THE
PDL Packet
THE NEWSLETTER ON PARALLEL DATA SYSTEMS • FALL 1998
http://www.pdl.cmu.edu

AN INFORMAL PUBLICATION FROM A UNIVERSITY RESEARCH COMMUNITY DEVOTED TO ADVANCING THE STATE OF THE ART IN STORAGE SYSTEMS AND TO EFFICIENTLY INTEGRATE STORAGE INTO PARALLEL AND DISTRIBUTED FILE SYSTEMS, HIGH BANDWIDTH NETWORKS AND COMPUTER CLUSTERS.

I • N • S • I • D • E

CONSORTIUM MEMBERS

CLARiiON Array Development
Compaq Corporation
Hewlett Packard Labs
Hitachi, Ltd.
IBM
Intel Corporation
LSI Logic (formerly Symbios)
Quantum Corporation
Seagate Technology
Siemens Microelectronics, Inc.
Storage Technology Corporation
Wind River Systems
3Com Corporation

Active Disks: A Scenario For Cost-effective Massive Data Processing

Erik Riedel & Garth Gibson

The continuously expanding computational power of today's technology industry (Figure 1) is influencing computer research in two different ways. At one extreme, typical of "big" science research, dramatic increases in computational power are employed in performance-record-breaking supercomputers to tame problems never before considered tractable. At the other extreme, typical of individual, minimal-budget research, increases in computational power are employed in desktop computers allowing knowledge discovery to be used where it has not been previously economically viable. It is this latter case, the ingress of knowledge/discovery computing into human society's everyday decision making, that drives our interest in personal, massive data mining and multimedia processing with next generation disk drives.

Consider a multi-outlet retail store chain interested in improving its effectiveness by better understanding its business. Already computerized for inventory and accounting purposes, the chain compiles on-line all sales records, inventory deployments, customer credit and check use records, staffing records and even video records of interactions between staff and customers. In a matter of months, terabytes of data accumulate. While the head office employs a small team of analysts, they want to allow equal access to the management of individual stores over the company's intranet. Data is centralized behind a set of servers in head office and all users interact

... continued on pg. 15

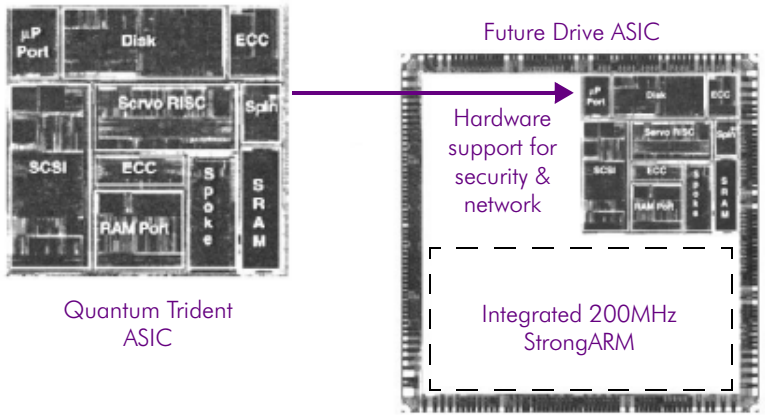


Figure 1: Drive electronics are trending toward higher levels of integration. For example, the Quantum Trident disk drive has several specialized hardware state machines combined into a single ASIC (left). The next generation of silicon technology would make it possible to integrate a much better control processor while still reducing cost and total chip count.

NASD Goes Active:

Programming the Infrastructure to Data-crunch

From the Director's Chair

GARTH GIBSON

Hello from the 'Burgh -- home of Steelers, Pirates, Penguins, AFS, Mach, RAIDframe, TIP, NASD and Active Disks.

In the two years since the last issue of this newsletter, the Parallel Data Laboratory has been very busy. Four PhD dissertations have been finished and defended. Twelve papers have been published or accepted for publications [MSST96, ICPDS96, OSDI96, Computing Surveys96, 2 in SIGMETRICS97, HICSS97, PER97, SIGMOD-DMKD98, VLDB98, ASPLOS98, OSDI99], all (but OSDI99) available on our web pages (www.pdl.cs.cmu.edu). Five faculty, ten students, and four staff members have joined our ranks. Added to our resources have been about 50 300+ MHz PCs and workstations, two gigabit networks, four 100Mbps ether LANs, a 50 port OC3 ATM switch and hundreds of disk drives. Accordingly, our physical space consumption has grown. We have increased our lab space from a 1000 square feet two years ago to over 2500 square feet today (spread over three rooms on the 3rd floor in Wean Hall and two rooms on D-level in Hammerschlag Hall). Moreover, the list of sponsoring companies in the Parallel Data Consortium has increased from 8 to 13.

Recognizing my limitations, it seemed intelligent to extend the PDL with an executive director. David Gordon, Array Technologies founder, helped us out in this role for six months before moving on. And just three months ago we were honored by the return to our midst of Bill Courtright, PDL graduate and RAIDframe author, as executive director of the Parallel Data Lab. I am very pleased to have Bill's energy, organization, leadership, industrial experience, and down-to-earth sensibility at the center of the Lab's operations. With Bill and the administrative excellence provided by Patty Mackiewicz, I am confident that the now thirty strong Parallel Data Lab is pulling together more effectively than ever.

Financially, our sponsoring companies have more than doubled their support for the Lab; DARPA's Information Technologies Office (ITO) has extended the NASD contract to cover FY99 and awarded us (under David Nagle's leadership) a new contract to develop Active Networking for Storage systems and applications. NSF, through its DSSC engineering research center, anticipates continued support through the end of the center in 2001 and we are working on proposals to fund a broad agenda of computation in the drive under the banner of Active Disks.

Enough on the Lab's byline; let me turn to the research. At this time two years ago, our prefetching research, whose goal was to exploit parallelism in storage devices for applications that are coded to access data serially, was already mature. Based on getting applications to "disclose" their future accesses explicitly, we had developed and implemented TIP, a cost-benefit economic model for allocating cache buffers and triggering prefetching automatically [see SOSP95 on the web pages]. This system realized a reduction in disk stall time of six diverse I/O-intensive applications by 75% - 95% when data is striped over 10 disks. Since then we have collaborated with researchers at Princeton and the University of Washington to increase TIP's aggressiveness in situations where data mappings onto storage devices are unbalanced [OSDI96, SIGMETRICS97-Tomkins]. We have also demonstrated that TIP technology transparently extends from locally attached disks to networked storage

… continued on pg. 5

THE PDL PACKET

The Parallel Data Laboratory
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3891
VOICE 412•268•6716
FAX 412•268•3010
FAX ALTERNATE 412•268•5576

PUBLISHER
Garth Gibson

EDITOR
Joan Digney

The *PDL Packet* is published approximately once per year and mailed to members of the Parallel Data Consortium. Copies are given to other researchers in industry and academia as well. Postscript and pdf versions will reside in the Publications section of the PDL Web pages. 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.

MISSION STATEMENT

To advance the state of the art in storage systems and integration into parallel and distributed file systems, high bandwidth networks, and computer clusters.

CONTACTING US

WEB PAGES

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

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

FACULTY

Garth Gibson (director)
412•268•5890
garth.gibson@cs.cmu.edu

Christos Faloutsos
christos.faloutsos@cs.cmu.edu

Greg Ganger
greg.ganger@ece.cmu.edu

Seth Goldstein
seth.goldstein@cs.cmu.edu

Todd Mowry
todd.mowry@cs.cmu.edu

David Nagle
david.nagle@ece.cmu.edu

Hui Zhang
hui.zhang@cs.cmu.edu

POST DOCTORAL FELLOW

Tara Madhyastha
tara.madhyastha@cs.cmu.edu

STAFF MEMBERS

Bill Courtright (executive director)
412•268•6733
bill.courtright@cs.cmu.edu

Patty Mackiewicz (business administrator)
412•268•6716
patty.mackiewicz@cs.cmu.edu

Mike Bigrigg
Joan Digney
Jennifer Landefeld
Sean Levy
Paul Mazaitis
Marc Unangst
Jim Zelenka

GRADUATE STUDENTS

Khalil Amiri
Stephen Bijanski
Angela Demke Brown
Jeff Butler
Fay Chang
Howard Gobioff
John Griffin
Garth Goodson
Charles Hardin
Stavros Harizopoulos
Jon Kliegman
David Petrou
Erik Riedel
David Rochberg
Chris Sabol
Jiri Schindler
Steve Schlosser
Tom Wagner
Ted Wong
Shuheng Zhou

New Executive Director for the Parallel Data Lab

The Executive Summary
BILL COURTRIGHT

Hello! For those of you who don't know me, my name is Bill and I arrived in July to become the Executive Director of the PDL. I am known by some of you because I recently completed a PhD in ECE here, under the tutelage of Garth Gibson. To others, I'm known because I was an employee of Symbios (formerly NCR, now LSI Logic) for 12 years.

I'm sure that many of you are wondering, "What does an 'Executive Director' really do?" As Executive Director, I have two fundamental responsibilities: operations and industry relations. On the operational side, I oversee the PDL staff as well as manage the PDL budget. In short, I keep the organization running smoothly so that our visionaries can concentrate on making the world a better place. On the industry side, I oversee the Parallel Data Consortium (PDC), the collection of companies (currently thirteen in number) who provide direct support of the PDL. I am your representative on campus. In addition to fundraising, I am deeply concerned about technology transfer and maximizing the value of your relationship with the PDL. To this end, I would like to visit your company to review our work and listen to your concerns. I have already begun this activity, and if I haven't already visited your location, I look forward to doing so in the coming months. I believe that there is substantial opportunity to increase the presence of each of our sponsors within the lab, as well as on the campus at large, and I want to enable this. For instance, there are opportunities to give talks at a variety of campus venues, making yourself known to the campus community.

A bit more about my industry background: while at Symbios, I worked in their RAID group as a hardware design engineer for six and a half years. I took an academic leave to work on a PhD at CMU, did a brief rotation in Strategic Marketing, and then concluded my tenure as an architect in their Server & Storage Architecture staff. Most recently, I represented Symbios at both SNIA and with the PDC. While a grad student at CMU, my research focussed upon the use of transactional guarantees to simplify the implementation of RAID systems. This work was embodied in RAIDframe, a collaborative project within the PDL (www.pdl.cs.cmu.edu/RAIDframe) which produced a rapid prototyping framework for implementing and evaluating novel RAID architectures.

Last but not least, "Why did I return to CMU?" This is the easiest question of all to answer: even a cursory look at the people and the research results of the PDL should convince you, as it did me, that the PDL is something special; I truly believe that any opportunity to participate, either as a staff member, or a member of the PDC, should be seized with both hands. I believe that my experience in industry and academia will enable me to understand your needs and represent you on campus. I am looking forward to working with all of you. If you haven't yet heard from me directly, you soon will. Also, please feel free to contact me at any time, either by email (wvcii@cs.cmu.edu) or by phone: (412) 268-6733. See you soon!

YEAR IN REVIEW

A synopsis of recent events in the PDL.

November 1998

- PDS Career Day, Retreat & Workshop

September 1998

- ASPLOS98 - Garth speaks on NASD
- Bill presents NASD to SNIA in Albuquerque
- NSIC/NASD workshop, Vancouver, B.C., Canada

August 1998

- SIGCOMM'98 - Dave attends
- VLDB98 - New York - Erik presented on Active Disks
- Usenix NT Symposium - Erik presented on Sequential I/O

July 1998

- DARPA PI meeting, San Diego CA
- Bill begins as Executive Director
- NSIC annual meeting

June 1998

- SIGMOD98 - Seattle: Erik presents Active Disks to Data Mining workshop; Garth receives *Test of Time* award for RAID paper in SIGMOD88.
- Garth Keynote speaker at Seagate
- NSIC/NASD workshop - Berkeley, CA
- SIGMETRICS98 - Madison, WI - Garth co-Chairs

April 1998

- DARPA site visit to CMU

March 1998

- DSSC - NSF Review at CMU
- NSIC/NASD workshop in conjunction with SNIA, Colorado Springs, CO
- SDI Seminar - Raj Jain, Ohio State

February 1998

- Qingming Ma defends Ph.D. thesis: Quality of Service
- SDI Seminar - Peter Chen, U. of Michigan

December 1997

- NSIC-NASD workshop, San Diego, CA
- Hugo Patterson defends PhD thesis: TIP

November 1997

- SC'97, San Jose CA - Tara presents on I/O Classification; Garth, Jim present on Low Level API
- IOPADS, San Jose CA
- SDI Seminar - David Wetherall, MIT

October 1997

- NSIC/NASD, Louisville, CO
- Parallel Data Systems Career Day & Retreat, Nemacolin, PA
- SDI Seminar - C. Faloutsos, U. of Maryland; Joel Emer, Digital Equipment

September 1997

- Andrew Tomkins defended Ph.D. thesis: TIPTOE
- NSIC-NASD workshop, Boulder, CO
- Hot Interconnects, San Francisco, D. Nagle on Networking for NASD
- SIO sponsors mtg., Houston, TX
- DSSC Fall Review
- SDI Seminar - Steve Lucco, Microsoft; Ken Shirriff, Sun

July 1997

- DARPA PI Meeting, D.C
- NSIC-NASD workshop, Shrewsbury, MA

June 1997

- SIO all hands meeting, Caltech, CA
- NSIC-NASD workshop, Monterey, CA
- SIGMETRICS97 - Seattle, WA: Garth presents NASD, Andrew Tomkins presents TIPTOE.

April 1997

- Bill Courtright defended Ph.D. dissertation: Transactional RAID
- SDI Seminar - Jim Gray, Microsoft

March 1997

- DSSC-NSF Review at CMU
- NSIC-NASD workshop, Almaden, CA
- SDI Seminar - Bob Broderson, Berkeley

February 1997

- Pittsburgh Supercomputing Center-Garth on High Performance SIO
- Tara's postdoctoral presentation

January 1997

- Task Force on Network Storage Arch., HICSS-30: Maui, Garth, chair, Dave Nagle, attending
- TIP Research Exchange meeting, Toronto
- SDI Seminar - Mic Bowman, Transarc

December 1996

- DARPA PI meeting, Dallas TX
- SDI Seminar - John Wilkes, HP Labs

November 1996

- Garth presents NASD at U. of Arizona
- OSDI conference, Seattle WA; T. Kimbrell presents Forestall/TIP
- SDI Seminar - John Hartman, U. Arizona; David Tennenhouse, MIT

October 1996

- Parallel Data Systems Retreat, Deep Creek, MD
- DARPA PI meeting, Dallas TX
- SDI Seminar - Tara Madhyastha, U. of Illinois

September 1996

- NASA Goddard Mass Storage Conf., Wash DC: Garth, keynote speaker.
- SDI Seminar - Kai Li, Princeton

Garth Gets Wired

This spring, in the May 1998 issue of *Wired* magazine, our fearless leader, Garth Gibson (and by association, the PDL) were profiled by Steve G. Steinberg as innovators in *Wired's* "Crucial Tech" pages noting that "NASD could mean sustainable profits for hard drive manufacturers".

Network - attached secure disks (NASD) are the focus of the article and said to be the coming thing in data storage/drive management.

NASD eliminates file server bottlenecks from the data path, allowing clients to speak directly to disks. This is accomplished by endowing the disks with an object, rather than block interface, and adding functionality in the drive to enforce security. A file manager is still present in the system, but only to control ACLs and namespace translation.

Garth takes aim at server bottlenecks.

Clients first consult the file manager which grants an encrypted capability and an identifier for the NASD device which contains the requested file (object). It checks to make sure that the client has the right to the particular data requested and issues a cryptographic ticket granting access. Since the computational requirements of this process are not great, a file server can now oversee many drives at a much lower cost than previously. Now that data must no longer move through the file server on its way to the client, the server bus is not a bottleneck to the process any more and data transfer rates are substantially improved and from that point on, the client communicates directly with the drive.

NASD Goes Active

… continued from pg. 2

[PER97]. Causing applications to disclose access patterns is also part of this project. On one hand, we are exploring the effectiveness and portability of competing disclosure abstractions, but in general we are not satisfied with relying on manual programmer efforts so we are developing two approaches to automatic disclosure. One approach builds on compiler transformation technology (strip mining), applying this to I/O-bearing nested loops to predict which accesses would not fit in the file cache. The second approach uses link-time transformations to add a side-effect-free (except for disclosure) speculative execution thread enabling an application to "analyze" its own dynamic state while it is otherwise stalled [OSDI99].

With cost-effective switched local area networks, cluster-based scalable server technologies and multi-hundred megabit-per-second storage media bandwidths, we recognized in 1995 that it was no longer cost-effective to devote file server machines to be store-and-forward front-ends for storage devices. Instead, the Lab's Network-Attached Secure Disks (NASD) project focuses on directly attaching storage devices to the cluster network and executing low-level file system primitives in the device [ASPLOS98]. By careful selection of the primitives to be offloaded, vestigial file servers can be made to scale more effectively [SIGMETRICS97-Gibson], and overall system security can be verified at the device without trusting all of the operating systems attached to the cluster network [TR-97-185].

In the last two years we have implemented a prototype NASD drive in the kernel of a 133 MHz Digital UNIX Alpha workstation, approximately the computing power we expect to be possible on a drive-specific multi-function ASIC today; ported two distributed file systems, NFS and AFS, to use NASD; developed and implemented transparent NASD striping in an interposition layer; and constructed a simple parallel file system based on the Scalable I/O Initiative's Low-Level Application Programming Interface [TR-96-193] and NASD. We have demonstrated a parallel data mining application (association rule discovery) on a 300 MB benchmark retail sales file in which eight 255 MHz Digital UNIX Alpha workstations process in parallel data striped over eight NASD prototypes, interconnected by an 155 MB/s ATM switch, at 45 MB/s - delivering over 95% of the bandwidth of the underlying hardware linearly at about six MB/s per drive-client pair. In comparison, a much more powerful 500 MHz NFS server with more than twice the total raw disk bandwidth and over 35 MB/s network bandwidth cannot provide more than 23 MB/s to eight clients even if each client has a separate copy of the data (avoiding contention).

Moreover, in the past two years the National Storage Industry Consortium's working group for Network Attached Storage Devices, with leadership and commitment from the Lab's NASD project, has hosted seven public workshops, drawing 50 speakers and 200-300 attendees representing the breadth of the storage industry. Under development by some members of this working group is an Object-Oriented Disk interface specification targeted for submission to the ANSI X3T10 (SCSI) standards committee. NASD also attained some notoriety with a "cowboy" article in the May 1998 issue of WIRED magazine!

I believe the most important aspect of the NASD design is the interface architecture which decouples policy from primitives and introduces an object container abstraction for storage. Policy is provided by file system personality modules in the form of (cachable rights describing) cryptographic capabilities. Performance-critical primitives, such as read and write, are directly executed at the NASD drive. An object abstraction for storage devices enables aggressive performance and self-management optimizations by granting a higher level of knowledge about the storage data.

Beyond NASD, we have observed that devices, through increasing CMOS chip density, will soon be able to include much more computing power with relatively small increases in cost. While NASD uses some this power to embed a simple, fixed-function object storage system, the Active Disk project proposes that storage devices include remote execution facilities so that it can apply the parallelism of large storage systems to off-load to storage and accelerate portions of parallelizable I/O-intensive applications. Specifically, we are examining high-dimensional searching codes in which indices are ineffective [VLDB98]. We have used our prototype NASD hardware environment to demonstrate the potential for Active Disk acceleration of the parallel data mining application described above, embedding a portion of the application into each NASD drive. For this application, an Active Disks implementation matches the parallel file system performance with negligible computing time on a single host computer and negligible network bandwidth. We have just begun this research, but are already confident that Active Disks reinforces the requirement for a NASD infrastructure because computing inside the drive makes little sense unless that drive can enforce its access controls and manage its resources -- the object interface facilitates in-drive programming.

In the long term, I believe that the traditional Von Neumann model of a central processor with peripheral devices may well give way to a utilitarian model in which mechanically justified devices such as cameras, monitors, disks, or network switches will be endowed with sufficient processing power to all but eliminate central processors from common computing. Systems of the future may well be cobbled together from required devices and cooperating software running in these devices. CMU's Parallel Data Lab plans to be there before the future happens.

The computer world's problems to solve and Jon's still smiling.

Hint Generation via Speculative Execution

F. Chang & J. Digney

Hints are a well-established, broadly applicable technique for improving system performance. Research in this area has used hints or historical predictions to both hide access latency and intelligently manage resources. While automatic prefetching techniques based on historical access patterns are effective for some workloads, they fall short of the effectiveness of manually-driven (programmer-inserted) prefetching for applications with irregular and input dependent access patterns.

The goal of speculative execution is to automate hint generation without requiring a programmer to modify source code by hand. Cycles during which an application is normally stalled waiting for I/O request completion are used for speculative execution to discover and hint future reads. Even high performance disk systems currently have at least 10 millisecond access latencies which means that processors may be wasting *millions* of cycles during each I/O stall.

During speculative execution, our system issues the appropriate (non-blocking) hint call whenever they encounters a read request in order to inform the underlying prefetching system that the data indicated by the request may soon be required. If the hinted data is not already cached and the prefetching system believes that prefetching the hinted data is the best use of disk and cache resources, then it should issue an I/O request for the hinted data. If the I/O system can parallelize fetching hinted data with its servicing of the outstanding read request, then the latency of fetching data that will be required later may be partially or completely hidden from the application.

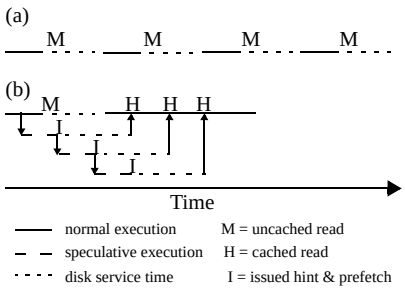


Figure 1(a) shows how normal execution proceeds for a hypothetical application; (b) shows execution proceeding for the application if speculative execution were used to generate I/O hints. Speculative execution could more than halve execution time for the application.

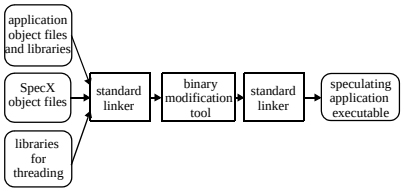


Figure 2: Transforming applications to use speculative execution. The SpecX object files contain various routines executed by the original or speculating thread in order to support speculative execution.

Figure 1 depicts the intuition behind why speculative execution works. Consider an application which issues 4 read requests for uncached data and processes for 1 million cycles before each of these read requests. Assume that we have an I/O system with a disk access latency of 3 million cycles and that there are sufficient cache resources to store all of the data used by this application. If we assume that speculative execution proceeds at the same pace as normal execution, then, while normal execution is stalled waiting for the first read request to complete, speculative execution may be able to issue hints for the remaining 3 read requests. If the I/O system can parallelize fetching all of the hinted data with servicing the outstanding read request and the subsequent processing, then all of the subsequent read requests will hit in the cache, such that the application's execution time will be more than halved.

Speculative execution requires a tool which is able to automatically add code to arbitrary programs. Recently, we completed a preliminary implementation of a binary rewriting tool, illustrated in Figure 2, which adds speculative execution to binaries generated by C compilers on Alpha workstations running Digital Unix 3.2.

During speculative execution, we prevent the speculating thread from changing the data values used during normal execution by performing software-enforced copy-on-write. This involves adding checks before each load and store instruction executed by the speculating thread, and adding a data structure to keep track of which memory regions have been copied and where the copies reside. Before each store instruction executed by the speculating thread, a check is added which accesses the data structure to discover whether the targeted memory region has already been copied. If so, the store is redirected to access the copy. If not, the memory region is copied, the data structure is updated, and the store is redirected to the newly created copy. Before each load instruction, a check is added which accesses the data structure to discover whether the referenced memory region has already been copied and, if so, redirects the load to obtain the value stored in the copy, which is the current value with respect to speculative execution.

Benchmark	TIP Execution time(s)		stddev	NO TIP
Agrep	no hints specX Δ *manual*	21.07 10.50 **-50.16%** *-72%*	0.30 0.21	**+2.6%**
Gnuld	no hints specX Δ *manual*	95.85 72.20 **-24.67%** *-66%*	0.48 2.18	**+4.6%**
XDS	no hints specX Δ *manual*	327.65 103.68 **-68.35%** *-70%*	2.25 0.47	**+1.0%**

Table 1: Implementation performance on 233MHz AlphaStation with data striped over 4 disks. TIP column shows results using a kernel performing hint driven prefetches; No TIP shows slowdown when the kernel ignores these hints. No hints is the original non-hinting application. SpecX is the application with speculative execution. Delta is the difference between non-hinting and speculating applications. Manual is the difference between the original and manually modified applications.

We evaluated our design with three real world disk-bound applications from the TIP benchmark suite: Agrep, Gnuld, and XDS. While our techniques are currently unsophisticated, they perform surprisingly well - all three achieve at least a 25% reduction in execution time, with one achieving a 68% reduction when the data files are striped over 4 disks.

File Server Scaling with Network Attached Secure Disks

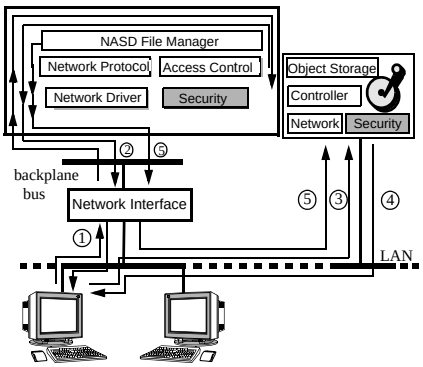
Gibson, Nagle, Amiri, Chang, Feinberg, Gobioff, Lee, Ozceri, Riedel, Rochberg & Zelenka

Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics '97), Seattle, Washington, June 15-18, 1997.

ABSTRACT

By providing direct data transfer between storage and client, network-attached storage devices have the potential to improve scalability for existing distributed file systems (by removing the server as a bottleneck) and bandwidth for parallel and distributed file systems (through network striping and more efficient data paths). Together, these advantages influence a large enough fraction of the storage market to make commodity network-attached storage feasible. Realizing the technology's full potential requires careful consideration across a wide range of file system, networking and security issues. This paper contrasts two network-attached storage architectures: (1) Networked SCSI disks (NetSCSI); network-attached storage devices with minimal changes from the familiar SCSI interface and (2) Network-Attached Secure Disks (NASD); drives that support independent client access to drive object services.

To estimate the potential performance benefits of these architectures, we develop an analytic model and perform trace-driven replay experiments based on AFS and NFS traces.Our results suggest that NetSCSI can reduce file server load during a burst of NFS or AFS activity by about 30%. With the NASD architecture, server load (during burst activity) can be reduced by a factor of up to five for AFS and up to ten for NFS.



NASD offloads more of the file system's operations. A client requests access to a file from the file manager (1), it delivers a capability to the client (2). The client may then make repeated accesses to different regions of the file (3, 4) without contacting the file manager again unless the file manager chooses to force reauthorization by revoking the capability (5).

File Systems for Network Attached Secure Disks

Gibson, Nagle, Amiri, Chang, Gobioff, Riedel, Rochberg & Zelenka

School of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-97-118, 1997.

ABSTRACT

Network-attached storage enables network-striped data transfers directly between client and storage to provide clients with scalable bandwidth on large transfers. Network-attached storage also decouples policy and enforcement of access control, avoiding unnecessary reverification of protection checks, reducing file manager work and increasing scalability. It eliminates the expense of a server computer devoted to copying data between peripheral network and client network. This architecture better matches storage technology's sustained data rates, now 80 Mb/s and growing at 40% per year. Finally, it enables self-managing storage to counter the increasing cost of data management. The availability of cost-effective network-attached storage depends on it becoming a storage commodity, which in turn depends on its utility to a broad segment of the storage market. Specifically, multiple distributed and parallel file systems must benefit from network-attached storage's requirement for secure, direct access between client and storage, for reusable, asynchronous access protection checks, and for increased license to efficiently manage underlying storage media. In this paper, we describe a prototype network-attached secure disk interface and filesystems adapted to network-attached storage implementing Sun's NFS, Transarc's AFS, a network-striped NFS variant, and an informed prefetching NFS variant. Our experimental implementations demonstrate bandwidth and workload scaling and aggressive optimization of application access patterns. Our experience with applications and filesystems adapted to run on network-attached secure disks emphasizes the much greater cost of client network messaging relative to peripheral bus messaging, which offsets some of the expected scaling results.

Prefetching over a Network: Early Experience with CTIP

(Please also see article on page 13)

Rochberg & Gibson

SIGMETRICS Performance Evaluation Review, V25, N3, December 1997. pp 29-36. Also appears as CMU-CS-TR-87-184.

ABSTRACT

We discuss CTIP, an implementation of a network file system extension of the successful TIP informed prefetching and cache management system. Using a modified version of TIP in NFS client machines (and unmodified NFS servers), CTIP takes advantage of application-supplied hints that disclose the application's future read accesses. CTIP uses these hints to aggressively prefetch file data from an NFS file server and to make better local cache replacement decisions.

… continued on pg. 14

Christos Faloutsos

Christos received his B.Sc. degree in Electrical Engineering (1981) from the National Technical University of Athens, Greece and his M.Sc. and Ph.D. degrees in Computer Science from the University of Toronto, Canada.

He has been at Carnegie Mellon University since 1997, first as a visitor and since the spring of 1998 as a regular faculty member. He has received the Presidential Young Investigator Award given by the National Science Foundation (1989), two "best paper" awards (SIGMOD 94, VLDB 97), and three teaching awards. He has published over 70 refereed articles, one monograph, and has filed for three patents.

His research interests within the PDL focus on data mining; other research directions include physical data base design, searching methods for text and geographic information systems indexing methods for multimedia databases.

Greg Ganger

Dr. Ganger has broad research interests in the design, implementation and evaluation of computer systems, from hardware architecture through system software to applications. His current research focuses mainly on three main areas: robust high-performance servers, extensible operating systems (OSs), and file systems. Dr. Ganger plans to expand his research to include embedded operating systems (e.g. for storage devices), system architectures for I/O-intensive environments (e.g. distributed storage) and mechanisms for reducing and/or mitigating the complexity of computer systems.

Dr. Ganger earned his graduate and undergraduate degrees at the University of Michigan, Ann Arbor, completing a Ph.D. on the architecture and evaluation of storage subsystems. In August of 1995 he joined Frans Kaashoek's Parallel and Distributed Operating Systems group at the M.I.T. Laboratory for Computer Science. In December of 1997, he joined the faculty of the Department of Electrical and Computer Engineering at C.M.U., with a courtesy appointment in the School of Computer Science. Dr. Ganger is a member of several professional societies, a referee for several computer systems conferences and journals, and an examiner for the RAID Advisory Board's fault-tolerance classification process. Over recent years, he has also enjoyed research associations with several companies.

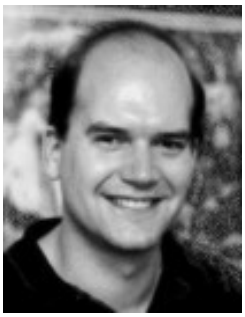
Seth Copen Goldstein

Seth Copen Goldstein joined School of Computer Science as an assistant professor in the fall of 1997. His areas of interest include reconfigurable computing and fine-grained parallelism. Seth comes to SCS from the University of California at Berkeley, where he worked on issues relating to high performance parallel systems. His thesis is entitled *Lazy Threads: Compiler and Runtime Structures for Fine-Grained Parallel Programming.*

Dr. Goldstein's research area is computer architecture and compilers. Currently his interest is focused on novel ways to use the hundreds of millions to billions of transistors that will be available to processor designers in the near future. In particular, he is interested in how user programs written high-level languages can be compiled directly into hardware, yielding order of magnitude speedups.

Seth's research fits within the purview of the PDL in the following ways. The compiler technology he is developing will be applicable to the new active disks technology, which will be made even more attractive to industry by the advances in reconfigurable computing.

Todd Mowry



Todd C. Mowry received his B.S.E.E. from the University of Virginia in 1988, and his M.S.E.E. and Ph.D. from Stanford University in 1989 and 1994, respectively. His Ph.D. dissertation was on *Tolerating Latency Through Software-Controlled Data Prefetching*, which he worked on under the supervision of Anoop Gupta and Monica Lam while participating in the Stanford DASH Multiprocessor (the predecessor to FLASH) and the SUIF Compiler projects. While at Stanford, he also worked part-time in the architecture group at MIPS (now a division of SGI) on projects including the MIPS R10000 processor. From 1994 through 1997, he was an assistant professor in the ECE and CS Departments at the University of Toronto. Starting in July, 1997, Prof. Mowry has been an associate professor in the CS Department at CMU, where he currently leads the STAMPede project. Todd's work within the PDL is exploring new ways that the compiler (with varying degrees of help from the hardware and the operating system) can use prefetching and other techniques to intelligently trade off consuming more bandwidth to reduce overall latency. Recent work in this area has included prefetching pointer-based codes, prefetching to hide disk latency in out-of-core numeric applications, and hiding network communication latency.

Informed Prefetching for Parallel Input/Output

Tara Madhyastha

Poor I/O performance is one of the primary obstacles to effective use of high performance multiprocessor systems. In an effort to provide high throughput commensurate with compute power, current multiprocessor systems provide multiple disks or disk arrays attached to I/O processors that communicate with compute processors through a fast network. Unfortunately, parallel file systems have not yet been able to consistently deliver peak hardware performance to applications with widely varying I/O requirements. The goal of this work is to study how a general informed prefetching mechanism can be used in conjunction with parallel I/O application programming interfaces to provide the high performance usually associated with specialized system-level optimizations.

One important optimization is for a collective input/output operation. When the threads of a parallel application simultaneously access portions of a shared file using a UNIX-style interface, a separate system call is required for each disjoint portion. This can result in non-sequential accesses, and consequently, poor performance at the I/O nodes. Providing high-level access pattern information to the file system through an application programming interface (API) allows it to reorder requests, servicing the request in the most efficient manner. This motivates optimizations such as two-phase and disk-directed I/O; given global, high-level knowledge that some data must be read or written before all the processors can proceed, the I/O operations can be reordered to occur as an efficient collective. In recognition of its importance to parallel file system performance, an interface for collective I/O is specified as an extension to the Scalable I/O initiative low-level API.

Informed prefetching is a more general technique for improving performance by providing access pattern information to a file system. The application constructs hints describing future accesses that a prefetcher uses to issue I/O requests in advance of the demand fetch stream. Deep prefetching enables better disk scheduling, improving throughput analogous to a collective I/O implementation that sorts disk requests. Unlike an efficient collective I/O implementation, the quality of disk scheduling is a function of the prefetch depth, or alternatively, the amount of memory available for prefetching.

We are investigating how to utilize memory for prefetch buffers to achieve throughput similar to a collective I/O implementation and have derived models for expected throughput using prefetching, validating these experimentally. We have proved that with only a small amount of additional memory that scales with the number of disks, informed prefetching can outperform collective I/O.

Informed Prefetching and Caching: From TIP to TIPTOE

http://www.pdl.cmu.edu/TIP/

Andrew Tomkins

This article reports on the PDL publication "Informed Multi-Process Prefetching and Caching", which appeared at the 1997 ACM Sigmetrics, in Seattle, Washington. The results in the publication build on existing work at the PDL, so I begin with a quick summary. In December of 1995, the PDL published "Informed Prefetching and Caching" at the Fifteenth ACM Symposium on Operating Systems Principles, describing the TIP system for prefetching and cache management in the presence of application-provided information about future I/O accesses. Since then, a number of extensions to the work have been in progress. This article reports on a series of detailed comparisons between TIP and other systems, culminating in the development of TIPTOE, or TIP with Temporal Overload Estimators.

TIP and its extension TIPTOE make both fetch and eviction decisions using cost-benefit analysis. Figure 1 shows a schematic of the cost-benefit cache manager. In the figure, independent *estimators* express different strategies for reducing I/O service time. The arrows on the left represent estimators of the benefit attained by initiating a particular type of I/O. Demand misses need a buffer immediately to minimize the stall that has already started. Informed prefetching would like a buffer to initiate a read and avoid disk latency. Similarly, the arrows on the right represent estimators of the cost incurred by evicting data that has been cached for a particular purpose. LRU caching holds recently-read data that may be accessed again soon, reducing stall by increasing hit rate. Informed caching holds blocks that will be accessed again soon, according to application hints. The buffer allocator takes the least-valuable buffer held by any cache to initiate the most beneficial new I/O whenever the estimated benefit exceeds the estimated cost.

The first evaluation of different systems for informed prefetching and caching was performed by the PDL in collaboration with other groups at the University of Washington and the Uni-

… continued on pg. 17

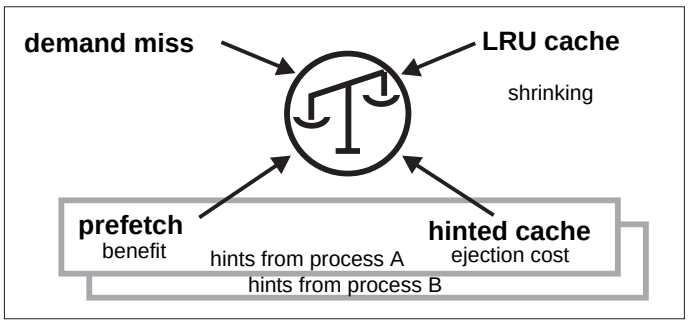


Figure 1: Informed Cache Manager Schematic

Since the last *PDL Packet*, three graduate students have proposed their Ph.D. thesis topics and four have successfully defended their Ph.D. dissertations. Abstracts follow.

THESIS PROPOSAL:
Security Issues for Network Attached Storage - April 16, 1997

Howard Gobioff, S.C.S

Distributed file systems are increasingly central to computational effectiveness and organizational data security. To improve performance by removing the file server from the data path between drive and client for common operations, researchers are investigating network attached storage systems. By moving the storage out from the protection of the file server, the storage is now exposed to a variety of hostile network attacks. I propose to address these network attacks through a software capability system along with a variety of hardware to support the security operations.

This proposal focuses on two major issues in the security of network attached storage devices: the costs and advantages of different hardware options and minimization of the number of times a client must request a capability. I propose to explore the design space for these two facets of the NASD security problem. Based on the exploration, I will implement a system that demonstrates the advantages of specific points in the design space. The expected contribution of this thesis is an understanding of the issues involved in the security of network attached storage including an understanding of the interaction the security of a software capability system and the performance of the overall storage system.

THESIS PROPOSAL:
A Highly Scalable Storage Service using Optimistic Synchronization and Smart NASDs - January 27, 1998

Khalil Amiri, E.C.E.

Storage systems that provide scalable throughput and use inexpensive commodity components are required to enable the growing pool of scientific and commercial applications to benefit from the availability of cycles in workstation clusters. In conventional systems, a server machine usually provides both directory service and storage management. It stacks several layers of abstractions to provide various levels of storage management functions. For example, a typical setup would stack a web server on top of a distributed file server on top of a local file system on top of a RAID controller. Each layer potentially induces a store-and-forward data copy and a synchronous serialization point. While this centralized layering simplifies implementation, it creates a performance and scalability bottleneck. Better scalability can be achieved if the *work* performed by each layer is decomposed to execute mostly at the client and the storage device. I propose to decompose this *work* into an untrusted client-resident part responsible for data access and an infrequently invoked trusted server responsible for policy, specifically authorization and management. The critical idea is to allow clients to operate with uncertain *cached* state when performing allocation, object migration and parity updates, and to support the necessary client synchronization optimistically through drive-embedded atomic primitives. Exploiting drive intelligence in the form of atomic primitives allows an asynchronous client organization that is highly scalable and available. The thesis is that using intelligent network- attached secure disks (NASD) serving data directly to clients and providing synchronization support, together with using optimistic concurrency control protocols at the clients provides the cornerstone for an ultra scalable and highly available storage system. The proposed research provides a proof of concept implementation, evaluates the proposed decomposition approach, and investigates the nature and cost of synchronization support that should be embedded in the NASD drive. I describe the research through Cheops-NASD, a prototype providing Client-exposed Highly Expandable OPtimistically Synchronized storage using NASDs.

THESIS PROPOSAL:
Active Disks - Remote Execution in Network-Attached Storage - May 29, 1998

Erik Riedel, E.C.E.

The increasing performance and decreasing cost of processors and memory are causing system intelligence to move into peripherals from the CPU. Storage system designers are using this trend toward "excess" compute power to perform more complex processing and optimizations inside storage devices. To date, such optimizations have been at relatively low levels of the storage protocol. At the same time, trends in storage density, mechanics, and electronics are eliminating the bottleneck in moving data off the storage media and putting pressure on interconnects and host processors to move data more efficiently. I propose a system called Active Disks that takes advantage of processing power on individual disk drives to run application-level code. Moving portions of an application's processing to execute directly at disk drives can dramatically reduce data traffic and take advantage of the storage parallelism already present in large systems. Execution of code directly at storage devices allows filter operations to be performed close to the data; enables support of application-aware scheduling of access and transfer; allows management functions to be easily customized; and makes possible more complex or specialized operations than a general-purpose storage interface would normally support. The focus of this work is to identify the characteristics of applications that make them suitable for execution at storage devices and quantify the benefits to individual application performance and overall system efficiency and scalability from the use of Active Disks.

DISSERTATION ABSTRACT:
A Transactional Approach to Redundant Disk Array Implementation - April 4, 1997
http://www.pdl.cmu.edu/PDL-FTP/RAID/CMU-CS-97-141.pdf

Bill Courtright

Redundant disk arrays are a popular method of improving the dependability and performance of disk storage and an ever-increasing number of array architectures are being proposed to balance cost, performance, and dependability. Despite their differences, there is a great deal of commonality between these architectures; unfortunately, it appears that current implementations are not able to effectively exploit this commonality due to their adhoc approach to error recovery. Such techniques rely upon a case-by-case analysis of errors, a manual process that is tedious and prone to mistakes. For each distinct error scenario, a unique procedure is implemented to remove the effects of the error and complete the affected operation. Unfortunately, this form of recovery is not easily extended because the analysis must be repeated as new array opera-

... continued on pg. 11

… continued from pg. 10

tions and architectures are introduced.

Transaction-processing systems utilize logging techniques to mechanize the process of recovering from errors. However, the expense of guaranteeing that all operations can be undone from any point in their execution is too expensive to satisfy the performance and resource requirements of redundant disk arrays.

This dissertation describes a novel programming abstraction and execution mechanism based upon transactions that simplifies implementation. Disk array algorithms are modeled as directed acyclic graphs: the nodes are actions such as "XOR" and the arcs represent data and control dependencies between them. Using this abstraction, we implemented eight array architectures in RAIDframe, a framework for prototyping disk arrays. Code reuse was consistently above 90%. The additional layers of abstraction did not affect the response time and throughput characteristics of RAIDframe; however, RAIDframe consumes 60% more CPU cycles than a hand-crafted non-redundant implementation.

RAIDframe employs roll-away error recovery, a novel scheme for mechanizing the execution of disk array algorithms without requiring that all actions be undoable. A barrier is inserted into each algorithm: failures prior to the barrier result in rollback, relying upon undo information. Once the barrier is crossed, the algorithm rolls forward to completion, and undo records are unnecessary. Experiments revealed this approach to have identical performance to that of non-logging schemes.

DISSERTATION ABSTRACT:
Practical and Theoretical Issues in Prefetching and Caching - September 17, 1997
http://www.pdl.cmu.edu/PDL-FTP/TIP/at_thesis.pdf

Andrew Tomkins

This thesis has two parts, the first more practical, and the second more theoretical. The first part considers *informed prefetching and caching* in which an application provides information about its upcoming I/O accesses to the operating system, allowing the system to prefetch data and to make informed cache replacement decisions. I compare existing algorithms for this problem using trace-driven simulation, and use the results to develop a new algorithm that performs better than previous approaches, again under trace-driven simulation.

The second part considers *weighted caching*, a theoretical problem from the domain of on-line algorithms. I present an algorithm with competitive ratio $O(\log 2\ k)$ on $(k + 1)$-point spaces, the first poly-logarithmic ratio for this problem. I also give an almost-tight lower bound of $\Omega(\log k)$ for any weighted caching problem on at least $k + 1$ points. I then show a connection between this problem and a new on-line k-server model in which the servers may rearrange themselves without cost during "free-time" between requests, and describe a series of results in the free-time model.

DISSERTATION ABSTRACT:
Informed Prefetching and Caching - December 22, 1997
http://www.pdl.cmu.edu/PDL-FTP/TIP/rhp_diss.pdf

R. Hugo Patterson

Disk arrays provide the raw storage throughput needed to balance rapidly increasing processor performance. Unfortunately, many important, I/O-intensive applications have serial I/O workloads that do not benefit from array parallelism. The performance of a single disk remains a bottleneck on overall performance for these applications. In this dissertation, I present aggressive, proactive mechanisms that tailor file system resource management to the needs of I/O-intensive applications. In particular, I will show how to use application-disclosed access patterns (hints) to expose and exploit I/O parallelism, and to dynamically allocate file buffers among three competing demands: prefetching hinted blocks, caching hinted blocks for reuse, and caching recently used data for unhinted accesses. My approach estimates the impact of alternative buffer allocations on application execution time and applies run-time cost-benefit analysis to allocate buffers where they will have the greatest impact. I implemented TIP, the informed prefetching and caching manager in the Digital UNIX operating system and measured its performance on a 175 MHz Alpha equipped with up to 10 disks running a range of applications. Informed pre-fetching on a ten-disk array reduces the wall-clock execution time of computational physics, text search, scientific visualization, relational database queries, speech recognition, and object linking by 10-84% with an average of 63%. On a single disk, where storage parallelism is unavailable and avoiding disk accesses is most beneficial, informed caching reduces the execution time of these same applications by up to 36% with an average of 13% compared to informed prefetching alone. Moreover, applied to multiprogrammed, I/O-intensive workloads, TIP increases overall throughput.

DISSERTATION ABSTRACT:
Quality of Service - February 12, 1998

Qingming Ma

Future integrated services networks will support multiple classes of service to meet the diverse quality-of-service (QoS) requirements of applications. In order to meet end-to-end performance guarantees, the paths being used have to meet strict resource constraints. The goal of QoS routing is to select paths that satisfy these constraints while achieving high resource utilization efficiency in the network. QoS routing is challenging because of the complicated and diverse resource sharing that happens between flows inside each service class, and across service classes. Moreover, for service classes that have multiple QoS constraints, selecting feasible paths can be computationally complex.

This thesis develops an integrated QoS routing framework that has two components. We define and evaluate (1) routing algorithms for individual service classes and (2) an architecture that integrates these per-class routing algorithms. We show that QoS routing, i.e.routing algorithms that consider both the properties of each service class and inter-class effects, is both necessary and feasible. By coupling routing with resource management mechanisms operating on shorter time scales, such as congestion control and scheduling, it is possible to define QoS routing algorithms that have both low computational complexity and high performance, i.e. low blocking rates or high throughput. This is achieved by striking the right balance between minimizing the resource utilization of individual connections and distributing the load in the network.

FACULTY

Four new faculty members have joined the PDL this academic year: Greg Ganger in the Dept. of Electrical and Computer Engineering and Christos Faloutsos, Seth Copen Goldstein and Todd Mowry in the School of Computer Science. Please see page 7 for brief biographies.

Tara Madhyastha joined the PDL last September as a Postdoctoral Fellow. Tara recently completed her Ph.D. dissertation *Automatic Classification of Input/Output Access Patterns* at the University of Illinois at Champaign-Urbana under the direction of Dr. Daniel Reed. The main focus of her research at CMU will involve extensions to the SIO LLAPI initiative.

STAFF

Since our last newsletter, the PDL has hired five new staff members: Mike Bigrigg, Bill Courtright, Joan Digney, Jennifer Landefeld and Sean Levy.

Mike moved to Pittsburgh and the PDL last November from Concurrent Technologies Corporation in Johnstown, PA where he was Project Manager for an object oriented technology research group. The plan is to get him in the middle of our NASD object store and linkage (RPC, remote execution, etc.).

Bill Courtright joined the PDL as our Executive Director in July 1998. He also serves as Director of NASD and as Program Manager of the NSIC-NASD project. Most recently, he was with Symbios Logic in Wichita, Kansas. We are happy to have him back.

Joan started with the PDL in September 1997 as a Research Associate/Writer to assist with technical writing, the PDL web pages and graphic design. Before coming to Pittsburgh, she was a technical writer with the Vehicle Concepts Group at the Canadian Defense Research Establishment in Medicine Hat, Alberta.

Jennifer comes to us from the East End Food Co-op in Pittsburgh where she was the office manager. Her interests in computing and higher education led her to apply for a support staff position in our group. Some of her other interests include glass beadmaking and microbrewing. Jennifer did her undergraduate work at the College of Wooster in Ohio.

Sean is the PDL's latest addition and has been an employee of Carnegie Mellon since 1988. He began working as a programmer with us at the beginning of September as a change from his most recent position at CMU with CS facilities.

STUDENTS

Several graduate students have begun their Masters or Ph.D. work within the PDL in the last year.

David Petrou is working with Garth on NASD. He graduated with his Bachelors degree from UC Berkeley in 1997.

Ted Wong received his M.Eng. in Computer Science from Cornell University. He is doing his Ph.D. research in the NASD area under Garth's direction.

Stephen Bijansky is a Masters student with David Nagle in the Dept. of Electrical and Computer Engineering working on the NASD project.

Jeff Butler is a Masters student in the Dept. of ECE, working with David Nagle. He is researching options for cooperative caching, active network participation in caching, and caching in the NASD framework.

Garth Goodson attended CMU for his undergraduate degree in ECE. He is currently a first year master's student working for Greg Ganger. His research involves porting NASD to the Exokernel OS.

Charles Hardin travelled from New Mexico to work on his Masters degree in ECE with David Nagle.

Jon Kleigman, after finishing his undergraduate degree here at CMU, decided on an M.S. program and is working with David Nagle on the NASD project.

Angela Demke Brown, recently of the University of Toronto, in Ontario, Canada is pursuing her Ph.D. supervised by Todd Mowry. Her area of research is compiler directed I/O prefetching.

Other new grad students in the PDL this fall include Stavros Harizopoulos, Tom Wagner (both CS), John Griffin, Chris Sabol, Jiri Schindler, Steve Schlosser and Shuheng Zhou (all in ECE).

We also have several new undergraduate programmers working with us: Tim Canfield, Jason Crawford, Jason Grosman, Kevin Killourhy, Nat Lanza, Adam Phelps and Ben Schmidt. As well, Meg Gaj is helping Jennifer in the office.

DEPARTURES

The past year has also seen several people move on to bigger and/or better things. (Things must have been better for them to have left us.)

Anne Byrne moved to a position with the Department of Design at CMU as its Computer Technology Manager.

Lisa Kicielinski, our librarian, left at the end of August for the bright lights of New York where she will be working towards a master's degree in theatre. We'll be able to say we knew her when...

Cheryl Gach, our graphic artist, finished with us in August 1998 and is currently employed in ECE, working on graphic design and photographic projects there.

Qingming Ma, Hugo Patterson and Andrew Tomkins have successfully defended their Ph.D. dissertations and moved on to continue their research in industry.

Qingming is working with CISCO Systems in Massachusetts.

Hugo reports that there is life after grad school and is enjoying his work with Quantum.

Andrew is in California and loving it. He is working with IBM and focusing on the theory of product choice for web display. He is also pursuing more traditional work on the classification of documents, a focus of his group.

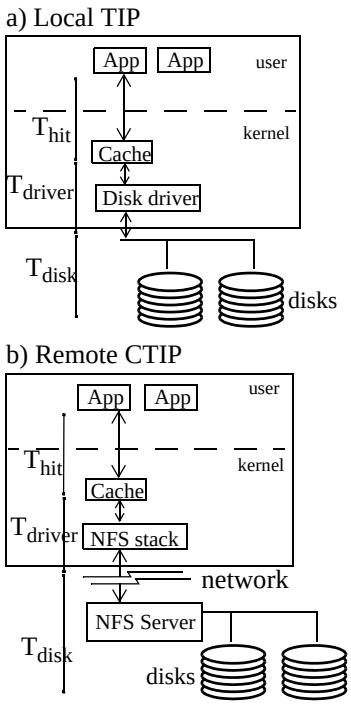
Tammo Spalink left CMU last summer to move to Tucson, Arizona where he has begun his Ph.D. research at the University of Arizona.

Extending TIP to the Network: CTIP

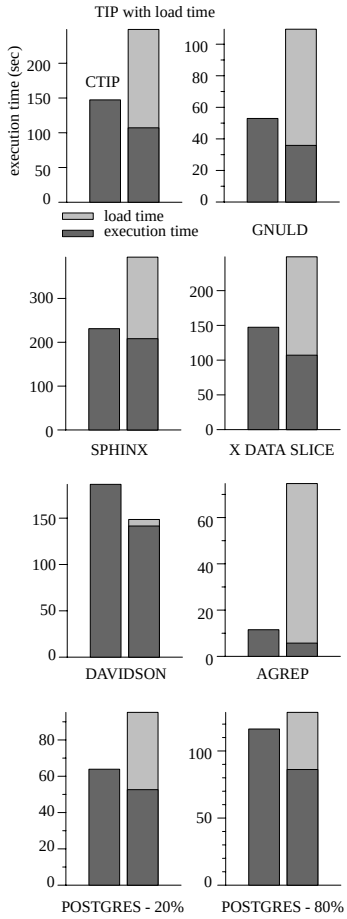
(Please also see the abstract on page 7)

D. Rochberg & J. Digney

The TIP (Transparent Informed Prefetching and Caching) system developed at the PDL demonstrates that informed file cache management can substantially reduce read stall time (SOSP95 on the publications web page). By *informed*, we mean that applications disclose their future read requests as *hints*. The TIP system uses this foreknowledge to manage the file cache. When a block must be ejected from the file cache (to make room for a new block to be read in), the TIP system uses hints to chose the block whose ejection will cause the smallest increase in application stall time.



The TIP and CTIP models: a) the local case and b) the remote case. All the time spent in filesystem code or waiting for the disk is stall time. We split that time into three components: T_{hit} is the CPU time to retrieve a block from the cache, T_{driver} measures the CPU time required to issue and receive a one-block read from storage, and T_{disk} measures the latency of retrieving a block from storage.



Hinted runs with execution times for the local case enlarged by the time required to load the application and its data from remote (NFS) storage into local storage.

Also, since the TIP system knows what blocks will be read in the near future, it can prefetch those blocks before the application requests them.

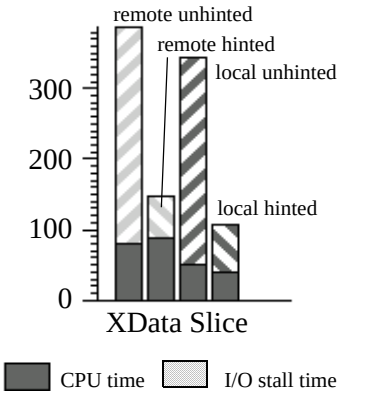
Until recently, the TIP system worked only with direct-attached disks. We have extended it to work with distributed file systems, removing the need for a high-bandwidth disk array at every client machine. It also paves the way for taking advantage of striping across multiple servers. If we view TIP as a latency-hiding technique, then distributed file systems are simply pseudo-disks with more latency for TIP to hide.

We have constructed CTIP, a straightforward extension of the TIP system based on treating remote storage as a reparameterized local disk. Though CTIP stretches TIP's simplistic storage model, TIP's cost-benefit buffer management strategies are robust enough to cope. CTIP reduces run times from 17% to 62% over our benchmark suite unaided by TIP or CTIP, and in every benchmark, the amount of time saved under CTIP exceeds the amount of time saved under local TIP. In terms of speedups, the geometric mean of CTIP speedups is only 10% less than that for TIP.

As the inset bar graphs demonstrate, for TIP to be used on shared data stored on servers that do not support computation, that data must first be copied to local storage on the client. In almost every case, the time required for this copy overwhelms the differential advantage of local TIP over CTIP. In this sense, CTIP is the right solution for network-storage client machines.

Bill's all business.

… continued from pg. 7



A comparison of run times (seconds) of the xdataslice benchmark running under CTIP (remote), TIP (local), and both local and remote unhinted accesses. All code and data was striped over four disks.

This prefetching hides disk latency and exposes storage parallelism. Preliminary measurements that show CTIP can reduce execution time by a ratio comparable to that obtained with local TIP over a suite of I/O-intensive hinting applications. (For four disks, the reductions in execution time range from 17% to 69%). If local TIP execution requires that data first be loaded from remote storage into a local scratch area, then CTIP execution is significantly faster than the aggregate time for loading the data and executing. Additionally, our measurements show that the benefit of CTIP for hinting applications improves in the face of competition from other clients for server resources. We conclude with an analysis of the remaining problems with using unmodified NFS servers.

Security for Network Attached Storage Devices

Gobioff, Gibson & Tygar

School of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-97-185, October 1997.

ABSTRACT

This paper presents a novel cryptographic capablity system addressing the security and performance needs of network attached storage systems in which file management functions occur at a different location than the file storage device. In our NASD system, file managers issue capabilities to client machines which can then directly access files stored on the network attached storage device without intervention by a file server. These capabilities may be reused by the client so that interaction with the file manager is kept to a minimum. Our system emphasizes performance and scalability while separating the roles of the decision maker (issues capabilities) and the verifier (validates a capability). We have demonstrated our system with adaptations of both the NFS and AFS distributed file systems using a prototype NASD implementation.

Informed Multi-Process Prefetching and Caching

Tomkins, Patterson & Gibson

Proceedings of the ACM Int'l Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), June 15-18, 1997.

ABSTRACT

Informed prefetching and caching based on application disclosure of future I/O accesses (hints) can dramatically reduce the execution time of I/O-intensive applications. A recent study showed that, in the context of a single hinting application, prefetching and caching algorithms should adapt to the dynamic load on the disks to obtain the best performance. In this paper, we show how to incorporate adaptivity to disk load into the TIP2 system, which uses cost-benefit analysis to allocate global resources among multiple processes. We compare the resulting system, which we call TIPTOE (TIP with Temporal Overload Estimators) to Cao et al.'s LRU-SP allocation scheme, also modified to include adaptive prefetching. Using disk-accurate trace-driven simulation we show that, averaged over eleven experiments involving pairs of hinting applications, and with data striped over one to ten disks, TIPTOE delivers 7% lower execution time than LRU-SP. Where the computation and I/O demands of each experiment are closely matched, in a two-disk array, TIPTOE delivers 18% lower execution time.

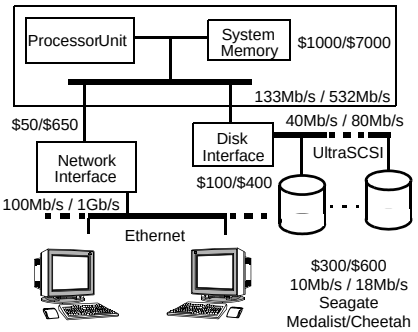
A Cost-Effective, High-Bandwidth Storage Architecture

Gibson, Nagle, Amiri, Butler, Chang, Gobioff, Hardin, Riedel, Rochberg & Zelenka

Proceedings of the 8th Conference on Architectural Support for Programming Languages and Operating Systems, 1998.

ABSTRACT

This paper describes the Network-Attached Secure Disk (NASD) storage architecture, prototype implementations of NASD drives, array management for our architecture, and three filesystems built on our prototype. NASD provides scalable storage bandwidth without the cost of servers used primarily for transferring data



Cost model for traditional server architecture. Reporting pairs of cost and bandwidth estimates, values for low cost system built from high volume components are on the left. On the right, values for high performance system built from components for mid-range servers.

… continued on pg. 19

… continued from pg. 1

through networked desktop computers.

Terabytes of data should not be very expensive. Consulting nearly any computing magazine today, we expect a megabyte of magnetic disk storage to cost about five cents, so a terabyte should cost about $50,000. Of course, packaging these raw disks into cabinets with power and cooling typically doubles the cost to $100,000 per terabyte. Using 4 GB disks typically available today, a terabyte store has 250 devices, each capable of sustaining in excess of 10 MB/s. Thus, our retail researchers might expect to be able to compile simple scan-based statistics over all data at 2.5 GB/s. Bandwidth, however, is quite expensive; servers capable of accessing disks and forwarding data to networked clients at this rate will easily cost more than the packaged cost of the terabyte. As well, the network, even with today's low-cost ethernet switches, will cost more than the packaged cost of the terabyte for the storage side alone, and that much again for the client side of the network switch ports. Estimates of the yearly cost of operating such a storage system range widely, but all exceed the capital cost of the (packaged) storage itself: a networked storage infrastructure capable of storing and accessing a terabyte at maximal bandwidth will cost in excess of eight times the base storage cost, or $400,000 today, to build and will cost in excess of twice the base storage cost, or $100,000 today, per year to operate.

Let's reconsider processing that simple statistics scan at 2.5 GB/s. No individual affordable personal computer receives more than about 100 MB/s, and few can handle even simple computations at that rate. If our retail researcher is very lucky and privileged, the company has one or two hundred idle workstations on the network and the tools to convert these to execute his operation in parallel. More likely our researchers either work with samples or summaries of the data or they may induce the company to spend a million dollars or more on supercomputer data mining servers with massive memory bandwidth and processor parallelism. In practical terms, even simple processing of a terabyte information store costs 10, 20, perhaps 40 times the cost of the raw storage.

Is there an alternative? Reconsider the raw storage system. Each of our retail company's 250 disk devices has a memory and processor with access to the raw bandwidth. As Figure 1 on page 1 infers, each *Active Disk's* processor may soon be only a generation or two slower than the fastest microprocessors in workstations and personal computers. So a raw storage system of 250 disks may soon have the computational power of more than 60 state-of-the-art microprocessors and have no limitation on access to full disk bandwidth (each raw storage disk being two to four times slower than the microprocessors mentioned). This is substantially more power than our retail researcher has on his or her desk. Further, adding storage capacity to a system adds computational power and bandwidth between storage and computation: inherent opportunities for scalable systems. Thus, an individual with a desktop computer networked to massively parallel Active Disk storage is equipped with extremely cheap supercomputing power for massive scan operations, where an Active Disk is a NASD drive with excess computational power and a remote execution environment used by application programs to customize disk function.

Preliminary Exploration with Data Mining and Multimedia Applications

Using our NASD testbed system with application code "shoehorned" into the NASD code, we have done some preliminary experiments to indicate the potential of Active Disks. Specifically, three real-world data-intensive data mining and multimedia applications were recoded so that their data-intensive kernels were partitioned to run in modified NASD drives

First, we recoded the *Apriori* algorithm for discovering association rules in sales transactions. For the Active Disks system, the counting portion of each phase of the algorithm is performed directly at the drives. The central server produces the list of candidate *k*-itemsets (a set of *k* retail products) and provides this list to each of the disks. Each disk counts occurrences of this itemset in its portion of the transactions locally, and returns these counts to the server. The server then combines these counts and produces a list of candidate "popular" *(k+1)*-itemsets which are then sent back to the disks for the next iteration.

For comparison purposes in these experiments, we model the host as a fast, but affordable, database server (500 MHz AlphaStation) with a variable number of (traditional or Active) disks. In Figure 2a, we see the implementation results of this comparison for the first two passes of the frequent sets application (*1-itemsets* and *2-itemsets*). Active Disk Apriori performance increases linearly with more disks, achieving a speedup of almost 3 at ten disks.

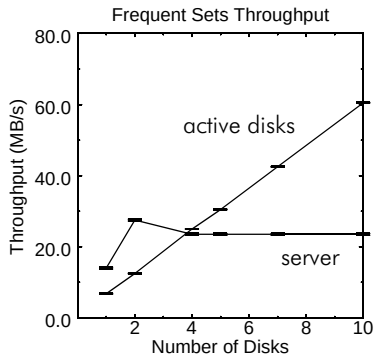


Figure 2a: The Apriori frequent sets application shows linear scaling to 60 MB/s while the server system bottlenecks at 24 MB/s

… continued on pg. 16

Active Disks

… continued from pg. 15

Next, we looked at image processing, a simple multimedia application that detects edges and corners in a set of grayscale images. Each Active Disk processes a set of 256 KB images and returns to the central host only the edges found in the data. This application is significantly more computation-intensive than the counting of the Apriori application, however, as shown in Figure 2b, it too experiences speedups of over 2 on an array of 10 disks.

Finally, we examined a variation of a standard database search that determines the *k* vectors in a database of vectors that are closest to a particular input vector. The application sequentially processes vectors from the high-dimensionality database, always keeping a list of the *k* closest matches up to that point. For the Active Disks system, each disk is assigned an integral number of vectors and the comparisons are performed directly at the drives. The server sends the target vector to each of the disks which determine the closest *k* vectors in their portions of the database. These lists are returned to the server where they are combined and the closest *k* vectors overall are noted. As Figure 2c shows, this database search application displays similar advantages: a factor of over 2 speedup with 10 disks.

Speedup Model for Scanning with Active Disks

There is a general pattern to be found in the results of our experiments, illustrated in Figure 3. For a small number of disks, the host server system performs better because its CPU is four times as powerful as a single Active Disk processor and because its interconnect is faster than a single disk. However, the host soon saturates its CPU or interconnect, while an Active Disk Array continues to process faster with more disks until it saturates its interconnection network with the filtered output or until serial components in the application begin to dominate.

These preliminary results are very promising for Active Disks. Applications that admit parallel, data-intensive codes with small memory footprints whose output is much smaller than their input -- a fair characterization of simple statistics, neural net training, most image processing and a significant chunk of database queries -- and can achieve linear speedups with more Active Disks.

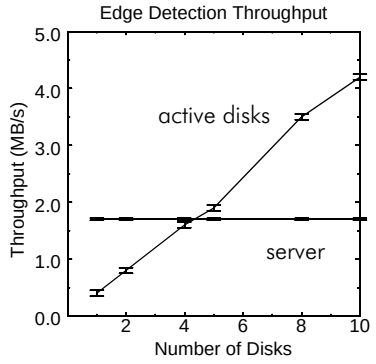


Figure 2b: The edge detection application shows linear scaling with number of disks while the server system bottlenecks at about 1.7 MB/s.

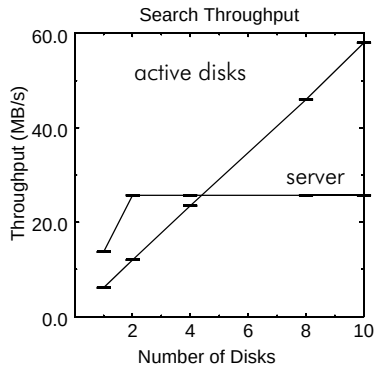


Figure 2c: The search application shows linear scaling with number of disks up to 58 MB/s, while the server system bottlenecks at 26 MB/s.

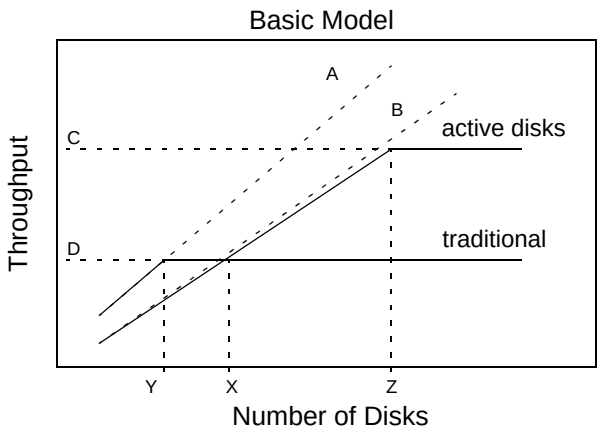


Figure 3: Simple model of the throughput of an application running in an Active Disk system compared to a traditional single server system. There are several regions of interest, depending on characteristics of the application and the underlying system configuration. The aggregate raw media rate of all disks in both cases is plotted as line A. The aggregate raw computation rate in the Active Disk system is line B, which varies by application. Line C shows the saturation of the interconnect between the disks and host. Line D represents the saturation of the server CPU in the traditional system, above which no further gain is possible as disks are added. The "sweet spot", where the number of disks is between X and Z, shows linear speedup for Active Disks over traditional systems. To the left of point Y, the traditional system is disk-bound. Above point Z, which is larger for applications that return less of their data to their host, even the Active Disk system is network-bottlenecked and no further improvement is possible. Below the crossover point X, the Active Disk system is slower than the server system due to its less powerful CPUs. With disk CPUs two generations behind host CPUs, X is about 4 times the number of host CPUs.In our experiments, Z numbered from tens to thousands.

… continued from pg. 9

versity of Wisconsin, and appeared at the 1996 Symposium on Operating Systems Design and Implementation. Our primary conclusion was that, in the context of a single application providing hints about its upcoming I/O's, prefetching and cache management should dynamically adjust to the changing load on the disks in order to attain the best performance. These results suggested that the estimators on the bottom row of the cache manager schematic, for informed prefetching and informed caching, could be improved by incorporating a more advanced model of disk load. Our Sigmetrics paper performed these modifications, and evaluated TIPTOE, the resulting algorithm.

TIPTOE

Figure 2 shows how TIPTOE determines which disks are overloaded. The upper bar represents the future request sequence; black segments represent requests to cached blocks, and all other segments represent requests for missing blocks on one of the three disks. The three lower bars represent schedules for each disk that satisfy all requests in time without incurring stall. Disk *a* represents the ideal situation because prefetching for each block can begin one fetch time before the block will be consumed. Disk *b* contains a burst of requests *b1*, *b2* and *b3*, but we do not need to begin fetching those blocks until we reach the line marked "disk *b* constrained". The schedule for disk *c* shows that in order to service all requests without stall, we would have to begin prefetching in the past, and therefore we will incur stall at some point. We say that disk *c* is currently the only overloaded disk of the three.

The resulting information about disk overload is then incorporated into the estimators for the benefit of performing informed prefetching, and the cost of evicting data from the hinted cache. We also incorporated information about disk overload into another system from the literature: LRU-SP/Aggressive, developed by Pei Cao and collaborators at Princeton and the University of Wisconsin, resulting in an algorithm called LRU-SP/Forestall. We performed a series of detailed comparisons between TIPTOE and LRU-SP/Forestall, and also compared their predecessors, TIP and LRU-SP/Aggressive. TIP and TIPTOE use cost-benefit analysis, while the other systems use an approach based on extending the LRU cache replacement algorithm to incorporate application hints.

Results

We evaluated these algorithms using trace-driven simulation. Our simulator is multi-threaded to allow simultaneous playback of multiple traces, and contains an accurate internal disk simulator that supports a number of data layout options across an array of disks (RAID0 for the results given below). The applications we trace were used to evaluate the original TIP system; they are drawn from a wide range of I/O intensive computing and engineering domains.

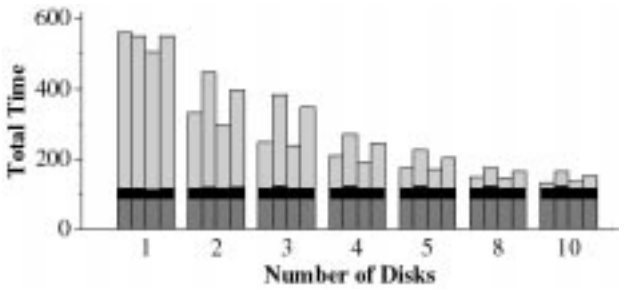


Figure 3: XDS and POSTGRES

Figure 3 is an example; it shows XDS, a scientific visualization tool rendering 2-dimensional planes through a large 3-dimensional dataset, running simultaneously with POSTGRES, a relational database performing a join. These two applications must share the buffer cache. As the figure shows, TIP and TIPTOE correctly infer, under cost-benefit analysis, that XDS has little re-use and therefore should not be given a large fraction of the buffer cache, while POSTGRES has significant re-use and should be given a large fraction of the cache. The LRU-SP algorithms do not perform this evaluation and performance suffers.

To summarize, TIPTOE outperforms LRU-SP with disk load information by about 6% on average across all experiments and array sizes, and outperforms LRU-SP without disk load information by 12%. TIP and TIPTOE perform similarly on large disk arrays, where disk overload does not tend to occur, but TIPTOE outperforms TIP by about 5% on average on a single disk where the new estimators are most effective.

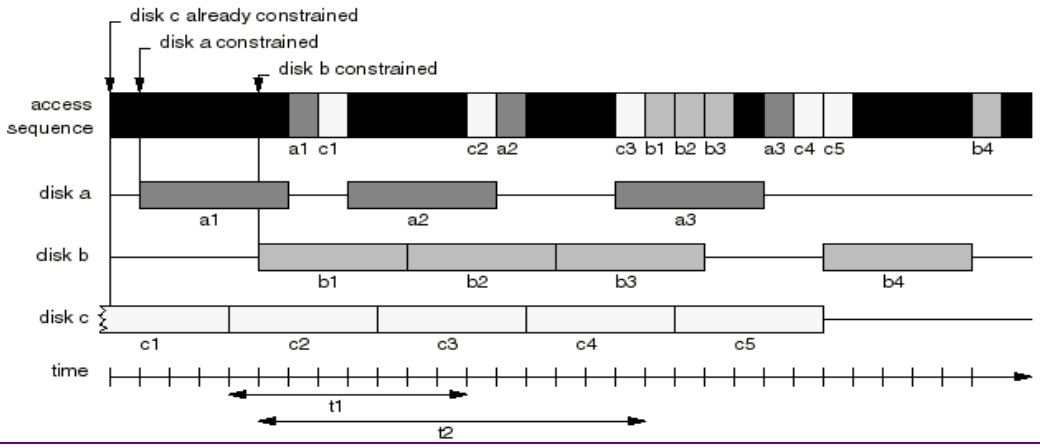


Figure 2: How TIPTOE finds overloaded disks

Active Storage Nets

David Nagle
http://www.ece.cmu.edu/~asn

Seeking to leverage the synergy that Network-attached Secure Disks (NASD) creates between storage and networking technologies, researchers in the Parallel Data Lab have launched a new, DARPA funded research program called Active Storage Networks. The project's goal is to enable flexible construction of sophisticated storage and file-system functionality that can migrate to the most appropriate location in the system (e.g., client, router, or NASD).

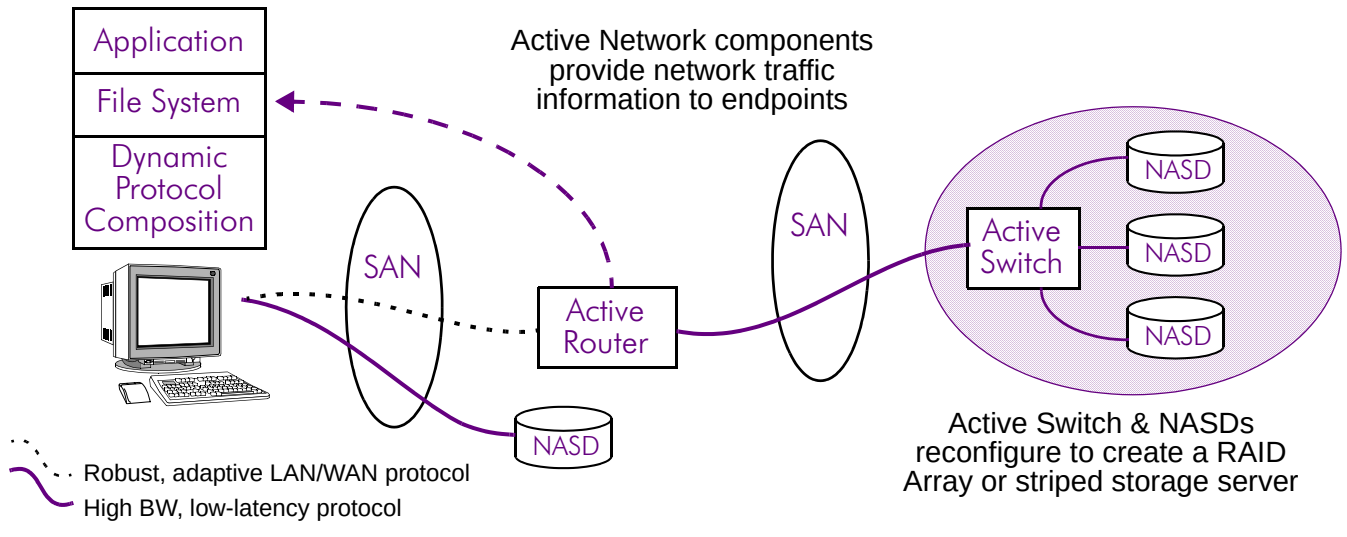
For the last several years, the PDL's work on NASD has examined how to exploit computational cycles in storage devices. The results have shown that NASD's high-level storage interface and self-management capability creates highly-scalable storage systems.

More recently, work at MIT, the University of Pennsylvania and other universities has focused on exploiting excess cycles in the network to create Active Nets. Emphasizing both specialized hardware-based function in switches and routers and software-based functionality, preliminary work has shown that Active Nets can enable the rapid deployment of new networking technologies and increase scalability of networks through sophisticated traffic management and caching policies.

We believe Active Storage Networks can play a significant role in the deployment of NASD through the integration of the two technologies. For example, video places significant real-time constraints on data movement through the network and within storage. An Active Network that understands the scheduling requirements should be able to ensure timely data delivery. However, if the data source (i.e. storage) is not ready to transmit or receive the data, then scheduling will fail. Likewise, NASD attempts to solve this problem by integrating real-time scheduling into storage and will fail if the network is unable to deliver data. An integrated Active Storage Network allows the NASDs and Network to cooperate to provide end-to-end delivery guarantees.

Further, if Active Networks can understand the NASD object model, the network will be able to make intelligent policy decisions based on an entire object (or set of objects) and not just on individual packets. This will allow current network-based caching technologies to cache complete NASD objects, or to cache enough of an object to hide initial fetch latencies. Further, integrating NASD's security model with the network will allow caching nodes to provide the same degree of security as storage, creating a complete end-to-end security model while enabling network-based caching.

Active Storage Nets can also enable SAN-, LAN-, and WAN-based communication between client and storage. Instead of relying on the NASD to provide both a highly-optimized SAN protocol and highly robust, wide-area protocol (e.g., TCP/IP), an active network component can serve as a protocol converter. Relying on information about physical network characteristics and traffic patterns, the client and protocol converter will adapt the protocol to minimize the load on the network node while enabling direct client-storage communication.



Active Storage Networks leverage a close integration between the network and NASD devices, to enable storage-based functionality that can migrate to the most appropriate location including client, storage and the network components.

… continued from pg. 14

from peripheral networks (e.g. SCSI) to client networks (e.g. ethernet). Increasing dataset sizes, new attachment technologies, the convergence of peripheral and interprocessor switched networks, and the increased availability of on-drive transistors motivate and enable this new architecture. NASD is based on four main principles: direct transfer to clients, secure interfaces via cryptographic support, asynchronous non-critical-path oversight, and variably-sized data objects. Measurements of our prototype system show that these services can be cost-effectively integrated into a next generation disk drive ASIC.

End-to-end measurements of our prototype drive and filesystems suggest that NASD can support conventional distributed filesystems without performance degradation. More importantly, we show scalable bandwidth for NASD-specialized filesystems. Using a parallel data mining application, NASD drives deliver a linear scaling of 6.2 MB/s per client-drive pair, tested with up to eight pairs in our lab.

Active Storage For Large-Scale Data Mining and Multimedia

Riedel, Gibson & Faloutsos

Appears in the Proceedings of the 24th international Conference on Very Large Databases (VLDB '98), New York, NY, August 24-27, 1998.

ABSTRACT

The increasing performance and decreasing cost of processors and memory are causing system intelligence to move into peripherals from the CPU. Storage system designers are using this trend toward "excess" compute power to perform more complex processing and optimizations inside storage devices. To date, such optimizations have been at relatively low levels of the storage protocol. At the same time, trends in storage density, mechanics, and electronics are eliminating the bottleneck in moving data off the media and putting pressure on interconnects and host processors to move data more efficiently. We propose a system called Active Disks that takes advantage of processing power on individual disk drives to run application-level code. Moving portions of an application's processing to execute directly at disk drives can dramatically reduce data traffic and take advantage of the storage parallelism already present in large systems today. We discuss several types of applications that would benefit from this capability with a focus on the areas of database, data mining, and multimedia. We develop an analytical model of the speedups possible for scan-intensive applications in an Active Disk system. We also experiment with a prototype Active Disk system using relatively low-powered processors in comparison to a database server system with a single, fast processor. Our experiments validate the intuition in our model and demonstrate speedups of 2x on 10 disks across four scan-based applications. The model promises linear speedups in disk arrays of hundreds of disks, provided the application data is large enough.

System	Component	Processor	On-Disk Processing	System Bus	Storage Throughput
Compaq TPC-C	Compaq ProLiant 7000 6/200 4 200 MHz Pentiums, 1 PCI 113 disks = 708 GB	800 MHz (4 x 200 MHz)	2,825 MHz (113 x 25 MHz)	133 MB/s	1,130 MB/s (113 x 10 MB/s)
Microsoft Terra Server	Digital AlphaServer 4100 4 400 MHz Alphas, 2 64-bit PCI 320 disks = 1.3 TB	1,600 MHz (4 x 400 MHz)	8,000 MHz (320 x 25 MHz)	532 MB/s	3,200 MB/s (320 x 10 MB/s)
Digital TPC-C	Digital AlphaServer 1000/500 500 MHz Alpha, 64-bit PCI 61 disks = 266 GB	500 MHz	1,525 MHz (61 x 25 MHz)	266 MB/s	610 MB/s (61 x 10 MB/s)
Digital TPC-D	Digital AlphaServer 4100 4 466 MHz Alphas, 2 64-bit PCI 82 disks = 353 GB	1,864 MHz (4 x 466 MHz)	2,050 MHz (82 x 25 MHz)	532 MB/s	820 MB/s (82 x 10 MB/s)

If we estimate that current disk drives have the equivalent of 25 MHz of host processing speed available, large database systems today already contain more processing power on their combined disks than at the server processors. Assuming 10 MB/s for sequential scans, we also see that the aggregate storage bandwidth is more than twice the backplane bandwidth of the machine in almost every case.

Congratulations to Hugo on receiving his Ph.D.

NSIC-NASD Meeting Agendas: 1997-1998

Informally chaired by CMU's Garth Gibson, the working group on Network Attached Storage Devices in the National Storage Industry Consortium (NSIC) sponsors a quarterly public workshop. Following are the agendas from the past year. Online versions of most of these presentations are avail-able on the NSIC web site at http://www.ncis.org/nasd/meetings.html/

NASD Meeting: September 25, 1997

Introduction to NASD from CMU's Perspective: Garth Gibson, CMU

Storage Area Network Frontrunners

- Fibre Channel Overview: Jim Coomes, Seagate
- An Implementation of the Hamlyn Sender-managed Interface Architecture: Richard Golding, HP Labs

System Area Network Programming Models

- The VI Architecture and SAN IO: Don Cameron, Intel
- IEEE 1394 High Performance Serial Bus: Mike Bryan and Joe Wach, Seagate

SANs with LAN Technology

- Overview of Gigabit Ethernet Technology: Wen-Tsung Tang, 3Com
- Characterization of VIA Compliant Interfaces: David Follett, gigaNET, Inc.

Potpourri

- Nexus: Ian Foster, Argonne National Labs
- ServerNet SAN Update: Bob Horst, Tandem

NASD Meeting: December 9, 1997

Setting the Stage: Garth Gibson, CMU

Network Filesystems and Databases

- Database Applications and System Storage: Catharine van Ingen, Microsoft
- DFS: Pete Messner, Transarc
- XFS: Geoff Peck, Quantum

Smart Storage Support for Filesystems

- NFS to AFS Port for CMU's NASD: Garth Gibson, CMU
- Shared Storage Solution: Craig Lund, Mercury Computer Systems
- Network Storage - A Unique Solution: Bill Bullers, ImpactData
- A Shared Disk File System for a Cluster of IRIX Workstations: Matthew O'Keefe, University of Minnesota

NASD Meeting: March 5, 1998

NSIC Technical Program Introduction: Barry Schechtman, National Storage Industry Consortium

From the Working Groups

- An Introduction to Network-Attached Storage Devices: CMU's Perspective: Garth Gibson, CMU
- Network Attached Storage Research: Dave Anderson, Seagate
- Attribute-based Storage Management: Liz Borowsky, Hewlett Packard
- CMU's NASD: Network-Attached Secure Disks: Garth Gibson, CMU
- Information Security for Network Attached Storage Research: Jim Hughes, StorageTek
- Network Storage Manager: Greg VanHise, IBM
- IEEE Media Management System (P1244): A Brief Introduction and Storage Objects: Geoff Peck, Quantum

From Our Colleagues

- ISI's NASD: Derived Virtual Devices: Rod Van Meter, Quantum
- The Swarm Scalable Storage System: John Hartman, University of Arizona
- Petal: Distributed Virtual Disks: Ed Lee, Systems Research Center, Digital Equipment Corporation
- Benchmarking and I/O Subsystems Presentation and Round Table Discussions: Roger Reich, Digital Equipment Corporation

NASD Panel Discussion, moderated by Paul Borrill, SNIA Chairman

- Dave Anderson, Seagate
- Liz Borowsky, Hewlett Packard
- Garth Gibson, CMU
- John Hartman, University of Arizona
- Jim Hughes, STK
- Ed Lee, DEC-SRC
- Geoff Peck, Quantum
- Greg VanHise, IBM
- Rod Van Meter, Quantum
- Don Cameron, Intel
- Gene Freeman, Compaq
- Jerry Fredin, Symbios
- Dave Hitz, Network Appliance
- Kim Minuzzo, Lawrence Livermore National Labs
- Percy Tzelnic, EMC
- Ed Zayas, Novell

NASD Meeting: June 8 - 9, 1998

Application Code in the Disk

- Workshop Introduction: What Is To Be Done With Lots More Computing Inside Storage? Garth Gibson, CMU
- Put EVERYTHING in the Storage Device: Jim Gray, Microsoft Research
- Active Disks for Data Mining and Multimedia: Erik Riedel, CMU
- Intelligent Disks: A New Computing Infrastructure for Decision Support Databases: Kimberly Keeton, University of California, Berkeley
- Active Disk Architectures for Rapidly Growing Datasets: Anurag Acharya, University of California, Santa Barbara

Storage And File Systems Support In The Disk

- Consideration for Smarter Storage Devices: David Anderson, Seagate
- SCSI Disk Requirements for Shared Disk File Systems: Matthew O'Keefe, University of Minnesota
- NFS v4 and Compound Requests: Brent Callaghan, Sun
- A File System for Intelligent Disks: Randy Wang, University of California, Berkeley

NASD Meeting: September 1, 1998

Introduction and Active Storage Nets: Dave Nagle, CMU

Networking for Storage

- VISA: Netstation's Virtual Internet SCSI Adapter: Rod Van Meter, Quantum & USC/Information Sciences Institute
- Fibre Channel and SANs: Norm Chase, Brocade
- I20 over VI Architecture: Don Cameron, Intel
- VI Architecture and Giganet: David Follett, gigaNET
- Devices on the Desk Area Network (DAN): Ian Pratt, University of Cambridge
- Authentication for Network Attached Storage: Ben Reed, IBM

Garth was in town...really.