segment into which to insert the incoming read (demoted) block based on the hit count in the corresponding ghost.

Analysis of Methods for Scheduling Low Priority Disk Drive Tasks

Schindler & Bachmat

Proceedings of SIGMETRICS 2002 Conference, June 15-19, 2002, Marina Del Rey, CA.

This paper analyzes various algorithms for scheduling low priority disk drive tasks. The derived closed form solution is applicable to class of greedy algorithms that include 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% short-

… continued on pg. 7

… continued from pg. 6

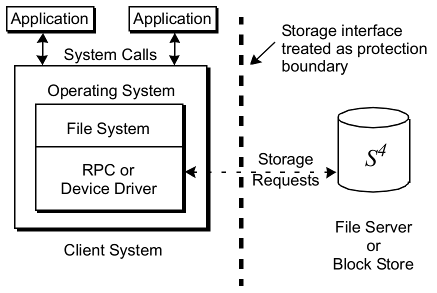
er response time for the foreground application requests and up to a 20% decrease in total background task run time compared to results from previously published techniques.

Intrusion Detection, Diagnosis, and Recovery with Self-Securing Storage

Strunk, Goodson, Pennington, Soules & Ganger

Carnegie Mellon University Technical Report CMU-CS-02-140, May 2002.

Self-securing storage turns storage devices into active parts of an intrusion survival strategy. From behind a thin storage interface (e.g., SCSI or CIFS), a self-securing storage server can watch storage requests, keep a record of all storage activity, and prevent compromised clients from destroying stored data. This paper describes three ways self-securing storage enhances an administrator's ability to detect, diagnose, and recover from client system intrusions. First, storage-based intrusion detection offers a new observation point for noticing suspect activity. Second, post-hoc intrusion diagnosis starts with a plethora of normally-unavailable information. Finally, post-intrusion recovery is reduced to restarting the system with a pre-intrusion storage image retained by the server. Combined, these features can improve an organization's ability to survive successful digital intrusions.



The self-securing storage interface provides a thin perimeter behind which a storage server can observe requests and safeguard data. Note that this same picture works for both block protocols, such as SCSI or IDE/ATA, and distributed file system protocols, such as NFS or CIFS. Thus, self-securing storage could be realized within many storage servers, including file servers, disk array controllers, and even disk drives.

The Set-Check-Use Methodology for Detecting Error Propagation Failures in I/O Routines

Bigrigg & Vos

Workshop on Dependability Benchmarking, in conjunction with the International Conference on Dependable Systems and Networks, DSN-2002; June 23-26, 2002, Washington, D.C.

A methodology is presented that will detect robustness failures in source code where I/O errors could occur and where there is no mechanism in place to handle the error. The details of the methodology are described showing how traditional compiler data flow analysis can be augmented to find structurally, within the application, code that can be used to perform error checking. In addition we describe how this code can be used to ensure the correctness of the I/O error checking

Verifiable Secret Redistribution for Threshold Sharing Schemes

Wong, Wang & Wing

Carnegie Mellon University Technical Report CMU-CS-02-114, February 2002.

We present a new protocol for verifiably redistributing secrets from an (m,n) threshold sharing scheme to an (m',n') scheme. Our protocol guards against dynamic adversaries. We observe that existing protocols either cannot be readily extended to allow redistribution between different threshold schemes, or have vulnerabilities that allow faulty old shareholders to distribute invalid shares to new shareholders. Our primary contribution is that in our protocol, new shareholders can verify the validity of their shares after redistribution between different threshold schemes.

Fractal Prefetching B$^+$-Trees: Optimizing Both Cache and Disk Performance

Chen, Gibbons, Mowry & Valentin

SIGMOD 2002, June 2002, Madison, Wisconsin.

B$^+$-Trees have been traditionally optimized for I/O performance with disk pages as tree nodes. Recently, researchers have proposed new types of B$^+$-Trees optimized for CPU cache performance in main memory environments, where the tree node sizes are one or a few cache lines. Unfortunately, due primarily to this large discrepancy in optimal node sizes, existing disk-optimized B$^+$-Trees suffer from poor cache performance while cache-optimized B$^+$-Trees exhibit poor disk performance. In this paper, we propose fractal prefetching B$^+$-Trees (fpB$^+$-Trees), which embed "cache-optimized" trees within "disk-optimized" trees, in order to optimize both cache and I/O performance. We design and evaluate two approaches to breaking disk pages into cache-optimized nodes: disk-first and cache-first. These approaches are somewhat biased in favor of maximizing disk and cache performance, respectively, as demonstrated by our results. Both implementations of fpB$^+$-Trees achieve dramatically better cache performance than disk-optimized

… continued on pg. 17

FRACTAL PREFETCHING B^+-TREES

Shimin Chen, Todd Mowry & Joan Digney

Fractal Prefetching B^+-Trees (fpB^+-Trees) are a type of B^+-Tree that optimize both cache and I/O performance by embedding "cache-optimized" trees within "disk-optimized" trees. This improves CPU cache performance in traditional B^+-Trees for indexing disk resident data and I/O performance in B^+-Trees optimized for cache. At a coarse granularity an fpB^+-Tree contains disk-optimized nodes that are roughly the size of a disk page; at a fine granularity, it contains cache-optimized nodes that are roughly the size of a cache line. The fpB^+-Tree is referred to as "fractal" because of its self-similar "tree within a tree" structure, as illustrated in Figure 1.

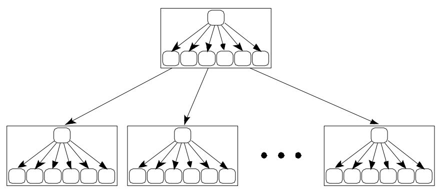


Figure 1: Self-similar "tree within a tree" structure.

Optimizing I/O Performance

One goal of fpB^+-Trees is to effectively exploit I/O parallelism by explicitly prefetching disk pages even when the access patterns are not sequential. Prefetching B^+-Trees (pB^+-Trees) are a proven technique for enhancing CPU cache performance for index searches and range scans on memory-resident data, but can they be applied to improving I/O performance for disk-resident data?

All nodes within a pB^+-Tree are multiple cache lines wide. To accelerate search performance, the pB^+-Tree prefetches all cache lines within a node before accessing it. Thus, multiple cache misses may be serviced in parallel, resulting in small overall miss penalties. The net result is that searches become faster because nodes are larger and trees are shallower. To apply this principle to disk-resident data, all pages of a node are prefetched when accessing it. By placing the pages that make up a node on different disks, multiple page requests can be serviced in parallel. While the I/O latency is likely to improve for a single search, I/O throughput may suffer because of the extra seeks for a node. Hence the target node size for optimizing disk performance of fpB^+-Trees is a single disk page.

Range scans are performed by searching for the starting key of a range, then reading consecutive leaf nodes in the tree until the end key for the range is encountered. To enhance range scan cache performance, a jump-pointer array scheme, containing the leaf node addresses of the tree used in range scans, is employed for prefetching the leaf nodes. By issuing a prefetch for each leaf node sufficiently in advance of when the range scan needs the node, the cache misses of these leaves are overlapped. The same technique can improve range scan I/O performance at page granularity, overlapping leaf page misses. It is particularly helpful in non-clustered indexes and when leaf pages are not sequential on disks. To prevent prefetching past the end key, fpB^+-Trees begin by searching for both the start key and the end key, remembering the range end page. This approach is applicable for improving the I/O performance of standard B^+-Trees, not just fractal trees, and can lead to a five-fold or more speedup for large range scans.

Optimizing Cache Performance

fpB^+-Trees can be implemented as disk-first or cache-first. The disk-first approach begins with a disk-optimized B^+-Tree, and organizes the keys and pointers within each page-sized node as a small tree. To pack more keys and pointers into the in-page tree, short in-page offsets rather than full pointers in all but the leaf nodes of the tree are used. The cache-first approach begins with a cache-optimized prefetching B^+-Tree and, ignoring disk page boundaries, seeks to place parent and child nodes on the same page. Adjacent leaf nodes are also placed on the same page. Ideally, both the disk-first and the cache-first approaches would achieve identical data layouts, and hence equivalent cache and I/O performance. In practice, however, mismatching almost always occurs between the size of a cache-optimized subtree and the size of a disk page causing the two approaches to be slightly biased in favor of disk and cache performance, respectively. Despite these slight disparities, both implementations of fpB^+-Trees achieve better cache performance than disk-optimized B^+-Trees.

Disk-First fpB^+-Trees start with a disk-optimized B^+-Tree, where page-sized nodes containing keys and pointers are organized into a cache-optimized tree called an in-page tree. Each node in an in-page tree is aligned on cache line boundaries and is several cache lines wide. When accessed in a search, all the cache lines comprising the node are prefetched. Disk-first fpB^+-Trees have both leaf and nonleaf in-page nodes. The nonleaf nodes contain pointers to other in-page nodes within the same page, and in-page leaf nodes contain pointers to nodes external to their in-page tree.

If considering cache performance only, there is an optimal in-page node size, calculated based on the relationships between the number of levels in the in-page tree, the number of cache lines of the nonleaf nodes and the number of cache lines of the leaf nodes. Ideally, in-page trees based on this optimal size would fit

... continued on pg. 9

… continued from pg. 8

tightly within a page. However, since optimal page size is determined by I/O parameters and disk and memory prices, there is likely a mismatch between the two sizes and it is recognized that in most cases, trees with cache-optimal node sizes are not possible. To combat overflow, the root node size can be reduced (or its fan-out restricted). Similarly, to combat underflow, the root node may be extended so that it can have more children.

Cache-First fpB$^+$-Trees begin with a cache-optimized B$^+$-Tree, and, ignoring page boundaries, try to intelligently place the cache-optimized nodes into disk pages. The tree node has the common structure of a cache-optimized B$^+$-Tree node: a leaf node contains an array of keys and tuple IDs, while a nonleaf node contains an array of keys and pointers. However, the pointers in nonleaf nodes are different. Since the nodes are to be put into disk pages, a pointer is a combination of a page ID and an offset in the page, which allows us to follow the page ID to retrieve a disk page and then visit a node in the page by its offset.

There are two goals in node placement within a disk page to minimize the structure's impact on I/O performance: (1) group sibling leaf nodes together into the same page so that range scans incur fewer disk operations, and (2) group a parent node and its children together into the same page so that searches only need one disk operation for a parent and its child. To satisfy the first goal, certain pages are designated as leaf pages, and contain only leaf nodes. Leaf nodes in the same leaf page are siblings of one another, ensuring good range scan I/O performance. The second goal cannot be satisfied for all nodes, because only a limited number of nodes fit within a page. Moreover, the node size mismatch problem means that placing a parent and its children in a page almost always results in either an overflow or an underflow for that page. Large underflow situations can be transformed by placing the grandchildren, the great grandchildren, and so on in the same page, until either a modest underflow or an overflow is incurred. There are two approaches for dealing with the overflow: an overflowed child can be placed into its own page to become the top-level node in that page and have its children placed in the same page, or it can be stored in a special overflow page.

There are several fundamental tradeoffs between the disk-first and the cache-first implementations of fpB$^+$-Trees. While the performance of each of these implementations remains slightly biased toward its original goal, both versions of fpB$^+$-Trees improve upon the cache performance of disk-optimized B$^+$-Trees (without significantly degrading I/O performance) as follows: (i) a factor of 1.1-1.8 improvement for search; (ii) up to a factor of 4.2 improvement for range scans; and (iii) up to a 20-fold improvement for updates. fpB$^+$-Trees can also be used to accelerate I/O performance. In particular, an over twofold to fivefold improvement for index range scans was demonstrated in an industrial-strength commercial DBMS (IBM's DB2). More information on the experimental procedures used to arrive at our conclusions, on the algorithms used in the creation of the fpB$^+$-Tree indexes and on performance in typical operations of the tree, such as bulkload, search, insertion and deletion, in both disk first and cache first implementations, is available elsewhere [1].

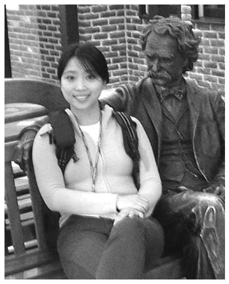
Conclusions

Previous studies on improving index performance have focused either on optimizing the cache performance of memory-resident databases, or on optimizing the I/O performance of disk-resident databases. What has been lacking prior to this study is an index structure that achieves good performance for both of these important levels of the memory hierarchy. Our experimental results demonstrate that Fractal Prefetching B$^+$-Trees, a novel index structure that optimizes both cache and disk performance simultaneously, are such a solution. They achieve large gains in cache performance compared with disk-optimized B$^+$-Trees for searches, range scans, and updates on modern systems. Moreover, they provide up to a fivefold improvement in the I/O performance of range scans on a commercial DBMS (DB2).

References

[1] Chen, S., Gibbons, P.B., Mowry, T.C. and Valentin, G. Fractal Prefetching B+Trees: Optimizing Both Cache and Disk Performance. *In Proceedings of SIGMOD 2002*, June 2002, Madison, Wisconsin.

[2] Chen, S., Gibbons, P.B. and Mowry, T.C. Improving Index Performance through Prefetching. *In Proceedings of SIGMOD 2001*, May 2001, pp. 235-246.



PDL students and faculty alike visit Mark Twain for inspiration

September 2002

Mengzhi Wang Receives Best Student Paper Award at Performance 2002

Mengzhi Wang's paper, "Capturing the Spatio-Temporal Behavior of Real Traffic Data," co-authored with Anastassia Ailamaki and Christos Faloutsos, received the Best Student Paper Award at the 22nd IFIP WG 7.3 Int'l Symposium on Computer Modeling, Measurement and Evaluation, held in Rome, Italy, Sept. 23-27.

July 2002

Congratulations Natassa and Babak!

Congratulations to Natassa Ailamaki and Babak Falsafi (Assistant Professor of ECE and CS) who were married on July 7 in Chania, Greece. The rest of the members of the PDL wish them all the best for many years to come!

July 2002

Greg Promoted

We are pleased to congratulate Greg on his promotion to Associate Professor this year.

June 2002

Welcome to the Newest Member of the Ganger Family!

Greg, Jenny and Tim Ganger are thrilled to announce the arrival of

Will Ganger - 6 days old!

William David Ganger, born at 11:46 am on June 30, 2002 (a few days earlier than planned). William weighed in at 7 lbs. 5 oz. and at birth was 20.5 inches.

June 2002†

New Center for Computer and Communications Security

Carnegie Mellon researchers have formed a Center for Computer and Communications Security (C3S) to tackle the challenges and problems related to Internet security, data storage and privacy issues stemming from America's ongoing war against terrorism.

The center is multidisciplinary with faculty coming from Electrical and Computer Engineering, the CERT/CC, Engineering and Public Policy, the School of Computer Science, the Statistics Department, and the Heinz School of Public Policy.

Pradeep Khosla, ECE Department Head; Philip and Marsha Dowd Professor of ECE and Robotics, director of the C3S, said although security technology is advancing, the Internet is still susceptible to viruses, computer intrusions and cyberterrorism. The new center will focus on cutting-edge technologies related to security in distributed systems and wireless and optical networks as well as new technologies to guarantee the privacy of information.

April 2002*

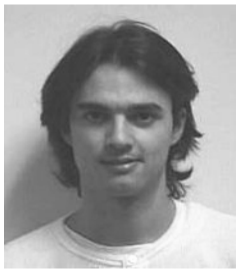
PDL Makes the Headlines

On April 10, C|net News published a press release outlining the use of medieval castle architecture by Greg Ganger and the PDL as the inspiration for an innovative approach to computer security. This approach has self-securing devices erecting their own security perimeters and defending their own critical resources just the way individual parts of medieval castles formed distinct protective barriers, such as moats, inner sanctums, and strategically placed guard towers. The Pittsburgh Post-Gazette and WPXI also visited Greg and the PDL to talk about computer security innovations.

April 2002

PDL Student Receives IBM Ph.D. Fellowship

Stavros Harizopoulos (CS, advised by Anastassia Ailamaki) has been awarded a prestigious IBM Ph.D. Fellowship for 2002/2003. These competitive awards recognize "outstanding research and technical excellence in areas of interest to IBM" and provide a stipend, tuition and fees, in addition to an opportunity to pursue technical careers in IBM's Research Division or development laboratories.



March 2002*

Ailamaki and Harchol-Balter Receive NSF Career Awards

Anastassia Ailamaki and Mor Harchol-Balter have each been awarded a National Science Foundation CAREER Award. This prestigious program recognizes and supports the early career development of "young

... continued on pg. 11

… continued from pg. 10

faculty members...most likely to become the academic leaders of the 21st century." Selection is made on the basis of creative, integrative, and effective research and education career development plans that build a firm foundation for a lifetime of integrated contributions to research and education. Anastassia's research focuses on "Bridging Databases and Computer Architecture: Optimizing DBMS for Deep Memory Hierarchies", while Mor explores "The Impact of Resource Scheduling on Improving Server Performance."

February 2002

PDL paper named Best Student Paper at FAST 2002

The program committee of the USENIX Conference on File and Storage technologies (FAST'02) presented the Best Student Paper Award to PDL researchers Jiri Schindler, John Linwood Griffin, Christopher R. Lumb, and Gregory R. Ganger for their paper "Track-Aligned Extents: Matching Access Patterns to Disk Drive Characteristics." The conference included 21 papers (three of which were PDL submissions) chosen from a pool of 110 submissions. Jiri, John and Chris also each gave talks on their research at the conference.

January 2002**

PDL Graduate Student Awarded Microsoft Research Fellowship

Microsoft Corporation has chosen Shimin Chen, a CS Ph.D. student, to receive a Microsoft Research Fellowship. Awarded to 13 of 52 applicants, the fellowship offers financial support for two years, including 100 percent of CMU tuition and fees; a stipend for living expenses of up to $20,000; a conference and travel allowance; a laptop computer complete with Microsoft software; and a $1,000 donation to the students' advisor, Todd Mowry, Associate Professor of CS and ECE. Chen also has the opportunity to participate in a 12-week paid internship, allowing him to interact with Microsoft Researchers and work in areas relevant to his own research.

January 2002

Intel Equipment Grant

The PDL would like to thank Intel Corporation for its generous donation of equipment in support of our research. The PDL received 3 fully equipped 1.7 GHz WS530 Xeon DP Workstations and 100 PIII 850 MHz boxed processors. Included with the donation is three years of product support.

December 2001*

PDL Student receives Honorable Mention in CRA Outstanding Undergraduate Awards

Cory Williams, a CS/Math Sciences senior and PDL member, received Honorable Mention when the Computing Research Association selected the recipients of their Outstanding Undergraduate Awards for 2002. Nominees were from universities across North America and it is a significant honor for Cory to have been selected for honorable mention from this group.

Cory's work focuses on Computer forensics and Intrusion detection, and the benefit achieved if system logs continued to be accurately recorded after a system compromise. Specifically, he is working on how to use these accurately recorded system logs and what should be recorded if accurate logging is expected.

November 2001

Congratulations to CMU's ACM Programming Contest Winners

A CMU team consisting of Cory Williams (PDL), Tom Murphy and Eric Heutchy received 4th place in the East Central North American Region in the 2001 ACM programming contest. In regional competition, they competed at Ashland University, where they placed first.

November 2001*

Goldstein Participates in ICCAD 2001 Nanotechnology Panel

Seth Goldstein, Assistant Professor of Computer Science and ECE, was one of six panelists to address the question "Will Nanotechnology Change the Way We Design and Verify Systems?" at the International Conference on Computer-Aided Design panel session on November 7. The panel was part of a conference for EE CAD professionals, held in San Jose, CA.

Goldstein predicted that nanotechnology systems would be reprogrammable and designers would use nanotechnology chips' reconfigurability to detect and avoid defects.

*SCS Today
**ECE News
†CMU 8 1/2 x 11 News

Andy concentrating hard on his research.

… continued from pg. 1

the available time. The first Retreat was held at the Hidden Valley Resort in Pennsylvania; for the past 6 years, we have gathered at the beautiful Nemacolin Woodlands Resort, in Farmington, Pennsylvania.

PDL's initial seed funding came from CMU's Data Storage Systems Center (DSSC), then directed by Mark Kryder, and from DARPA (from which most PDL funding has come over the years). Additional funding came from the member companies of the PDL Consortium, whose initial members were AT&T Global Information Systems, Data General, IBM, Hewlett-Packard, Seagate and Storage Technology. Today, PDL funding comes from DARPA, NSF, the Air Force Office of Sponsored Research, and the PDL Consortium members (listed on the front page).

In 1995, Gibson and Dr. David Nagle (then a new ECE faculty member) launched a new PDL project called Network-Attached Secure Disks (NASD). NASD was a new network-attached storage architecture for achieving cost-effective scalable bandwidth. In addition to their fundamental research advances, Gibson founded and chaired an industry working group within the National Storage Industry Consortium (NSIC) to transfer and move towards standardization of the NASD architecture. In '99, working group members produced a concrete proposal to launch an ANSI standards effort around object-based storage devices (with essentially the NASD architecture). Since its start, the NASD project has stimulated much derivative research and development in academia and industry.

Many lasting friendships have been formed.

The PDL in 1996.

In 1999, Nagle took over as PDL Director when Gibson went on leave. In 2000, Greg Ganger, who joined the ECE faculty and the PDL in 1997, jointly directed the PDL with Nagle, then became Director in 2001 when Nagle went on leave. In its ten-year lifetime, many faculty, staff, and students from both CS and ECE have been active members of the PDL. PDL's first Ph.D. graduate was Dr. Mark Holland (1994), who wrote his dissertation on "On-Line Data Reconstruction in Redundant Disk Arrays." Since then, 11 PDL students have graduated with Ph.D.s, 20 others with Masters degrees, and more than a dozen with undergraduate degrees, many of whom have moved on to employment with PDL Consortium companies. There are currently 30 PDL students.

From the beginning, the PDL logo has included Skibo Castle, Andrew Carnegie's summer home. In the past, it has represented "a fortress of storage" (like a redundant disk array). More recently, it represents a "fortress of security" (à la self-securing devices). Perhaps, though, it is simply our vision of the ideal PDL Retreat venue.

Eleven Ph.Ds, twenty Masters and over a dozen undergraduate degrees have been granted to PDL members in the PDL's first nine years.

Greg Ganger & Joan Digney

Digital information is a critical resource. We need storage systems to which users can entrust critical information, ensuring that the data persists, is accessible, cannot be destroyed, and is kept confidential. A survivable storage system provides these guarantees, despite failures and malicious compromises of storage nodes, client systems, and user accounts. This article overviews two PDL projects that address this need: the PASIS project focuses on surviving attacks on storage servers, and self-securing storage focuses on surviving intrusions into client systems.

PASIS:

Survivable systems operate from the fundamental design thesis that no individual service, node, or user can be fully trusted; having some compromised entities is viewed as a common case rather than an exception. Survivable storage systems must therefore encode and distribute data across independent storage nodes, entrusting data persistence to sets of nodes rather than to individual nodes. Further, if confidentiality is required, unencoded data should not be stored directly on individual storage nodes; otherwise, compromising a single storage node would let an attacker bypass access-control policies. With well-chosen encoding and distribution schemes, significant increases in availability, confidentiality, and integrity are possible.

Many research groups now explore the design and implementation of such survivable storage systems. These systems build on mature technologies from decentralized storage systems and also share a common high-level architecture (Figure 1). In fact, development of survivable storage with the same basic architecture was pursued over 15 years ago. As it was then, the challenge now is to achieve acceptable levels of performance and manageability. Moreover, a means to evaluate survivable storage systems is needed.

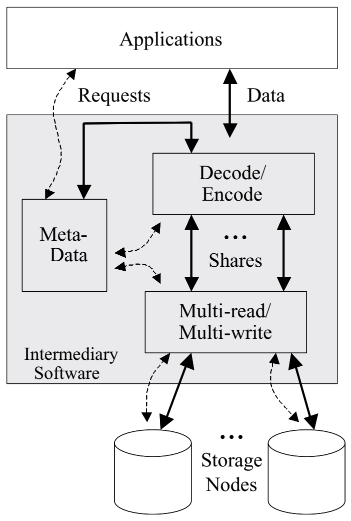


Figure 1: *Generic decentralized storage architecture.* Intermediary software translates the applications' unified view of storage to the decentralized reality. Encoding transforms blocks into shares (decoding does the reverse). Sets of shares are read from (written to) storage nodes. Intermediary software may run on clients, leader storage nodes, or at some point in between.

One key to maximizing survivable storage performance is mindful selection of the data distribution scheme. A data distribution scheme consists of a specific algorithm for data encoding & partitioning and a set of values for its parameters. There are many algorithms applicable to survivable storage, including encryption, replication, striping, erasure-resilient coding, secret sharing, and various combinations. Each algorithm has one or more tunable parameters, such as the number of fragments generated during a write and the subset needed for a read. The result is a large toolbox of possible schemes, each offering different levels of performance (throughput), availability (probability that data can be accessed), and security (effort required to compromise the confidentiality or integrity of stored data). For example, replication provides availability at a high cost in network bandwidth and storage space, whereas short secret sharing provides availability and security at lower storage and bandwidth cost but higher CPU utilization. Likewise, selecting the number of shares required to reconstruct a secret-shared value involves a trade-off between availability and confidentiality: if more machines are compromised to steal a secret, then more must be operational to provide it legitimately.

No single data distribution scheme is right for all systems. Instead, the right choice for any particular system depends on an array of factors, including expected workload, system component characteristics, and desired levels of availability and security. Unfortunately, most system designs appear to involve an ad hoc choice, often resulting in a substantial performance loss due to missed opportunities and over-engineering.

The PASIS project is developing a better approach to selecting the data distribution scheme. At a high level, this new approach consists of three steps: enumerating possible data distribution schemes (<algorithm, parameters> pairs), modeling the consequences of each scheme, and identifying the best-performing scheme for any given set of availability and security requirements. The surface shown in Figure 2 illustrates one result of the approach. Generating such a surface requires codifying each dimension of the trade-off space such that all data distribution schemes fall into a total order. The surface serves two functions: (1) it enables informed trade-offs among security, availability, and performance; and (2) it identifies the best-performing scheme for each point in the trade-off space. Specifically, the surface shown represents the performance of the best-performing scheme that provides at least the corresponding levels of availability and security. Many schemes are not best at any of the points in the space and, as such, are not visible on the surface.

... continued on pg. 14

… continued from pg. 13

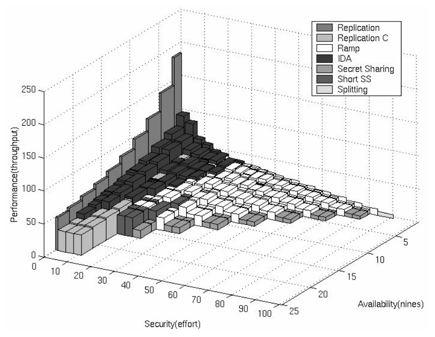


Figure 2: Data distribution scheme selection surface plotted in trade-off space.

Wylie et al. [2] demonstrate the feasibility and importance of careful data distribution scheme choice. The results show that the optimal choice varies as a function of workload, system characteristics, and the desired levels of availability and security. Minor (~2x) changes in these determinants have little effect, which means that the models need not be exact to be useful. Large changes, which would correspond to distinct systems, create substantially different trade-off spaces and best choices. Thus, failing to examine the trade-off space in the context of one's system can yield both poor performance and unfulfilled requirements.

Of course, the research is not done. We continue to refine the configuration approach, exploring useful ways to approximate security metrics, and to push the boundaries of efficient decentralization.

Self-Securing Storage:

Desktop compromises and misbehaving insiders are a fact of modern computing. Once an intruder infiltrates a system, he can generally gain control of all system resources, including its storage access rights (complete rights, in the case of an OS accessing local storage). Crafty intruders can use this control to hide their presence, weaken system security, and manipulate sensitive data. Because storage acts as a slave to authorized principals, evidence of such actions can generally be hidden. In fact, so little of the system state is trustworthy after an intrusion that the common "recovery" approach starts with reformatting storage.

Self-securing storage is an exciting new technology for enhancing intrusion survival by enabling the storage device to safeguard data even when the client OS is compromised. It capitalizes on the fact that storage servers (whether file servers, disk array controllers, or even IDE disks) run separate software on separate hardware. This opens the door to server-embedded security that cannot be disabled by any software (even the OS) running on client systems as shown in Figure 3. Of course, such servers have a narrow view of system activity, so they cannot distinguish legitimate users from clever impostors. But, from behind the thin storage interface, a self-securing storage server can actively look for suspicious behavior, retain an audit log of all storage requests, and prevent both destruction and undetectable tampering of stored data. The latter goals are achieved by retaining all versions of all data; instead of over-writing old data when a write command is issued, the storage server simply creates a new version and keeps both. Together with the audit log, the server-retained versions represent a complete history of system activity from the storage system's point of view.

Strunk et al. [3] introduced self-securing storage and evaluated its feasibility. It was demonstrated that, under a variety of workloads, a small fraction of the capacity of modern disk drives is sufficient to hold several weeks of complete storage history. With a prototype implementation, it was also demonstrated that the performance overhead of keeping the complete history is small. Two recent papers, listed elsewhere in this newsletter, delve more deeply into how self-securing storage can improve intrusion survival by safeguarding stored data and providing new information regarding storage activities before, during, and after the intrusion. Specifically, self-securing storage contributes in three ways:

First, a self-securing storage server can assist with intrusion detection by watching for suspicious storage activity. By design, a storage server sees all requests and stored data, so it can issue alerts about suspicious storage activity as it happens. Such storage-based intrusion detection can quickly and easily notice several common intruder actions, such as manipulating system utilities (e.g., to add backdoors) or tampering with audit log contents (e.g., to conceal evidence). Such activities are exposed to the storage system even when the client system's OS is compromised.

Second, after an intrusion has been detected and stopped, self-securing storage provides a wealth of information to security administrators who wish to analyze an intruder's actions. In current systems, little information is available for estimating

… continued on pg. 16

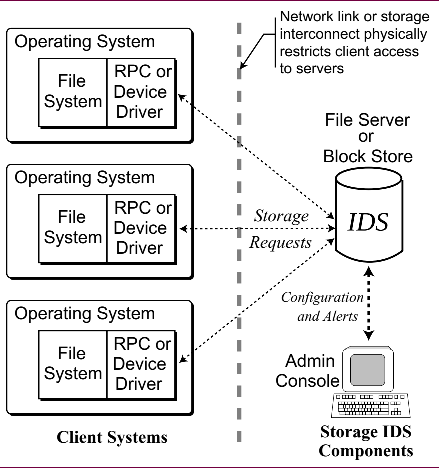


Figure 3: *The compromise independence of self-securing storage.* The storage interface provides a physical boundary between a storage server and client OSes. Note that this samd picture works for block protocols, such as SCSI or IDE/ATA, and distributed file system protocols such as NFS or CIFS.

THESIS PROPOSAL:

A System for Matching Application Resource Supply and Demand

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, May 22, 2002.

David Petrou, ECE

My thesis will show how to ease the management of several classes of adaptive applications. The motivation is that many important applications offer the user a stupefying number of parameters, like whether a data mining workload should be function- or data-shipping, the resolution of a graphics renderer, the sizes of the caches in a web browser, etc. Further, when running more than one application, the user has the additional opportunity (or burden) of deciding how resources should be allocated among running applications. My work will demonstrate situations in which these decisions can be automated.

THESIS PROPOSAL:

Matching Access Patterns to Storage Device Characteristics

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, June 24, 2002.

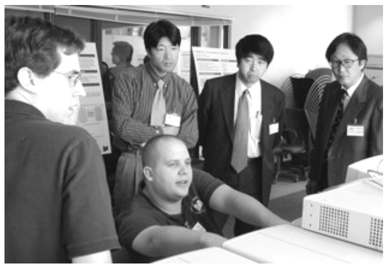
Jiri Schindler, ECE

While both operating systems (OSes) and storage devices are complex systems with advanced algorithms attempting to improve I/O performance, there is very little or no communication between the two about device strengths and weaknesses. As a result, both systems make decisions in isolation that do not realize the full potential of the storage device's capabilities. I propose to investigate what information about device's characteristics, and in what format, should be communicated so that the OS can appropriately adjust application access patterns. These modified access patterns will enable the storage device to service the requests much more efficiently.

As part of my research, I want to identify a minimal set of attributes that can effectively describe characteristics of various classes of storage devices (e.g., disk drives or high-end storage arrays). These attributes should not include any storage-specific or proprietary information that would break the model of a single storage manager controlling different device classes. Finally, I will demonstrate the benefits of this approach on two concrete examples: a block-based file system (e.g., the FreeBSD implementation of FFS) and query evaluation inside a database system.

PDL SPRING OPEN HOUSE

Open House poster session



Jay demonstrates his research to Sony

5 neat guys

Greg introduces self-securing devices to the media

SURVIVABLE STORAGE SYSTEMS

... continued from pg. 14

damage (e.g., what information the intruder might have seen or what data was modified) and determining how he gained access. Because intruders can directly manipulate data and metadata in conventional systems, they can remove or obfuscate traces of such activity. With self-securing storage, intruders lose this ability–in fact, attempts to do these things become obvious red flags for intrusion detection and diagnosis efforts. Although technical challenges remain in performing such analyses, they will start with much more information than forensic techniques can usually extract from current systems.

Third, self-securing storage can speed up and simplify the intrusion recovery process. In today's systems, full recovery usually involves reformatting, reinstalling the OS from scratch, and loading user data from back-up tapes. These steps are taken to remove backdoors or Trojan horses that may have been left behind by the intruder. Given server-maintained versions, on the other hand, an administrator can simply copy-forward the pre-intrusion state (both system binaries and user data) in a single step. Further, all work done by the user since the security breach remains in the history pool, allowing incremental (albeit potentially dangerous) recovery of important data.

In continuing work, we are exploring administrative interfaces for configuring and utilizing the features of self-securing storage. We will also explore how they complement security functionality embedded in other devices, such as network interface cards and network switches/routers.

References

[1] Wylie, J., et al. Survivable Information Storage Systems. IEEE Computer, Aug 2000.

[2] Wylie, J., et al. Selecting the Right Data Distribution Scheme for a Survivable Storage System. CMU SCS Technical Report CMU-CS-01-120, May 2001.

[3] Strunk, J.D., et al. Self-Securing Storage: Protecting Data in Compromised Systems. Proc. of the 4th Symposium on Operating Systems Design and Implementation, October, 2000.

COMINGS & GOINGS

FACULTY

Three new faculty members have joined the PDL this academic year. Please see brief biographies for Chris Long, Adrian Perrig and Dawn Song beginning on page 3.

STAFF

Stan Bielski joined the PDL staff as a systems programmer in May, following his graduation from Penn State University with a B.S. in Computer Engineering.

Semih Oguz left his position as a research programmer to accompany his wife to California where she has taken up a faculty position at Stanford. Semih himself is visiting his family in Turkey before beginning his search for employment.

Before joining the PDL as office assistant to Greg, Mike and Chenxi, Linda Whipkey worked for the Hillman Company in Pittsburgh, where she worked at the Help Desk, maintained their computer inventory and managed their software contracts.

GRAD STUDENTS

Mehmet Bakkaloglu completed his Master's research and submitted his thesis entitled "On Correlated Failures in Survivable Storage Systems" in May. He is now at IBM.

Angela Demke Brown is now an Assistant Professor of Computer Science at the University of Toronto in Toronto, Ontario, Canada.

David Friedman completed his Master's degree in ECE and is now working at Oracle.

James Hendricks comes to us from UC Berkeley. He is pursuing his Ph.D. in Computer Science and will be researching intrusion-tolerant software with Greg and Adrian.

Mike Mesnier, on educational sabbatical from Intel, joined the PDL as a graduate student in August. Greg is advising his research on object-based storage and iSCSI.

Vijay Pandurangan has completed his Masters Degree in ECE and is now working with Google.

Adam Pennington began his M.S. in ECE this spring. He is working on self-securing storage with Greg.

Brandon Salmon is beginning work on his M.S. with Greg, working on continuous reorganization. He joins us from Stanford University.

Eno Thereska has joined the PDL to begin his Masters Degree in ECE. He will be working with Greg Ganger on freeblock scheduling and continuous reorganization.

Monica Ullagaddi joined the PDL as an undergraduate programmer and this fall is beginning her Master's degree in ECE, working on self-securing NICs with Greg.

UNDERGRADUATES

Two new undergrads, Chris Costa and Vinod Das Krishnan, joined us over the summer. Greg is advising Chris on AFS tracing and Vinod on freeblock scheduling. Cory Williams has completed his degrees and has moved on to Microsoft. Russ Koenig is focusing on his studies in ECE.

… continued from pg. 7

B$^+$-Trees: a factor of 1.1-1.8 improvement for search, up to a factor of 4.2 improvement for range scans, and up to a 20-fold improvement for updates, all without significant degradation of I/O performance. In addition, fpB$^+$-Trees accelerate I/O performance for range scans by using jump-pointer arrays to prefetch leaf pages, thereby achieving a speed-up of 2.5-5 on IBM's DB2 Universal Database.

Blurring the Line Between OSes and Storage Devices

Ganger

Carnegie Mellon University Technical Report CMU-CS-01-166, December 2001.

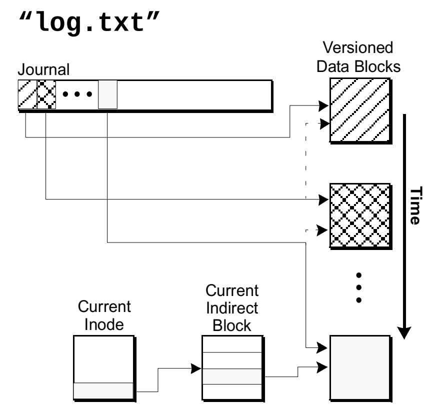
This report makes a case for more expressive interfaces between operating systems (OSes) and storage devices. In today's systems, the storage interface consists mainly of simple read and write commands; as a result, OSes operate with little understanding of device-specific characteristics and devices operate with little understanding of system priorities. More expressive interfaces, together with extended versions of today's OS and firmware specializations, would allow the two to cooperate to achieve performance and functionality that neither can achieve alone.

Metadata Efficiency in a Comprehensive Versioning File System

Soules, Goodson, Strunk & Ganger

Carnegie Mellon University Technical Report CMU-CS-02-145, May 2002.

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.



Journal-based metadata system. This figure shows a single logical block of the file "log.txt" being overwritten several times. Journal-based metadata retains all versions of the data block. However, each block is tracked using journal entries. Each entry points to both the new block and the block that was overwritten. Only the current version of the inode and indirect block are kept, significantly reducing the amount of space required for metadata.

Decentralized Storage Consistency via Versioning Servers

Goodson, Wylie, Ganger & Reiter

Carnegie Mellon University Technical Report CMU-CS-02-180, September 2002.

This paper describes a consistency protocol that exploits versioning storage nodes. The protocol provides linearizability with the possibility of read aborts in an asynchronous system that may suffer client and storage-node crash failures. The protocol supports both replication and erasure coding (which precludes posthoc repair of partial-writes), and avoids the excess work of two phase commits. Versioning storage-nodes allow the protocol to avoid excess communication in the common case of no write sharing and no failures of writing clients.

Delegation of Cryptographic Servers for Capture-Resilient Devices

MacKenzie & Reiter

DIMACS Technical Report 2001-37, November 2001. DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Inst. and Telcordia Technologies.

A device that performs private key operations (signatures or decryptions), and whose private key operations are protected by a password, can be immunized against offline dictionary attacks in case of capture by forcing the device to confirm a password guess with a designated remote server in order to perform a

… continued on pg. 18

... continued from pg. 17

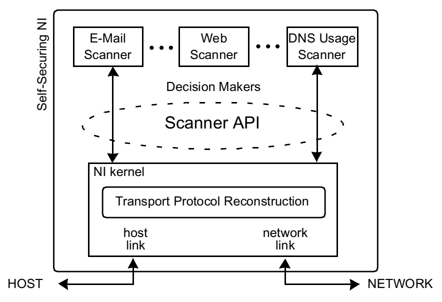
private key operation. Recent proposals for achieving this allow untrusted servers and require no server initialization per device. In this paper we extend these proposals to enable dynamic delegation from one server to another; i.e., the device can subsequently use the second server to secure its private key operations. One application is to allow a user who is traveling to a foreign country to temporarily delegate to a server local to that country the ability to confirm password guesses and aid the user's device in performing private key operations, or in the limit, to temporarily delegate this ability to a token in the user's possession. Another application is proactive security for the device's private key, i.e., proactive updates to the device and servers to eliminate any threat of offline password guessing attacks due to previously compromised servers.

Self-Securing Network Interfaces: What, Why and How

Ganger, Economou & Bielski

Carnegie Mellon University Technical Report CMU-CS-02-144, May 2002.

Self-securing network interfaces (NIs) examine packets as they move between network links and host software, looking for and potentially blocking malicious network activity. This paper describes self-securing network interfaces, their features, and examples of how these features allow administrators to more effectively spot and contain malicious network activity. We present a software architecture for self-securing NIs that separate scanning software into applications (called scanners) running on an NI kernel. The resulting scanner API simplifies the construction of scanning software and allows its powers to be contained even if it is subverted. We illustrate



Self-securing NI software architecture. An "NI kernel" manages the host and network links. Scanners run as application processes. Scanner access to network traffic is limited to the API exported by the NI kernel.

the potential via a prototype self-securing NI and two example scanners: one that identifies and blocks known e-mail viruses and one that identifies and inhibits rapidly-propagating worms, e.g. Code-Red.

Examining Semantics In Multi-Protocol Network File Systems

Hogan, Gibson & Ganger

Carnegie Mellon University Technical Report CMU-CS-02-103, January 2002.

Network file systems provide a robust tool that can be used by many physically dispersed clients. They provide clients with a means of permanent storage and communication. In order to exploit the resources available on a network file system server, a client must use the protocol of the server's file system. Although the goal of any protocol is to guarantee that the client and server can communicate, the introduction of new protocols divides clients into incompatible sets. Soon clients can no longer cooperate and share because they are using different protocols. In addition, each network file system is constructed with a different set of semantics. The result is that it is increasingly difficult to provide a single storage solution that supports all of these clients. Although difficult, it is extremely desirable to build a multi-protocol network file system, that is, a storage solution that can be used simultaneously by clients of different protocols and semantic sets. A semantic mismatch is a major complexity in building a multi-protocol network file system. These are situations that arise when the normal behavior of a server, expected by a client using a particular semantic set, does not occur because of the effects of a client from a separate semantic set. To achieve the goal of building a multi-protocol file system, the file system semantic sets of the targeted file systems must be carefully examined to determine where semantic mismatches will occur. Next, the possible means of resolving a semantic mismatch can be analyzed for their particular tradeoffs. Finally, data from file system traces can be used to determine the frequency of possible semantic mismatches. The data collected from the file system traces, when examined in the context of a cost-benefit analysis, can provide designers of multi-protocol network file systems with important information for examining and resolving semantic differences.

Exploring Congestion Control

Akella, Seshan, Shenker & Stoica

Carnegie Mellon University Technical Report CMU-CS-02-139, May 2002.

From the early days of modern congestion control, ushered in by the development of TCP's and DECbit's congestion control algorithm and by the pioneering theoretical analysis of Chiu and Jain, there has been widespread agreement that linear additive-increase-multiplicative-decrease (AIMD) control algorithms should be used. However, the early congestion control design deci-

... continued on pg. 19

… continued from pg. 18

sions were made in a context where loss recovery was fairly primitive (e.g. TCP Reno) and often timed-out when more than a few losses occurred and routers were FIFO drop-tail. In subsequent years, there has been significant improvement in TCP's loss recovery algorithms. For instance, TCP SACK can recover from many losses without timing out. In addition, there have been many proposals for improved router queueing behavior. For example, RED active queue management and Explicit Congestion Notification (ECN) can tolerate bursty ow behavior. Per-flow packet scheduling (DRR and Fair Queueing) can provide explicit fairness.

In view of these developments, we seek to answer the following fundamental question in this paper: Does AIMD remain the sole choice for congestion avoidance and control even in these modern settings? If not, can other mechanism(s) provide better performance? We evaluate the four linear congestion control styles–AIMD, AIAD, MIMD, MIAD–in the context of these various loss recovery and router algorithms. We show that while AIMD is an unambiguous choice for the traditional setting of Reno-style loss recovery and FIFO drop-tail routers, it fails to provide the best goodput performance in the more modern settings. Where AIMD fails, AIAD proves to be a reasonable alternative.

Web Servers Under Overload: How Scheduling Can Help

Schroeder & Harchol-Balter

Carnegie Mellon University Technical Report CMU-CS-02-143, May 2001.

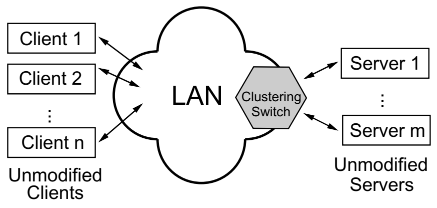
Most well-managed web servers perform well most of the time. Occasionally, however, every popular web server experiences transient overload. An overloaded web server typically displays signs of its affliction within a few seconds. Work enters the web server at a greater rate than the web server can complete it, causing the number of connections at the server to build up. This implies large delays for clients accessing the server. This paper provides a systematic performance study of exactly what happens when a web server is run under transient overload, both from the perspective of the server and from the perspective of the client. Second, this paper proposes and evaluates a particular kernel-level solution for improving the performance of web servers under overload. The solution is based on SRPT connection scheduling. We show that SRPT-based scheduling improves overload performance across a variety of client and server-oriented metrics.

Cuckoo: Layered clustering for NFS

Klosterman & Ganger

Carnegie Mellon University Technical Report CMU-CS-02-183, September 2002.

Layered clustering allows unmodified distributed file systems to enjoy many of the benefits of cluster-based file services. By interposing between clients and servers, layered clustering requires no changes to clients, servers, or the client-server protocol. *Cuckoo* demonstrates one particular use of layered clustering: spreading load among a set of otherwise independent NFS servers. Specifically, Cuckoo replicates frequently-read, rarely-updated files from each server onto others. When one server has a queue of requests, read requests to its replicated files are offloaded to other servers. No client-server protocol changes are involved. Sitting between clients and servers, the



Layered clustering architecture. Clients, servers, and the client-server protocol are unmodified. The "clustering switch" is the only change, and it's role is to add the clustering functionality by transparently translating some client requests into redirected server requests. The same role can be played by a collection of small intermediaries at the front-ends of the servers.

interposer simply modifies selected fields of NFS requests and responses. Cuckoo provides this load shedding with only 2000 semicolons of C code. Further, analyses of NFS traces indicate that replicating only 1000-10,000 objects allows 42-77% of all operations to be offloaded.

On Correlated Failures in Survivable Storage Systems

Bakkaloglu, Wylie, Wang & Ganger

Carnegie Mellon University Technical Report CMU-CS-02-129, May 2002.

The design of survivable storage systems involves inherent trade-offs among properties such as performance, security, and availability. A toolbox of simple and accurate models of these properties allows a designer to make informed decisions. This report focuses on availability modeling. We describe two ways of extending the classic model of availability with a single "correlation parameter" to accommodate correlated failures. We evaluate the efficacy of the models by comparing their results with real measurements. We also show the use of the models as design decision tools: we analyze the effects of availability and correlation on the ordering of data distribution schemes and we investigate the placement of related files.

… continued on pg. 20

RECENT PUBLICATIONS

… continued from pg. 19

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

Pennington, Strunk, Griffin, Soules, Goodson & Ganger

Carnegie Mellon University Technical Report CMU-CS-02-179, September 2002.

Storage-based intrusion detection allows storage systems to transparently watch for suspicious activity. Storage systems are well-positioned 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. It describes and evaluates a storage IDS, embedded in an NFS server, demonstrating both feasibility and efficiency of storage-based intrusion detection. In particular, both the performance overhead and memory required 40 KB for a reasonable set of rules) are minimal. With small extensions, storage IDSs can also be embedded in block-based storage devices.

DEVELOPING STORAGE SYSTEMS EDUCATION

Greg Ganger, David Nagle & Joan Digney

At Carnegie Mellon we are tackling the important problem of creating education in storage systems, a subject that is at least as broad and deep as other computer systems topics on which universities teach class sequences (e.g., processor architecture, operating systems, networking, databases, and compilers). Ever underappreciated, storage systems are at the core of the Information Age, offering unmatched opportunities for current and future computing professionals. Seemingly boundless growth comes with the transition from paper to digital storage and digital video, and storage systems offer fascinating design and implementation challenges. Their components' inner workings require amazing feats of engineering. Building efficient, scalable, reliable, secure, cost-effective, manageable storage systems from these components requires a storage-oriented combination of architecture, operating systems, networking, and distributed computing knowledge. Further, storage systems usually dominate the performance of a system, making them one of the few remaining places for performance engineers to thrive. Within the field of computer systems and computer engineering, there is no area whose demand for bright people and better solutions is more robust.

Sadly, storage systems are among the least understood areas of computer systems. The field is rife with buzzwords, like RAID and NAS and SAN, and bold claims of novelty, scalability, and manageability. But, many seem not to understand the details of storage systems, their consequences, or even the fact that the buzzwords rarely describe new technologies (just new names for old ideas). Historically, universities have provided little education in this space and there have been few useful books.

For two years, we have taught storage systems as a full-semester, 4th-year course focused on storage's incorporation and role in computer systems. 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 approaches involved in tying them together and making them useful. Along the way, we examine several real case studies, the demands placed on storage systems by important applications, and the impact of trends and emerging technologies on future storage systems. In the Spring 2002 offering, designated "18-546: Storage Systems," base lecture material was complemented by real-world expertise generously shared by 8 guest speakers from industry (including 2 CTOs and 4 of the 8 members of the SNIA Technical Council). The students who have taken the class will now be better prepared to contribute to the storage industry today and in the future. More information can be found at http://www.ece.cmu.edu/~ganger/ece546.spring02/.

This course and an associated book on storage systems are an attempt to fill the gaping hole in computer systems education. The book, which is evolving as the class is taught, will make it much easier for other universities to start offering storage systems education. Hopefully, it will also be useful to graduates who did not have access to a storage systems class.

Stay tuned!