Active Storage For Large-Scale Data Mining and Multimedia

Erik Riedel
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
riedel@cs.cmu.edu

Garth Gibson
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
garth@cs.cmu.edu

Christos Faloutsos
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
christos@cs.cmu.edu

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.

This research was sponsored by DARPA/ITO through Order D306, and issued by Indian Head Division, NSWC under contract N00174-96-0002. Additional support was provided by NSF under grants EEC-94-02384 and IRI-9625428 and NSF, ARPA and NASA under NSF Agreement IRI-9411299. We are also indebted to generous contributions from the member companies of the Parallel Data Consortium: Hewlett-Packard Laboratories, Symbios Logic, Data General, Compaq, Intel, Quantum, Seagate, Wind River Systems, and StorageTek. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any supporting organization or the U.S. Government.

Proceedings of the 24th VLDB Conference, New York, USA, 1998

1 Introduction

In this paper we evaluate the performance advantages of exploiting the processors embedded in individual storage device for some of the data-intensive applications common in data mining and multimedia databases. This system is architecturally similar to the processor-per-disk database machines dismissed in the literature 15 years ago as expensive and unnecessary. In the intervening years, technology trends have made possible commodity storage devices with excess general-purpose computational power and application trends are creating massive, complex data sets commonly processed with scans. It will soon be possible for collections of commodity storage devices to couple parallel processing and high-selectivity filtering to dramatically reduce execution time for many of these applications.

General purpose microcontrollers with 100-200 MHz processing speeds are common in disk array controllers and are already being incorporated into high-end commodity disk drives. Vendors of storage devices would welcome new uses for this largely underutilized processing power if it allowed their products to compete on metrics beyond simple capacity and cost ($/MB). We propose a storage device called an *Active Disk* that combines in-the-field software downloadability with recent research in safe remote execution of code for execution of application-level functions directly at the device.

In this paper, we emulate an Active Disk with a six-year-old workstation and contrast host-resident to Active-Disk-assisted processing of four applications: nearest neighbor search in a high dimensionality database, frequent set counting to discover association rules, edge detection in images, and image registration in a medical database. These applications all process large volumes of data, ensuring substantial storage parallelism simply to accommodate the volume of data, and often operate with a relatively small number of instructions per byte of storage accessed. The processing in all these applications is scan-intensive, either due to the nature of the application and data, or because the high-dimensionality queries being processed are not accelerated by traditional indices.

Active Disks benefit I/O-bound scans in two principle ways: 1) parallelism - massive amounts of data partitioned over many disks allows "embarrassingly parallel" scans to convert a group of Active Disks into a programmable parallel-scan database machine, 2) bandwidth reduction - scans that filter data with a high degree of selectivity or compute only summary statistics transfer a very small fraction of the data from the disks to the host. For highly selective scans, a group of Active Disks can process data at the aggregate disk rate in a machine whose interconnect bandwidth was designed for applications demanding much less bandwidth.

Section 2 compares our work with past research on database processing performed at storage (i.e. database machines), discusses the trends in storage systems that have brought us to this point, and motivates the areas of data mining and multimedia as fertile ground for applications of Active Disks. Section 3 provides an analytical model to illustrate the potential benefit of using Active Disks and give some intuition on the speedups possible. Section 4 outlines the four representative applications we have chosen for detailed study. Section 5 describes our experimental setup and compares the performance of an existing server system to a prototype system using Active Disks. Section 6 further explores issues of performance and the characteristics of applications that make them successful on Active Disks. Section 7 discusses related work in the area. Finally, Section 8 concludes and briefly discusses areas of future work.

2 Background

The prevailing counter-arguments to the database machines of the 80s were that 1) for a significant fraction of database operations, such as sorts and joins, simple select filters in hardware did not provide significant benefits, 2) special-purpose hardware increased the design time and cost of the machine, and 3) a single general purpose host processor was sufficient to execute select at the full data rate of a single disk [DeWitt81, Boral83].

Boral and DeWitt concluded that aggregate storage bandwidth was the principle limitation of database machines. Fortunately, as shown in Table 1, in the intervening years aggregate storage bandwidth has dramatically improved. The improvement comes from disk array hardware and software that enable individual database operations to exploit disk parallelism [Livny87, Patterson88] and because databases are now large enough to justify hundreds of disks. Moreover, high-end disk rates are now 15 MB/s sustained [Seagate97] and continue to grow at 40% per year [Grochowski96]. In place of raw disk bandwidth limitations, modern systems have a limited peripheral interconnect bandwidth, as seen in the system bus column of Table 1. We see that more MB/s can be read into the memory of a large collection of disk controllers than can be delivered to a host processor. In this case, the power of the host is irrelevant to the overall bandwidth limitation for large scans.

If we next consider the objection to the cost and complexity of special-purpose hardware in database machines, technology trends again change the trade-offs. The increasing transistor count possible in inexpensive CMOS microchips today is driving the use of microprocessors in increasingly simple and inexpensive devices. Network interfaces, peripheral adapters, digital cameras, graphics adapters, array controllers and disk drives all have microcontrollers whose processing power exceeds the host processors of 15 years ago. For example, Quantum's high-end disk drives today contain a 40 MHz Motorola 68000-based controller that manages the high-level functions of the drive.

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

Table 1: 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 a reasonable 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. Data from [TPC98] and [Barclay97].

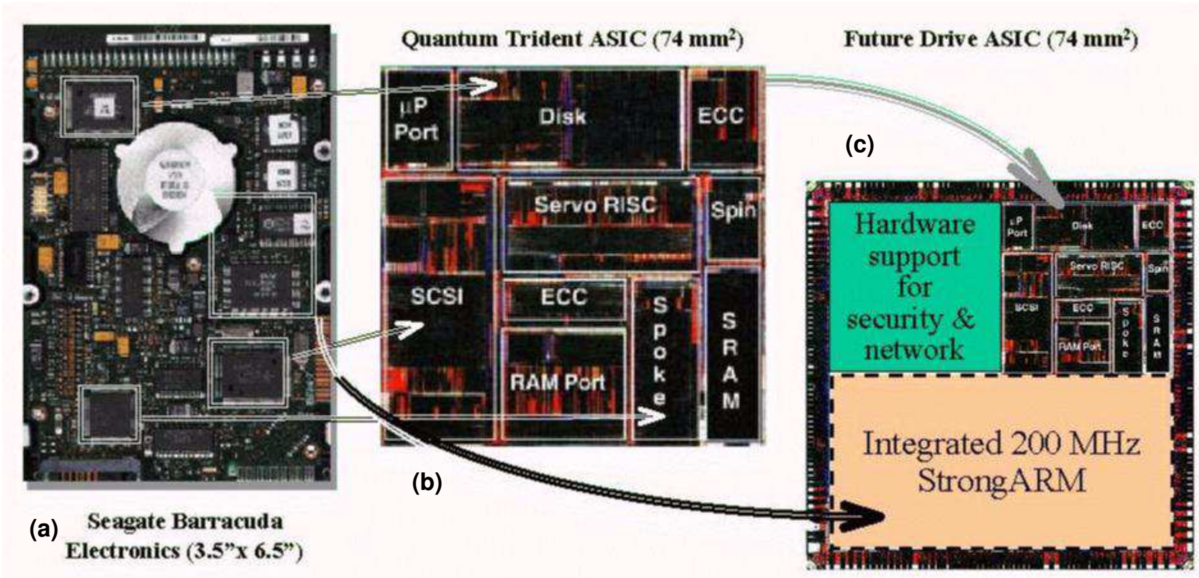


Figure 1: The trend in drive electronics is toward higher and higher levels of integration. The Barracuda drive on the left contains separate chips for servo control, SCSI processing, ECC, and the control microprocessor. The Trident chip in the center has combined many of the individual specialized chips into a single ASIC, and the next generation of silicon makes it possible to both integrate the control processor and provide a significantly more powerful embedded core while continuing to reduce total chip count and cost.

In Figure 1 we show the effects of increasing transistor counts on disk electronics. Figure 1a reminds us that the electronics of a disk drive include all the components of a simple computer: a microcontroller, some amount of RAM, and a communications subsystem (SCSI), in addition to the specialized hardware for drive control. Figure 1b shows that this special control hardware has already been largely integrated into a single chip in current-generation disks. Extrapolating to the next generation of technology (from .68 micron to .35 micron CMOS in the ASIC), the specialized drive hardware will occupy about one quarter of the chip, leaving sufficient area to include a 200 MHz Digital StrongARM microprocessor [Turley96], for example. Commodity disk and chip manufacturers are already pursuing processor-in-ASIC technology. Siemens has announced a chip that offers a 100 MHz 32-bit microcontroller, up to 2 MB of on-chip RAM with up to 800 MB/s bandwidth, external DRAM and DMA controllers and customer-specific logic (that is, die area for the functions of Figure 1b) in a .35 micron process [TriCore97]. Fundamentally, VLSI technology has evolved to the point that significant additional computational power comes at negligible cost.

Processing power inside drives and storage subsystems has already been successfully used to optimize functions behind standardized interfaces such as SCSI. This includes many innovative optimizations for storage parallelism, bandwidth and access time [Patterson88, Drapeau94, Wilkes95, Cao94, StorageTek94] and for distributed file system scalability [Lee96, VanMeter96, Gibson97]. With Active Disks, excess computation power in storage devices is available directly for application-specific function in addition to supporting these existing storage-specific optimizations. Instead of etching database functions into silicon as envisioned 15 years ago, Active Disks are programmed in software and use general purpose microprocessors.

Downloading application code directly into devices has significant implications for language, safety, and resource management [Riedel97]. With block-oriented application codes, it is efficient to exploit standard memory management hardware at the drive and provide protected address spaces for applications as in standard multiprogrammed systems today. For the cases where efficiency, space or cost constraints require that application code be co-located with "core" drive code, recent research offers a range of efficient and safe remote execution facilities that provide innovative ways to ensure proper execution of code and safeguard the integrity of the drive [Gosling96, Necula96, Romer96, Bershad95, Small95, Wahbe93]. Some of these mechanisms also promise a degree of control over the resource usage of remote functions to aid in balancing utilization of the drive between demand requests, opportunistic optimizations such as readahead, and demand requests.

The third objection to database machines was the limited utility of full scan operations. However, a variety of emerging applications require sequential scanning over large amounts of data. We focus on two sets of applications: multimedia and data mining. In multimedia, applica-

Application Parameters	System Parameters	Active Disk Parameters
N_{in} = number of bytes processed	s_{cpu} = CPU speed of the host	s_{cpu}' = CPU speed of the disk
N_{out} = number of bytes produced	r_d = disk raw read rate	r_d' = active disk raw read rate
w = cycles per byte	r_n = disk interconnect rate	r_n' = active disk interconnect rate
t = run time for traditional system		
t_{active} = run time for active disk system		

Traditional vs. Active Disk Ratios

$\alpha_N = N_{\text{in}} / N_{\text{out}}$ $\quad \alpha_d = r_d' / r_d$ $\quad \alpha_n = r_n' / r_n$ $\quad \alpha_s = s_{\text{cpu}}' / s_{\text{cpu}}$

tions such as searching by content [Flickner95, Virage98] are particularly good candidates. The user provides a desirable image and requests a set of similar images. The general approach to such a search is to extract feature vectors from every image, and then search these feature vectors for nearest neighbors [Faloutsos96]. The dimensionality of these vectors may be high (e.g. moments of inertia for shapes [Faloutsos94], colors in histograms for color matching, or Fourier coefficients). It is well-known [Yao85], but only recently highlighted in the database literature [Berchtold97], that for high dimensionalities, sequential scanning is competitive with indexing methods because of the "dimensionality curse." Conventional database wisdom is that indices always improve performance over scanning. This is true for low dimensionalities, or for queries on only a few attributes. However, in high dimensionality data and nearest neighbor queries, there is a lot of "room" in the address space and the data points are far from each other. The two major indexing methods, grid-based and tree-based, both suffer in high dimensionality data. Grid-based methods require exponentially many cells and tree-based methods group similar points together, resulting in groups with highly overlapping bounds. One way or another, a nearest neighbor query will have to visit a large percentage of the database, effectively reducing the problem to sequential scanning. This is exactly the idea behind recent high-dimensionality indexing methods such as X-trees [Berchtold96] which deliberately revert to sequential scanning for high dimensionalities.

In data mining, algorithms such as association discovery and classification also require repeated scans of the data [Agrawal96].

In addition to supporting complex, scan-based queries, trends are toward larger and larger database sizes. One hour of video requires approximately 1 GB of storage and video databases such as daily news broadcasts can easily contain over 1 TB of data [Wactlar96]. Such databases can be searched by content (video, text, or audio) and utilize both feature extraction and a combination of the searching algorithms mentioned above. Medical image databases also impose similarly heavy data requirements [Arya94]. In data mining applications, point-of-sale data is collected over many months and years and grows continually. Telecommunication companies maintain tens of TB of historical call data. Large databases mean many disks, and therefore, highly parallel Active Disk systems.

3 Basic Approach

The basic characteristics of successful remote functions for Active Disks are that they 1) can leverage the parallelism available in systems with large numbers of disks, 2) operate with a small amount of state, processing data as it "streams past" from the disk, and 3) execute a relatively small number of instructions per byte.

In this section we develop an analytical model for the performance of such applications. The purpose of this model is to develop an intuition about the behavior of Active Disk systems relative to a traditional server.

To keep the model simple, we assume that our applications have the three characteristics mentioned above, that disk transfer, disk computation, interconnect transfer and host computation can be pipelined and overlapped with negligible startup and post-processing costs, and that interconnect transfer rates always exceed single disk rates.

Starting with the traditional server, overall run time is the largest of the individual pipeline stages: disk read time, disk interconnect transfer time, and server processing time which gives:

$$t = max\left(\frac{N_{\text{in}}}{d \cdot r_d}, \frac{N_{\text{in}}}{r_n}, \frac{N_{\text{in}} \cdot w}{s_{\text{cpu}}}\right) \quad \text{and}$$

$$\text{throughput} = \frac{N_{\text{in}}}{t} = min\left(d \cdot r_d, r_n, \frac{s_{\text{cpu}}}{w}\right)$$

For the Active Disks system, the comparable times for disk read, interconnect transfer, and on-disk processing are:

$$t_{active} = max\left(\frac{N_{\text{in}}}{d \cdot r_d'}, \frac{N_{\text{out}}}{r_n'}, \frac{N_{\text{in}} \cdot w}{d \cdot s_{\text{cpu}}'}\right) \quad \text{and}$$

$$\text{throughput}_{\text{active}} = \frac{N_{\text{in}}}{t_{active}} = min\left(d \cdot r_d', r_n' \cdot \frac{N_{\text{in}}}{N_{\text{out}}}, d \cdot \frac{s_{\text{cpu}}'}{w}\right)$$

Each of these throughput equations is a minimum of three limiting factors: the aggregate disk bandwidth, the storage interconnect bandwidth, and the aggregate computation bandwidth.

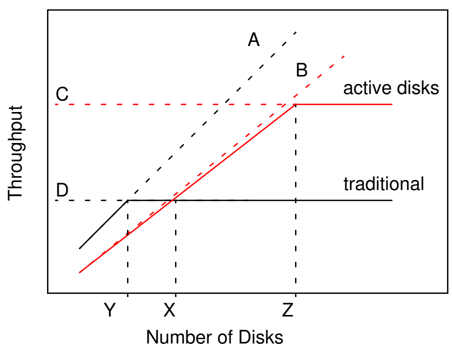


Figure 2: A simple model of the throughput of an application running in an Active Disk system compared to a traditional single server system. There are several regions of interest, depending on characteristics of the application and the underlying system configuration. The raw media rate of the disks in both cases is plotted as line A. The 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 Active Disks and host, which varies with the selectivity of the application processing. Line D represents the saturation of the server CPU or interconnect in the traditional system, above which no further gain is possible as additional disks are added. To the left of point Y, the traditional system is disk-bound. Below the crossover point X, the Active Disk system is slower than the server system due to its less powerful CPU. Above point Z, even the Active Disk system is network-bottlenecked and no further improvement is possible.

If we rewrite the equation for throughput with Active Disks in terms of the parameters of the traditional server and the ratios between traditional and Active Disk parameters - the total data moved (the selectivity α_N), the disk bandwidth (α_d, which should be 1), the interconnect bandwidth (α_n), and the relative CPU power (α_s), we have:

$$\text{throughput}_{\text{active}} = min\left(\alpha_d \cdot (d \cdot r_d), \alpha_N \cdot \alpha_n \cdot (r_n), d \cdot \alpha_s \cdot \left(\frac{s_{\text{cpu}}}{w}\right)\right)$$

This equation captures the basic advantages of Active Disks. Applications with high selectivity (large α_N) experience less restrictive interconnect limitations, and configurations with many disks ($d \cdot \alpha_s > 1$) can achieve effective parallel processing.

3.1 Estimating System Ratios

The kinds of applications we discuss here exhibit selectivities (α_N) of 100 to 10^8 or more, providing throughput possible only with effectively infinite interconnect bandwidth in the traditional system. Practically, this allows system cost to be reduced with lower bandwidth interconnects while maintaining high throughput. Therefore, we allow for slower Active Disk interconnects on the order of $0.1 < \alpha_n < 1.0$. Active Disk processors will be slower than traditional server CPUs. In our experiments, first generation Active Disk CPUs cannot scan data at disk rates.

The final and critical system parameter is the ratio of Active Disk to server processor speed. We expect 100 and 200 MHz microcontrollers in near-term high-end drives, and individual server CPUs of 500 to 1,000 MHz in the same time frame, so a ratio of about $\alpha_s = 1/5$ may be practical. In this case, the aggregate Active Disk processing power exceeds the server processing power once there are more than 5 disks working in parallel.

3.2 Implications of the Model

Figure 2 illustrates the basic trade-offs for Active Disk systems. The slope of line A represents the raw disk limitation in both systems. Because we expect that Active Disks will not be able to keep up with the disk transfer rates for many applications ($s_{\text{cpu}}' < w \cdot r_d'$), their aggregate throughput will have the somewhat lower slope shown by line B on the chart.

Active Disks saturate their interconnects at line C, with $\text{throughput}_{\text{active}} = r_n' \cdot \alpha_N \le min(d \cdot r_d', d \cdot s_{\text{cpu}}'/w)$. Since $x \ge min(x, y)$ and interconnect bandwidth is assumed to be greater than a single disk's bandwidth ($r_n' > r_d'$), the number of disks must be larger than the selectivity of the application ($r_n' \cdot \alpha_N < r_n' \cdot d$) before this limit sets in. This is shown to the right of point Z in the figure. With the large selectivities of the applications discussed here, we would expect our perfect overlap assumption to fail (Amdahl's Law) before this point is reached.

Traditional server systems are likely to exhibit both interconnect and server CPU bottlenecks, represented by line D in the figure. The point X in the figure, at which the Active Disk throughput exceeds the traditional server system is determined by $X \cdot s_{\text{cpu}}'/w = min(r_n, s_{\text{cpu}}/w)$, so $X \le s_{\text{cpu}}/s_{\text{cpu}}' = 1/\alpha_s$.

If we combine all of the above analysis and define speedup as Active Disk throughput over server throughput, we find that for $d < 1/\alpha_s$, the traditional server is faster. For $1/\alpha_s < d < \alpha_N$, the speedup is:

$$S = \frac{d \cdot (s_{\text{cpu}}'/w)}{min(r_n, s_{\text{cpu}}/w)} \ge d \cdot \alpha_s$$

and for $d > \alpha_N$, is:

$$S = \frac{(r_n' \cdot \alpha_N)}{min\left(r_n, \frac{s_{\text{cpu}}}{w}\right)} = max\left(\alpha_N \cdot \alpha_n, \alpha_N \cdot \alpha_s \cdot \left(\frac{w \cdot r_n'}{s_{\text{cpu}}'}\right)\right) > \alpha_N \cdot max(\alpha_n, \alpha_s)$$

for at least the first few generations of Active Disks.

We do not consider the "slowdown" of Active Disks when $d < 1/\alpha_s$ (the area to the left of point X in the figure), because this condition is independent of the application parameters, so a query optimizer can determine *apriori* when to prefer traditional execution of the scan for a particular system configuration, rather than executing the scan at the drives.

Finally, if we consider the prevailing technology trends, we know that the processor performance (B) improves by 60% per year and disk bandwidth (A) by 40% per year. This will cause the ratio of processing power to disk bandwidth in both systems to increase by 15% per year, narrowing the gap between line A and B, bringing Active Disks closer to the ideal total storage bandwidth.

We now look in greater detail at some specific applications that benefit from Active Disks.

4 Applications

In this study, we examine four real-world data-intensive data mining and multimedia applications that meet the assumptions of our Active Disks model.

4.1 Database - Nearest Neighbor Search

Our first application is a variation on a standard database search that determines the *k* items in a database of attributes that are closest to a particular input item. We use synthetic data from the Quest data mining group at IBM Almaden [Quest97] which contains records of individuals applying for loans and includes information on nine independent attributes: `<age>`, `<education>`, `<salary>`, `<commission>`, `<zip code>`, `<make of car>`, `<cost of house>`, `<loan amount>`, and `<years owned>`. In searches such as this across a large number of attributes, it has been shown that a scan of the entire database is as efficient as building extensive indices [Berchtold97]. Therefore, an Active Disk scan is appropriate. The basic application uses a target record as input and processes records from the database, always keeping a list of the *k* closest matches so far and adding the current record to the list if it is closer than any already in the list. Distance, for the purpose of comparison is the sum of the simple cartesian distance across the range of each attribute. For categorical attributes we use the Hamming distance, a distance of 0.0 is assigned if the values match exactly, otherwise 1.0 is assigned.

For the Active Disks system, each disk is assigned an integral number of records and the comparisons are performed directly at the drives. The central server sends the target record to each of the disks which determine the ten closest records in their portions of the database. These lists are returned to the server which combines them to determine the overall ten closest records. Because the application reduces the records in a database of arbitrary size to a constant-sized list of ten records, selectivity is arbitrarily large. Finally, the state required at the disk is simply the storage for the list of *k* closest records.

4.2 Data Mining - Frequent Sets

The second application is an implementation of the Apriori algorithm for discovering association rules in sales transactions [Agrawal95]. Again, we use synthetic data generated using a tool from the Quest group to create databases containing transactions from hypothetical point-of-sale information. Each record contains a `<transaction id>`, a `<customer id>`, and a list of `<items>` purchased. The purpose of the application is to extract rules of the form "if a customer purchases item A and B, then they are also likely to purchase item X" which can be used for store layout or inventory decisions. The computation is done in several passes, first determining the items that occur most often in the transactions (the *1-itemsets*) and then using this information to generate pairs of items that occur often (*2-itemsets*) and larger groupings (*k-itemsets*). The threshold of "often" is called the *support* for a particular itemset and is an input parameter to the application (e.g. requiring support of 1% for a rule means that 1% of the transactions in the database contain a particular itemset). Itemsets are determined by successive scans over the data, at each phase using the result of the *k*-itemset counts to create a list of candidate *(k+1)*-itemsets, until there are no *k*-itemsets above the desired support.

For the Active Disks system, the counting portion of each phase is performed directly at the drives. The central server produces the list of candidate *k*-itemsets and provides this list to each of the disks. Each disk counts 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 *(k+1)*-itemsets which are sent back to the disks. The application reduces the arbitrarily large number of transactions in a database into a single, variably-sized set of summary statistics - the itemset counts - that can be used to determine relationships in the database. The state required at the disk is the storage for the candidate *k*-itemsets and their counts at each state.

4.3 Multimedia - Edge Detection

For image processing, we looked at an application that detects edges and corners in a set of grayscale images [Smith95]. We use real images from Almaden's CattleCam [Almaden97] and attempt to detect cows in the landscape above San Jose. The application processes a set of 256 KB images and returns only the edges found in the data using a fixed 37 pixel mask. The intent is to model a class of image processing applications where only a particular set of features (e.g. the edges) in an image are important, rather than the entire image. This includes tracking, feature extraction,

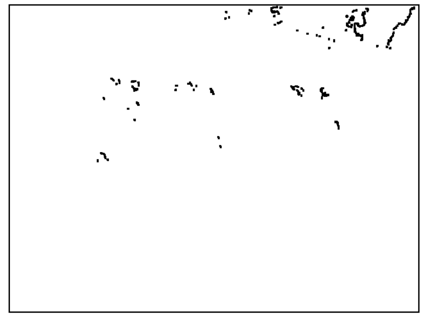


Figure 3: Edge detection in a scene outside the IBM Almaden Research Center. On the left is the raw image and on the right are the edges detected with a brightness threshold of 75.

and positioning applications that operate on only a small subset of the original images data. This application is significantly more computation-intensive than the comparisons and counting of the first two applications.

Using the Active Disks system, edge detection for each image is performed directly at the drives and only the edges are returned to the central server. A request for the raw image in Figure 3 returns only the data on the right, which can be represented much more compactly. The application reduces the amount of data transferred to the server by a large fraction (from 256 KB to 9 KB for this particular image). The state required on disk is the storage for a single image that is buffered and processed as a whole.

4.4 Multimedia - Image Registration

Our second image processing application performs the image registration portion of the processing of an MRI brain scan analysis [Welling98]. Image registration determines the set of parameters necessary to register (rotate and translate) an image with respect to a reference image in order to compensate for movement of the subject during the scanning. The application processes a set of 384 KB images and returns a set of registration parameters for each image. This application is the most computationally intensive of the ones studied. The algorithm performs a Fast Fourier Transform (FFT), determines the parameters in Fourier space and computes an inverse-FFT on the resulting parameters. In addition to this, the algorithm may require a variable amount of computation since it is solving an optimization problem using a variable number of iterations to converge to the correct parameters. Unlike the other applications, the per byte cost of this algorithm varies significantly with the data being processed. The average computation cost of each of the algorithms discussed in this section is shown in Table 2 in the next section.

For the Active Disks system, this application operates similarly to the edge detection. The reference image is provided to all the drives and the registration computation for each processed image is performed directly at the drives with only the final parameters (1.5 KB for each image) returned to the central server. The application reduces the amount of data transferred to the server by a large, fixed fraction. The state required at the disk is the storage for the reference image and the current image.

5 Prototype / Experiments

Our experimental testbed contains ten prototype Active Disks, each one a six-year-old DEC Alpha 3000/400 (133 MHz, 64 MB, Digital UNIX 3.2g) with two 2.0 GB Seagate ST52160 Medalist disks. For the server case, we use a single DEC AlphaStation 500/500 (500 MHz, 256 MB, Digital UNIX 3.2g) with four 4.5 GB Seagate ST34501W Cheetah disks on two Ultra-Wide SCSI busses (with more bandwidth than the server can use). All these machines are connected by an Ethernet switch and a 155 Mb/s OC-3 ATM switch.

Our experiments compare the performance of a single server machine with directly-attached SCSI disks against the same machine with network-attached Active Disks, each of which is a workstation with two directly-attached SCSI disks in our prototype. In the Active Disk experiments, as we increase the number of disks we increase the total amount of data processed, so the results we report are the throughputs (MB/s) for both systems. These results all show significant improvements with Active Disks and confirm the intuition provided by the model of Section 3.

5.1 Database - Nearest Neighbor Search

Figure 4 compares the performance of the single server system against a system with Active Disks as the number of disks is increased from 1 to 10. As predicted by our model, we see that for a small number of disks, the single server system performs better. The server processor is four times as powerful as a single Active Disk processor and can perform the computation at full disk rate. We see that the server system CPU saturates at 25.7 MB/s with two disks and performance does not improve as two additional disks are added, while the Active Disks system continues to scale linearly to 58 MB/s with 10 disks. Our prototype system was limited to 10 Active Disks by the amount of hardware we had available, and four traditional disks by the length limitations of Ultra SCSI, but if we extrapolate the data from the prototype to a larger system with 60

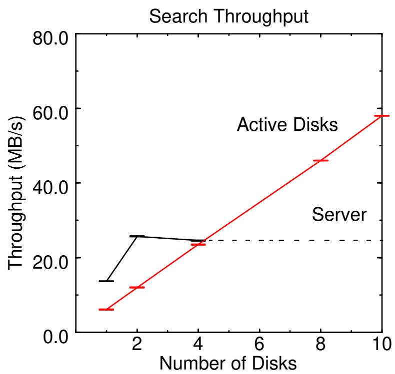


Figure 4a: The search application shows linear scaling with number of disks up to 58 MB/s, while the server system bottlenecks at 26 MB/s.

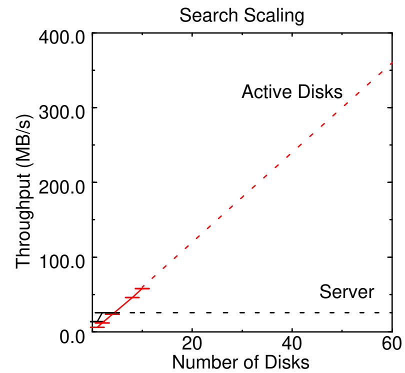


Figure 4b: Because of the high selectivity of this search, we would not expect the Active Disks system to saturate for at least a few hundred disks.

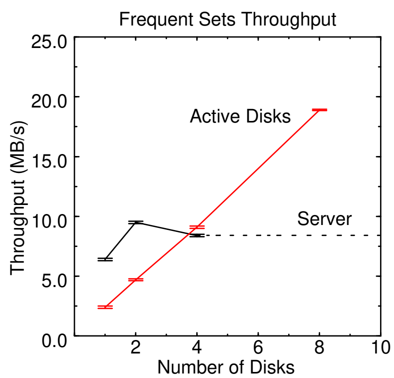


Figure 5a: The frequent sets application shows linear scaling to 18.9 MB/s with eight Active Disks, while the server system bottlenecks at 8.4 MB/s.

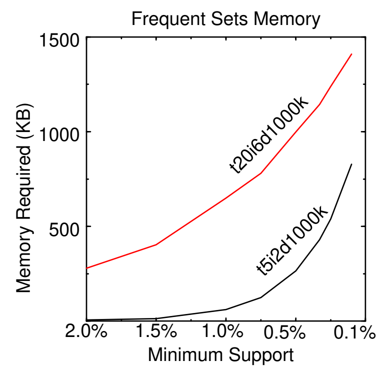


Figure 5b: The amount of memory necessary for the frequent sets application increases as the level of support required for a particular rule decreases. Very low support values may require multiple megabytes of memory at each Active Disk.

disks, the smallest of the systems in Table 1, we would expect throughput near the 360 MB/s that our model predicts for this configuration.

5.2 Data Mining - Frequent Sets

In Figure 5, we show the results for the first two passes of the frequent sets application (the *1-itemsets* and *2-itemsets*). We again see the crossover point at four drives, where the server system bottlenecks at 8.4 MB/s and performance no longer improves, while the Active Disks system continues to scale linearly to 18.9 MB/s. Figure 5b illustrates an important property of the frequent sets application that affects whether or not a particular analysis is appropriate for running on Active Disks. The chart shows the memory requirements across a range of input support values on two different databases. The lower a support value, the more itemsets are generated in successive phases of the algorithm and the larger the state that must be held on disk. We expect that the support will tend toward the higher values since it is difficult to deal with a large number of rules, and the lower the support, the less compelling the generated rules will be. For very low values of the support, though, the limited memory at Active Disk may become an issue. Modern disk drives today contain between 1 MB and 4 MB of cache memory, so we might expect 4 - 16 MB in the timeframe in which Active Disks could become available. This means that care must be taken in designing algorithms and in choosing when to take advantage of execution at the disks.

5.3 Multimedia

Figure 6 shows the results for the image processing applications. As we see in Table 2, the image processing applications require much more CPU time than search or

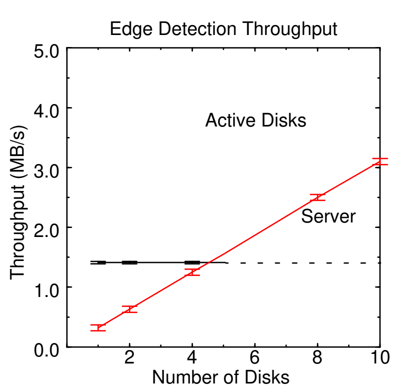


Figure 6a: The edge detection application shows linear scaling with number of disks while the server system bottlenecks at about 1.4 MB/s.

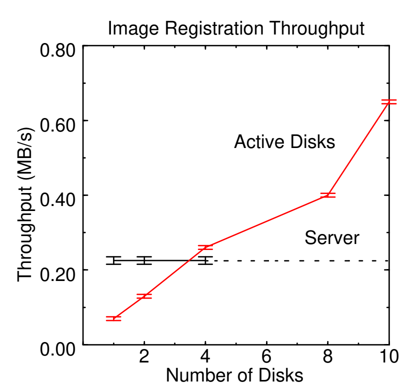
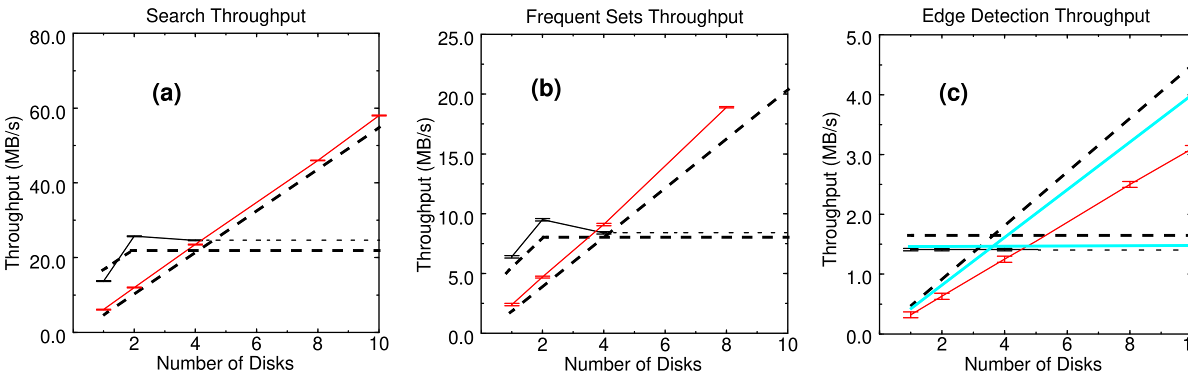


Figure 6b: The image registration application also scales linearly, but requires almost a factor of ten more CPU cycles, reducing throughput in both systems.



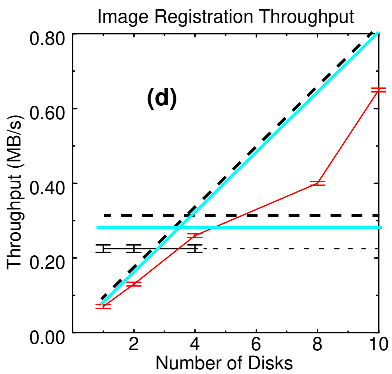


Figure 7: Validation of the analytical model against the prototype measurements. The values predicted by the model using the system and application parameters in Section 5.4 are superimposed as dashed lines on the measurements from the prototype systems. The differences are within 15% for search and frequent sets. Edge detection and image registration do not precisely fit the assumptions of our model. Both applications suffer additional, unmodelled I/O stall time because they read from the disks in image-sized chunks, rather than streaming in sequential accesses as search and frequent sets are able to do. This means that there is disk access time that cannot be overlapped with computation, reducing the throughput. There is serial overhead in all of the applications. This is largely amortized by the large size of the databases used in search and frequent sets, but shows up in the image processing applications that process less total data. The gray lines on the figures estimate the expected performance when these factors are accounted for. This estimated performance is now within 15%.

frequent sets do, leading to much lower throughputs on both systems. The edge detection bottlenecks the server CPU at 1.4 MB/s, while the Active Disk system scales to 3.2 MB/s with 10 disks. Image registration is the most CPU-intensive of the applications we have considered. It achieves only 225 KB/s on the server system, and scales to 650 KB/s with 10 Active Disks.

5.4 Model Validation

The graphs of Figure 4, 5, and 6 confirm the shape of the model in Section 3. To confirm the values, we need the specific parameters of this testbed. We have $\alpha_s = 133/500 = 1/3.8$ (estimated directly from the clock rates because the processors use the same basic chip, and the code is identical for both cases). Ideally, we would have $\alpha_d = \alpha_n = 1$ for our tests, but this was not possible in our testbed. Instead $r_d = 14\text{ MB/s}$, $r_d' = 7.5\text{ MB/s}$, $r_n = 60\text{ MB/s}$ and $r_n' = 10\text{ MB/s}$.

Estimating the applications' selectivity was a straightforward exercise of counting bytes and these are shown in Table 2. Estimating the number of cycles per byte was not so straightforward. We began by instrumenting the server implementation of each application to determine the total number of cycles spent for the entire computation when all code and data are locally cached, and dividing this by the total number of bytes processed. This ignores the cost of forming, issuing and completing the physical SCSI disk operations, measured in a previous study as 0.58 microseconds on a 150 MHz Alpha or 10.6 cycles per byte [Patterson95]. We add this to our "hot cache" numbers and report the resulting estimate of the cycles per byte required by each application in Table 2.

Figure 7 combines the results for all four applications and superimposes the predictions of the model based on these system and application parameters. The search and frequent sets applications show strong agreement between the model and the measurements. The largest error, a 14%

application	computation (cycles/byte)	memory (KB)	selectivity	parameter
Search	23.1	72	80,500	k=10
Frequent Sets	61.1	620	15,000	s=0.25%
Edge Detection	288	1776	110	t=75
Image Registration	1495	672	150	-

Table 2: Parameters of the applications presented in the text: computation time per byte of data, memory required at each Active Disk, and the selectivity factor in the network.

disagreement between the server model and implementation of the search may reflect an overestimate of the cycles per byte devoted to disk processing because the estimate is based on an older machine with a less aggressive superscalar processor. The other two applications, however, differ significantly from the model predictions. The problem with these applications is that they do not yet overlap all disk accesses with computation, as our model assumes. For example, the edge detection application reads 256 KB images as a single request and, since the operating system read-ahead is not deep enough, causes additional stall time as each image is fetched. Using asynchronous requests or more aggressive prefetching in the application should correct this inefficiency. An additional contributor to this error is the serial portion of the applications which affects the image processing applications more seriously since they process less total data than the other two. To estimate the performance of these applications if the overlapping were improved, we estimated the total stall time experienced by each application and subtracted it from the application run time.We report these "improved" prototype estimates as additional lines in Figure 7c and d. With this modification, our model predicts performance within 15% for all applications. Given our goal of using the model to develop intuition about the performance of Active Disks applications, these are strong results.

6 Discussion

The largest single benefit from using Active Disks, and the principle effect in our experiments, is the *parallelism* available in large storage systems. Although processing power on disk drives will always be less than on top-of-the-line server CPUs, there will very often be more aggregate CPU power in the disks than the server. Applications that can be partitioned to take advantage of this parallelism, and that can be "split" across the server and drive CPUs, have available a much higher total computational power than applications running only on the server.

The other large benefit of Active Disks is the ability to dramatically reduce interconnect bandwidth by *filtering* at the disks. In many systems in use today, interconnect bandwidth is at a premium compared to computational power, and is all too often a significant bottleneck. If an application is scanning large objects in order to select only specific records or fields or gather summary statistics, a large fraction of the data otherwise moved across the interconnect will simply be discarded, dramatically reducing the bottleneck.

These two advantages are the focus of this paper because they promise orders of magnitude potential improvements. In storage systems research, however, the most common application-specific optimizations are scheduling, batching and prefetching of disk operations [Bitton88, Ruemmler91]. Active Disks can be expected to execute these types of remote functions as well. In particular, we might expect Active Disks to participate as part of a disk-directed I/O model, where scatter/gather accesses are optimized using local information at the disks [Kotz94]. Or in prefetching systems where disks are provided with hints about future accesses [Patterson95].

A promising variant of these common optimizations is interconnect transfer *scheduling*. While network scheduling alone cannot be expected to yield benefits like we have seen in this paper, it can be an integral part of Active Disk computations for complex operations such as hash-join [Kitsuregawa83, DeWitt85] or variants of sort [Salzberg90, DeWitt91]. The key observation is that if data is going to move through the network after it is read from disk, it may be possible to send it to the "right" place under Active Disks control, reducing network traffic through scheduling at the disk, rather than sending it to the "wrong" place and then communicating among the processing nodes.

Consider a parallel sample sort algorithm running across a network of workstations similar to the setup of NowSort [Arpaci-Dusseau97]. The algorithm is composed of a sample phase and a sort phase [Blelloch98]. During the sample phase, a subset of the total data is read and a histogram is created allowing the key space to be divided into *n* buckets of roughly equal size. In the parallel server (cluster) version of this sort, the entire data set is then 1) read as is into the nodes from their local disks, 2) exchanged across the network according to the key space distribution, 3) sorted locally at each node, and 4) written back to the assigned disks.

Using network scheduling in an Active Disks system, we can remove the need for step 2 by having the drives perform the read and distribution operations at the same time. Instead of sending all data to a particular node, the drive is given the key ranges determined in the sample phase and responds to a request from client *n* with only the data "belonging" to client *n* as its portion of the key space. This means that data destined for a particular node will get to that node as soon as possible, and will never need to be exchanged among nodes. This reduces the number of transits of all the data across the network from three to two. In systems where the network is the bottleneck resource, this will improve overall performance of the algorithm by up to one-third.

7 Related Work

The basic idea of executing functions in processing elements directly attached to individual disks was explored extensively in the context of database machines such as CASSM [Su75], RAP [Ozkarahan75], and numerous others [DeWitt81]. These machines fell out of favor due to the limited performance of disks at the time and the complexity of building and programming special-purpose hardware that could only handle limited functions. Instead, database research has developed large-scale, shared-nothing database servers with commodity processing elements. It has recently been suggested that the logical extension is to perform all database processing inside programmable, "smart" system peripherals [Gray97].

Our work on Active Disks follows from our prior work on network-attached secure disks (NASD), in which we exploit computational power at storage for parallel and network file system functions, as well as traditional storage optimizations [Gibson97, Gibson98]. Our initial work discussed several classes of applications that can benefit from Active Disks - including filters, multimedia, batching, and storage management - and enumerated the challenges to providing an execution environment on commodity disk drives [Riedel97].

Work at Santa Barbara and Maryland has applied Active Disk ideas to a set of similar applications, including database select, external sort, datacubes, and image processing, using an extended-firmware model for next-generation SCSI disks [Acharya98]. Similarly, a group at Berkeley has independently estimated the benefit of Active (Intelligent in their terminology) Disks for improving the performance of large SMP systems running scan, hash-join, and sort operations in a database context [Keeton98].

8 Conclusions and Future Work

Commodity disks drives with an excess of computational power are visible on the horizon. Active Disks take advantage of this trend to provide an execution environment for application-specific code inside individual disk drives. This allows applications to take advantage of the parallelism in storage, greatly increasing the total computational power available to them, and circumventing the limited interconnect bandwidth, greatly increasing the apparent storage data rate.

We have demonstrated an important class of applications that will see significant gains (linear scaling in the number of devices added to the system) from the use of Active Disks. We have also provided an analytical model for estimating traditional server and Active Disk performance. Our prototype Active Disk system realizes speedups of more than 2 over a comparable single server system with up to 10 disks. Our system should easily scale to speedups of more than 10x in reasonably-sized systems similar to those already in use for large databases today.

Emerging applications such as data mining, multimedia feature extraction, and approximate searching involve the huge data sets, on the order of 100s of GB or TB, justifying large numbers of Active Disks. Many of these applications have small CPU and memory requirements and are attractive for execution across Active Disks.

There are a variety of areas to be explored before the benefits presented here can be put into practice. Providing a safe environment for application code inside the drive in order to both protect the integrity of data on the drive and ensure proper function in the presence of misbehaved application code is critical. The issue of resource management becomes considerably more complex as the computation becomes more distributed. Active Disks will need to make more complex scheduling decisions than disk drives do today, but they also open many new areas for optimization by exploiting the much richer interfaces they provide.

9 References

[Acharya98] Acharya, A., Uysal, M. and Saltz, J. "Active Disks" *Technical Report TRCS98-06*, March 1998.

[Agrawal95] Agrawal, R. and Srikant, R. ''Fast Algorithms for Mining Association Rules'' *VLDB*, September 1994.

[Agrawal96] Agrawal, R. and Schafer, J. "Parallel Mining of Association Rules" *IEEE Transactions on Knowledge and Data Engineering* 8,6. December 1996.

[Almaden97] Almaden CattleCam, IBM Almaden Research Center *www.almaden.ibm.com/almaden/cattle*, January 1998.

[Arpaci-Dusseau97] Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., Culler, D.E., Hellerstein, J.M. and Patterson, D.A. "High-Performance Sorting on Networks of Workstations" *ACM SIGMOD*, June 1997.

[Arya94] Arya, M., Cody, W., Faloutsos, C., Richardson, J. and Toga, A. "QBISM: Extending a DBMS to Support 3D Medical Images" *International Conference on Data Engineering*, February 1994.

[Barclay97] Barclay, T. "The TerraServer Spatial Database" *www.research.microsoft.com/terraserver*, November 1997.

[Berchtold96] Berchtold, S., Keim, D.A. and Kriegel, H. "The X-tree: An Index Structure for High-Dimensional Data" *VLDB*, 1996.

[Berchtold97] Berchtold, S., Boehm, C., Keim, D.A. and Kriegel, H. "A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space" *ACM PODS*, May 1997.

[Bershad95] Bershad, B.N., Savage, S., Pardyak, P., Sirer, E.G., Fiuczynski, M.E., Becker, D., Chambers, C. and Eggers, S. "Extensibility, Safety, and Performance in the SPIN Operating System" *SOSP*, December 1995.

[Bitton88] Bitton, D. and Gray, J. "Disk Shadowing" *VLDB*, 1988.

[Blelloch98] Blelloch, G.E., Leiserson, C.E., Maggs, B.M., Plaxton, C.G., Smith, S.J. and Zagha, M. "An Experimental Analysis of Parallel Sorting Algorithms" *Theory of Computing Systems* 31 (2), March 1998.

[Boral83] Boral, H. and DeWitt, D.J. "Database Machines: An Idea Whose Time Has Passed?" *International Workshop on Database Machines*, September 1983.

[Cao94] Cao, P., Lim, S.B., Venkataraman, S. and Wilkes, J. "The TickerTAIP Parallel RAID Architecture" *ACM Transactions on Computer Systems* 12 (3), August 1994.

[DeWitt81] DeWitt, D.J. and Hawthorn, P. "A Performance Evaluation of Database Machine Architectures" *VLDB*, September 1981.

[DeWitt85] DeWitt, D.J. and Gerber, R. "Multiprocessor Hash-

Based Join Algorithms" *VLDB*, August 1985.
[DeWitt91] DeWitt, D.J., Naughton, J.F. and Schneider, D.A. "Parallel Sorting on a Shared-Nothing Architecture using Probabilistic Splitting" *PDIS*, 1991.
[DeWitt92] DeWitt, D.J. and Gray, J. "Parallel Database Systems: The Future of High Performance Database Processing" *Communications of the ACM* 36 (6), June 1992.
[Drapeau94] Drapeau, A.L., Shirriff, K.W., Hartman, J.H., Miller, E.L., Seahan, S., Katz, R.H., Lutz, K., Patterson, D.A., Lee, E.K. and Gibson, G.A. "RAID-II: A High-Bandwidth Network File Server" *ISCA*, 1994.
[Faloutsos94] Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D. and Equitz, W. "Efficient and Effective Querying by Image Content" *Journal of Intelligent Information Systems* 3 (4), July 1994.
[Faloutsos96] Faloutsos, C. *Searching Multimedia Databases by Content*, Kluwer Academic Inc., 1996.
[Flickner95] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D. and Yanker, P. "Query by Image and Video Content: the QBIC System" *IEEE Computer*, September 1995.
[Gibson97] Gibson, G., Nagle, D., Amiri, K., Chang, F., Feinberg, E., Gobioff, H., Lee, C., Ozceri, B., Riedel, E., Rochberg, D. and Zelenka, J. "File Server Scaling with Network-Attached Secure Disks" *ACM SIGMETRICS*, June 1997.
[Gibson98] Gibson, G., Nagle, D., Amiri, K., Butler, J., Chang, F., Gobioff, H., Hardin, C., Riedel, E., Rochberg, D. and Zelenka, J. "A Cost-Effective, High-Bandwidth Storage Architecture" *Technical Report CMU-CS-98-115*, March 1998.
[Gosling96] Gosling, J., Joy, B. and Steele, G. *The Java Language Specification*. Addison-Wesley, 1996.
[Gray97] Gray, J. "What Happens When Processing, Storage, and Bandwidth are Free and Infinite?" Keynote Address, *IOPADS*, November 1997.
[Grochowski96] Grochowski, E.G. and Hoyt, R.F., "Future Trends in Hard Disk Drives" *IEEE Transactions on Magnetics* 32 (3), May 1996.
[Hsiao79] Hsiao, D.K. "DataBase Machines Are Coming, DataBase Machines Are Coming!" *IEEE Computer*, March 1979.
[Keeton98] Keeton, K., Patterson, D.A. and Hellerstein, J.M. "The Intelligent Disk (IDISK): A Revolutionary Approach to Database Computing Infrastructure" *White Paper*, University of California Berkeley, May 1998.
[Kitsuregawa83] Kitsuregawa, M., Tanaka, H. and Moto-Oka, T. "Application of Hash To Data Base Machine and Its Architecture" *New Generation Computing* 1, 1983.
[Kotz94] Kotz, D. "Disk-directed I/O for MIMD Multiprocessors" *OSDI*, November 1994.
[Lee96] Lee, E.K. and Thekkath, C.A., "Petal: Distributed Virtual Disks" *ASPLOS*, October 1996.
[Livny87] Livny, M., "Multi-disk management algorithms" *ACM SIGMETRICS*, May 1987.
[Necula96] Necula, G.C. and Lee, P. "Safe Kernel Extensions Without Run-Time Checking" *OSDI*, October 1996.
[Ozharahan75] Ozharahan, E.A., Schuster, S.A. and Smith, K.C. "RAP - Associative Processor for Database Management" *AFIPS Conference*, 1975.
[Patterson88] Patterson, D.A., Gibson, G. and Katz, R.H., "A Case for Redundant Arrays of Inexpensive Disks" *ACM SIGMOD*, June 1988.
[Patterson95] Patterson, R.H., Gibson, G., Ginting, E., Stodolsky, D. and Zelenka, J. "Informed Prefetching and Caching", *SOSP*, 1995.
[Quest97] Quest Project, IBM Almaden Research Center "Quest Data Mining Project" *www.almaden.ibm.com/cs/quest*, December 1997.
[Riedel97] Riedel, E. and Gibson, G. "Active Disks - Remote Execution for Network-Attached Storage" *Technical Report CMU-CS-97-198*, December 1997.
[Romer96] Romer, T.H., Lee, D., Voelker, G.M., Wolman, A., Wong, W.A., Baer, J., Bershad, B.N. and Levy, H.M. "The Structure and Performance of Interpreters" *ASPLOS,* October 1996.
[Ruemmler91] Ruemmler, C. and Wilkes, J., "Disk Shuffling" *HP Labs Technical Report HPL-CSP-91-30*, 1991
[Seagate97] Seagate Technology "Cheetah: Industry-Leading Performance for the Most Demanding Applications", *www.seagate.com*, 1997.
[Small95] Small, C. and Seltzer, M. "A Comparison of OS Extension Technologies" *USENIX Technical Conference*, January 1996.
[Smith79] Smith, D.C.P. and Smith, J.M. "Relational DataBase Machines" *IEEE Computer*, March 1979.
[Smith95] Smith, S.M. and Brady, J.M. "SUSAN - A New Approach to Low Level Image Processing" *Technical Report TR95SMS1c*, Oxford University, 1995.
[StorageTek94] Storage Technology Corporation, "Iceberg 9200 Storage System: Introduction" *STK Part Number 307406101*, 1994.
[Su75] Su, S.Y.W. and Lipvski, G.J. "CASSM: A Cellular System for Very Large Data Bases" *VLDB*, 1975.
[TPC98] Transaction Processing Performance Council "TPC Executive Summaries" *www.tpc.org*, February 1998.
[TriCore97] TriCore News Release "Siemens' New 32-bit Embedded Chip Architecture Enables Next Level of Performance in Real-Time Electronics Design" *www.tri-core.com*, September 1997.
[Turley96] Turley, J. "ARM Grabs Embedded Speed Lead" *Microprocessor Reports* 2 (10), February 1996.
[VanMeter96] Van Meter, R., Holtz, S. and Finn G., "Derived Virtual Devices: A Secure Distributed File System Mechanism" *Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, September 1996.
[Virage98] Virage "Media Management Solutions" *www.virage.com*, February 1998.
[Wactlar96] Wactlar, H.D., Kanade, T., Smith, M.A. and Stevens, S.M. "Intelligent Access to Digital Video: Informedia Project" *IEEE Computer*, May 1996.
[Wahbe93] Wahbe, R., Lucco, S., Anderson, T.E. and Graham, S.L. "Efficient Software-Based Fault Isolation" *SOSP*, December 1993.
[Welling98] Welling, J. "Fiasco: A Package for fMRI Analysis" *www.stat.cmu.edu/~fiasco*, January 1998.
[Wilkes95] Wilkes, J., Golding, R., Staelin, C. and Sullivan, T. "The HP AutoRAID hierarchical storage system" *SOSP*, December 1995.
[Yao85] Yao, A.C. and Yao, F.F. "A General Approach to D-Dimensional Geometric Queries" *ACM STOC*, May 1985.