

a quarterly bulletin of the
Computer Society
technical committee on

Database Engineering

CONTENTS

| | |
|--|----|
| Letter from the Editor-in-Chief | 1 |
| <i>W. Kim</i> | |
| Letter from the Editor | 2 |
| <i>D. Bitton</i> | |
| Optical Content Addressable Memories for Managing an Index to a Very Large Data/Knowledge Base <i>P.B. Berra, and S. Marcinkowski</i> | 4 |
| Optical Mass Storage Systems and Their Performance | 14 |
| <i>S. Christodoulakis, K. Elliott, D. Ford, K. Hatzilemonias, E. Ledoux, M. Leitch, R. Ng</i> | |
| The Impact of Disk Striping on Reliability | 26 |
| <i>H. Garcia-Molina, and K. Salem</i> | |
| A Project on High Performance I/O Subsystems | 40 |
| <i>R. Katz, J. Ousterhout, D. Patterson, and M. Stonebraker</i> | |
| Disk Performance and Access Patterns for Mixed Database Workloads | 48 |
| <i>C. Turbyfill</i> | |
| Calls for Papers | 55 |

SPECIAL ISSUE ON HIGH PERFORMANCE I/O SYSTEMS

Editor-in-Chief, Database Engineering

Dr. Won Kim
MCC
3500 West Balcones Center Drive
Austin, TX 78759
(512) 338-3439

Associate Editors, Database Engineering

Prof. Dina Bitton
Dept. of Electrical Engineering
and Computer Science
University of Illinois
Chicago, IL 60680
(312) 413-2296

Prof. Michael Carey
Computer Sciences Department
University of Wisconsin
Madison, WI 53706
(608) 262-2252

Prof. Roger King
Department of Computer Science
campus box 430
University of ColoradoDr.
Boulder, CO 80309
(303) 492-7398

Prof. Z. Meral Ozsoyoglu
Department of Computer Engineering and Science
Case Western Reserve University
Cleveland, Ohio 44106
(216) 368-2818

Dr. Sunil Sarin
Computer Corporation of America
4 Cambridge Center
Cambridge, MA 02142
(617) 492-8860

Chairperson, TC

Dr. Sushil Jajodia
Naval Research Lab.
Washington, D.C. 20375-5000
(202) 767-3596

Vice-Chairperson, TC

Prof. Krithivasan Ramamrithan
Dept. of Computer
and Information Science
University of Massachusetts
Amherst, Mass. 01003
(413) 545-0196

Treasurer, TC

Prof. Leszek Lilien
Dept. of Electrical Engineering
and Computer Science
University of Illinois
Chicago, IL 60680
(312) 996-0827

Secretary, TC

Richard L. Shuey
2338 Rosendale Rd.
Schenectady, NY 12309
(518) 374-5684

Database Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Database Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, database architecture, database machines, intelligent front ends, mass storage for very large databases, distributed database systems and techniques, database software design and implementation, database utilities, database security and related areas.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews, summaries, case studies, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Database Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in the Database Engineering Technical Committee is open to individuals who demonstrate willingness to actively participate in the various activities of the TC. A member of the IEEE Computer Society may join the TC as a full member. A non-member of the Computer Society may join as a participating member, with approval from at least one officer of the TC. Both full members and participating members of the TC are entitled to receive the quarterly bulletin of the TC free of charge, until further notice.

Letter from the Editor-in-Chief

It is my sad duty to inform you that Sushil Jajodia will be ending his term as Chairman of the Technical Committee on Data Engineering. Sushil has been very active, innovative, and effective as the TC Chair, and I have come to appreciate him as one of the best colleagues I have ever worked with.

During 1987, there have been a few changes to the editorial staff of Data Engineering. Yannis Vassiliou, C. Mohan and Haran Boral have resigned as Associate Editors. On behalf of the Technical Committee, I thank them for their services.

I am happy to announce that Z. Meral Ozsoyoglu, Dina Bitton and Roger King have agreed to join the staff as new Associate Editors. I am sure that these outstanding professionals will help make Data Engineering remain a great publication for database professionals. I look forward to working with them.

Roger King is organizing a special issue for June on semantics database systems; and Meral Ozsoyoglu will edit the September issue on non-first normal form relational theory. Richard Snodgrass will guest editor the December issue on temporal databases.



Won Kim
Austin, Texas
February, 1988

LETTER FROM THE EDITOR

This issue is devoted to high performance I/O systems. It examines current trends in mass-storage technology and new design ideas for high bandwidth, reliable I/O systems. At the 1987 Sigmod Conference, a panel on Technology Trends in Mass-Storage Systems projected magnetic versus optical disk trends and investigated their effect on database systems [Bitton 1987]. The five papers presented in this issue reflect address the problems identified in last year's panel, and provide an interesting cross-section of the solutions that the database research community is currently investigating.

Unlike processor speed, the I/O rate of mass-storage devices has not changed substantially in recent years. Magnetic disks remain the prevalent media for on-line secondary storage. Improvements in magnetic recording increase bit density and disk capacity, but mechanical delays have kept their random access time around 30 milliseconds. Optical disks provide very low cost per bit and high capacities, but have even higher access times, around 100 milliseconds. The main problem with using optical disks for secondary storage is that they are not rewritable, which disqualifies them for on-line transaction processing systems. While increasingly used for archival purposes and storage of multimedia documents, optical disks cannot currently be used for on-line database storage.

What will be the paradigm for mass-storage systems of the future? Advocates of optical storage predict the advent of rewritable disks and substantial improvement in access performance. For magnetic recording, the projection is by the end of the decade, density is likely to reach the range now identified with optical storage. Thus there is no consensus of whether optical and magnetic storage will become complementary or competitive. On the other hand, a clear trend that was acknowledged by last year's panel was the rapid and revolutionary change in the small disk industry. The personal computer market is a strong driving force in the development of high-density, low-cost magnetic disks and cartridges. It has made small, inexpensive disks widely available, and at the same time triggered innovations in disk controller technology and standardization of interfaces. This trend may make arrays of interleaved small disks a desirable alternative to single large disks [Kim 1986, Patterson 1987].

Two approaches in designing a high performance I/O system are illustrated by the papers in this issue. One is based on replacing or augmenting magnetic storage by optical storage, the other on coupling large numbers of off-the-shelf magnetic disks. The first two papers exemplify the first approach, although they consider two very different types of optical memory: an associative optical memory for caching index data (Berra and Marcinkowski), and an optical archival system (Christodoulakis et al). The next two papers exemplify the second approach. They both investigate techniques for coupling magnetic disks, as a way to obtain higher I/O bandwidths and enhance reliability.

Berra's and Marcinkowski's paper investigates Optical Content Addressable Memory (OCAM), based on an intriguing holographic technique. The authors contend that, because of their cost and performance characteristics, these memories can be used for storing and managing large indices in a Data/Knowledge Base. Index pages stored in OCAM can be read in parallel, at the speed of light, thus providing almost instantaneously all the matches to a query.

The paper by Christodoulakis, Elliot, Ford, Hadzilomonias, Ledoux, Leitch, and Ng describes work that is on-going at the University of Waterloo on the MINOS project, a multimedia database management system with an optical disk server. Because of the very large storage requirements of text, graphics, and the archival nature of the applications involved, Write Once Read Many times (WORM) disks are used. The authors have developed a performance model for WORM's, which accounts for optical disk characteristics, notably the ability to read a *span* of neighboring

tracks without moving the access mechanism.

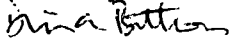
Garcia-Molina's and Salem's paper takes a new look at disk striping. It summarizes a performance study of striped file systems and addresses the issue of reliability. Pure striping schemes, where data is interleaved across a group of disks without any redundancy, are less reliable than their single disk counterpart in terms of their mean time between failure (MTBF). However, the authors contend that a more appropriate metric is the mean computation before failure (MCBF), which is lower for striped disks.

Katz, Ousterhout, Patterson, and Stonebraker describe a new project at Berkeley on high performance I/O systems. They contrast recent improvements in the speed of workstation CPU's, with the nearly constant speed of I/O devices and advocate the need for new I/O architectures based on arrays of small magnetic disks. These arrays should combine data redundancy and interleaving, in order to provide both high I/O rates and reliability. Furthermore, the arrays should be reconfigurable in order to match the requirements of different I/O domains: transaction processing, network file service, and supercomputer I/O.

In the last paper, Carolyn Turbyfill presents an interesting analysis of disk access patterns in a transaction processing system. The analysis contrasts sequential and random access requirements, which must be efficiently and concurrently supported in a balanced database workload. A close look at current disk limitations clearly demonstrates that a solution to this problem requires multiplexing disks and I/O paths.

Editing a special issue on this timely topic was a challenge. Of the many current projects on high performance I/O systems, only a few could be described here. I hope that the set of papers included in this issue will stimulate your interest in this important area.

Sincerely,



Dina Bitton

References

- [Bitton 1987] Bitton D., "Current Trends in Mass-Storage Technology", *Proceedings Sigmod 1987*.
- [Patterson 1987] Patterson, Gibson, and Katz, "A Case for Redundant Arrays of Inexpensive Disks, *Berkeley Tech. Report*, UCB/CSD 87/391.
- [Kim 1986] Kim M., "Synchronous Disk Interleaving", *IEEE Trans. on Computers*, C-35, No 11, Nov. 1986.

Optical Content Addressable Memories
for
Managing an Index to a Very Large Data/Knowledge Base

P. Bruce Berra and Slawomir J. Marcinkowski

Dep't of Electrical and Computer Engineering
Syracuse University
Syracuse, NY 13244-1240
U. S. A.

ABSTRACT

In this paper we present an optical approach for managing an index to a very large data/knowledge base. Specifically, our research is focused on optical content addressable memories (OCAM) based on holographic principles. OCAMs offer advantages over conventional CAMs in managing a large index, because OCAMs offer a considerably larger storage capacity (10^8 - 10^{10} bytes) and parallel output. With the large capacity of the OCAM, the potential exists for storing full index data to a very large data/knowledge base, which would otherwise be prohibitively costly in conventional electronic CAMs.

INTRODUCTION

One of the major problems in the management of very large data and knowledge bases (10^{12} - 10^{16} Bytes) is the time needed to access the desired data/knowledge. There exist memory hierarchies along with various indexing and search schemes which are designed to minimize the time it takes to access the desired data/knowledge. These include: hashing, B-trees, inverted lists, pointer systems, etc. All of these techniques have their strengths and weaknesses; so no one method is optimal for all situations. However, due to the fact that there is an increase in performance when using an index over a full sequential search of the entire data/knowledge base all data/knowledge base management systems have the facility to construct an index. While retrieval performance can be improved through the use of indexing it is not without its costs. The main costs are associated with the update of the index data when changes, additions, and deletions are required and in the management of the index data due to its large physical size. In fact, in some applications where considerable index data are required, the size of the indexes will be as large as or even larger than the data/knowledge being managed. Research over the years has focused on how to improve the access time without incurring too much overhead in terms of update and storage space required for indexing data. Content-addressable memories (CAM), a hardware technique, have attracted a lot of attention. The CAM is attractive to database applications, because data can be accessed based on content, and not by addressing. Accessing data by content offers considerable speed up, because the search is conducted in

parallel. Also, various operations can be performed in searching the data depending on the application. For instance, in certain computations such as text retrieval it is only necessary to locate entries which exactly match a given search argument. In other database applications, the equality search is not the only type of search performed. Frequently, there occurs a need to do a masked search involving only a subset of the search argument, or to locate entries which satisfy specified magnitude relations, for instance, having one or several of their attributes above, below, or between specified limits, or absolutely greatest or smallest. The CAM is well suited to the performance of these types of operations. In fact, CAMs can be used to perform match (equality), mismatch (not equal to), less than, less than or equal to, greater than, greater than or equal to, maximum, minimum, between limits, outside of limits, next higher, next lower, and various forms of add/subtract.

However, CAMs are not without their disadvantages. The chief constraint on CAMs is that they are quite expensive, and therefore only a small CAM can be included in any system. The problem of a small CAM is that for data/knowledge base applications the CAM is input output bound. The performance of the CAM is governed by the time it takes to load the data into the CAM, rather than the time to process CAM resident data. In fact, this disparity may be several orders of magnitude. It is for this reason that CAMs have not found widespread use in data/knowledge base systems.

The above comments apply to CAMs that are constructed from electronic components. However, CAMs can be constructed from optical devices, particularly holograms, and these are called optical content addressable memories (OCAM). They have the desirable search processing properties of electronic CAMs yet offer a considerably larger storage capacity (10^8 - 10^{10} bytes) and parallel output. Their main disadvantage is that they are read only. However, if the database under consideration is static and requires fast access then OCAMs offer a potentially cost effective solution.

In our research, we are considering OCAMs for the processing of index data to a very large data/knowledge base. With the large capacity of the OCAM, the potential exists for storing full index data to a very large data/knowledge base. We are taking a page oriented approach which serves to insulate us somewhat from changes.

In this paper we provide an overview of holography and some of its characteristics. We then discuss a particular OCAM developed by Sakaguchi 1970. We show how it might be used to implement an indexing scheme and discuss some of the issues that one must consider in managing the index to a very large data/knowledge base with OCAMs.

HOLOGRAPHY

Introduction

The term holography is derived from the Greek "holos" meaning "the whole or entirety" [Gabor 69]. A hologram records the whole of the wave information about the light wave at each point on a 2-dimensional surface. The wavefront can easily be reconstructed so that the 3-dimensional information about the wave can be retrieved at will.

In order to establish why holography is useful in storing information we draw an analogy between holography and photography. A photograph captures a light

image, and when it is developed we can see that image. Photography basically provides a method of recording the 2-dimensional irradiance distribution of an image. Generally speaking each "scene" consists of a large number of reflecting or radiating points of light. The waves from each of these elementary points all contribute to a complex wave, which is called the "object wave". This complex wave is transformed by the optical lens in such a way that it collapses into an image of the radiating object. It is this image that is recorded on the photographic emulsion (film).

Holography is quite different. With holography, one records the object wave itself and not the optically formed image of the object. This wave is recorded in such a way that a subsequent illumination of this record serves to reconstruct the original object wave. A visual observation of this reconstructed wavefront then yields a view of the object or scene which is practically indiscernible from the original. It is thus the recording of the object wave itself, rather than an image of the object, which constitutes the basic difference between conventional photography and holography sometimes referred to as "lensless photography" [Smith 69],[Caulfield 70].

Basics of Recording a Hologram

The basic technique of forming a hologram is shown in Figure 1. The coherent light beam coming from a laser is divided into two beams. One of the beams is used to illuminate the object and the other acts as a reference, and therefore are referred to as the *object beam* and *reference beam*, respectively.

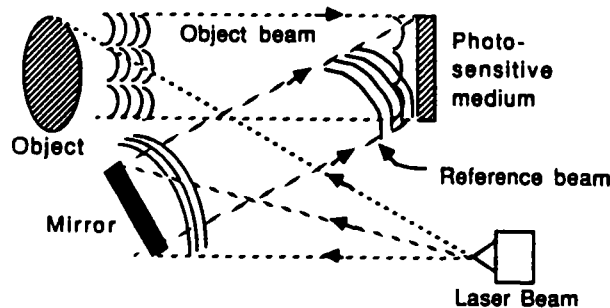


Fig. 1 Formation of a Hologram

The reference beam is directed so that it will intersect and overlap the object beam at a point where a photosensitive medium has been placed. The reference beam and the object beam, will form an interference pattern on the photosensitive medium. After the beams are removed the medium is processed and the interference pattern formed by the overlapping of the object and reference beams is called a *hologram* [Collier 79].

Basics of Reading a Hologram

A hologram is read by illuminating it with a beam having the same physical properties as the reference beam with which it was recorded. As the beam passes through the hologram it is diffracted into a recreation of the original object wave coming from the object.

An observer viewing a wave identical to the original object wave, perceives it to diverge from a virtual image of the object located precisely at the original object's

location. On the other hand if the hologram's backside is illuminated by a conjugate reference beam (a beam in which all the rays are exactly opposite to the original reference beam), then a real image is formed at the original object's position, as shown in Figure 2. Since the light converges to a real image, the image can be detected without the use of a lens by photodetectors. Hence, a hologram is a diffracting record of the interference of an object and reference beam.

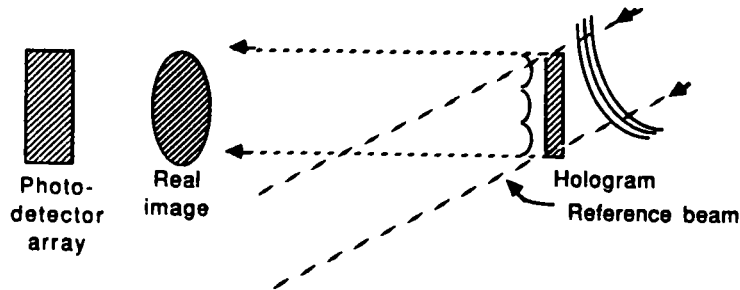


Fig. 2 Reading a Hologram

Page Holograms: 2-Dimensional Recording

A *page hologram* can be defined as a hologram made up of many nonoverlapping subholograms or pages [Rajchman 70]. A page is recorded through the use of a page composer, which is a device that stores data and converts it from electronic form to optical form, as indicated in Figure 3.

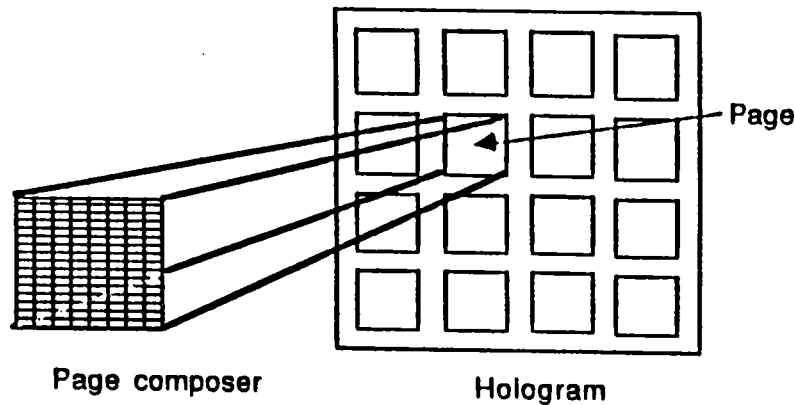


Fig. 3 Page Hologram with Page Composer

To record a page, data is first loaded from the computer to the page composer, then a small area of the emulsion is exposed by an aperture, and the information stored in the page composer is flashed onto the exposed area. To record another page the aperture is moved, new data is loaded into the page composer and the process is repeated. Depending on the medium used, information can be erased from a page, and the page can be rewritten with new data.

Volume Holograms: 3-Dimensional Recording

If the recording medium is thick, a hologram may be classified as a *volume hologram* in which many wave patterns can be superimposed. Volume holograms have some very useful and interesting features which are distinct from those of plane holograms. Volume holograms lend themselves very nicely to the storage of information since many holograms are superimposed in the same recording volume. Each hologram is discernible from the others due to the fact that each hologram is recorded with a unique reference beam angle.

To reconstruct (read) a particular hologram the reference beam must be deflected to within a narrow angular corridor about the Bragg angle at which that particular hologram was recorded. Beam deflection for this purpose is accomplished by an acousto-optical modulator, or otherwise known as a Bragg Cell, which is discussed in the next section.

As the number of holograms recorded in a volume increases, the angular corridor decreases [Gaylord 79]. As early as 1968 it was shown that it was possible to superimpose 1000 holograms in the same photographic emulsion [LaMacchia 68].

The recording of more and more holograms in a particular volume, introduces the problem that a hologram being written will affect the holograms already present in the volume. So this is one of the factors, that limits the number of holograms that can be stored in a volume.

This particular problem was largely solved by applying an external electrical field to the material in which the recording is taking place, Lithium Niobate. The application of the electrical field, greatly increased the material's writing sensitivity, while its erasure sensitivity did not change. Thus, as a new hologram is written into the volume, only a slight erasure of the other holograms occurs.

Acousto-Optic Deflector (Bragg Cell)

A Bragg Cell is an acousto-optical deflector which allows the redirection of a light beam passing through it, as shown in Figure 4.

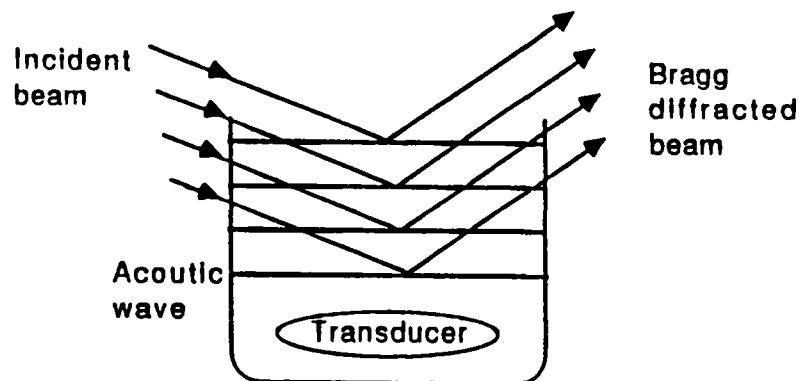


Fig. 4 Acousto-Optical Deflector

At the bottom of the deflector cell there is an acoustic transducer. This transducer introduces an acoustic wave into the elasto-optic medium of the cell. In this way a periodic strain is established in the medium. The cell is now interpreted to be a stack of reflecting layers spaced by the acoustic wave's wavelength. By turning on and off and varying the acoustic wave's wavelength, the Bragg Cell can conveniently and quickly deflect the light beam passing through it.

Usually for holographic applications there will be a deflector for each dimension that the beam must be deflected in. For example, Figure 5 is a xy deflector system. Generalizing, a cascade combination of m deflectors allows 2^m deflection angles [Gaylord 79].

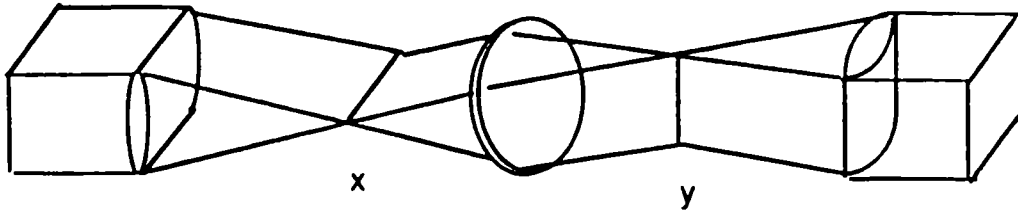


Fig. 5 Page Addressing Deflector System

Magneto-Optic Holograms

Most holograms mentioned previously are write once holograms. A category of holograms that offers possible read write capabilities is magneto-optical holograms. The only difference between these and the holograms we have already described is the recording process. The storage medium is a thin film of MnBi, a magnetic material, that is magnetized normally to the surface into which holographic patterns are recorded by Curie-point writing, and read out is by the Kerr or Faraday effects [Mezrich 70]. These holograms, which have considerable update capability, could be useful in indexing data/knowledge bases that are not static.

Computer Generated Holograms

Computer Generated Holograms (CGH) are important because a hologram can be made of an object which does not physically exist. Such a hologram can be constructed by calculating the ideal wavefront on the basis of diffraction theory. The field on the hologram surface is represented by a transparency produced by a computer driven plotter, laser beam, or electron beam on correspondingly diverse materials [Tricoles 87]. The possibility here exists that holograms can be recorded close to real time.

DATABASE INDEX USING OCAMs

Sakaguchi demonstrated a relatively straight forward extension of conventional holographic techniques in order to develop a one dimensional optical content addressable memory [Sakaguchi 70]. Later [Knight 74] refined the idea into a page oriented holographic associative memory. Sakaguchi used two object beams $A(x)$, $B(x)$ and one reference beam, instead of one object and one reference beam to record a thin hologram. The two beams $A(x)$ and $B(x)$, contain the bit information, where $A=(A_1, A_2, A_3 \dots A_n)$ and $B=(B_1, B_2, B_3 \dots B_m)$. The A information is represented by beam $A(x)$, which consists of $2n$ coherent parallel beams, $a_1, \bar{a}_1, a_2, \bar{a}_2, a_3, \bar{a}_3 \dots a_n, \bar{a}_n$, while the B information is represented by beam $B(x)$ consisting of $2m$ diverging beams $b_1, \bar{b}_1, b_2, \bar{b}_2, b_3, \bar{b}_3 \dots b_m, \bar{b}_m$. Each bit of A and B is represented by a true beam a_i and b_i and a complement beam \bar{a}_i and \bar{b}_i , where i represents the i th bit.

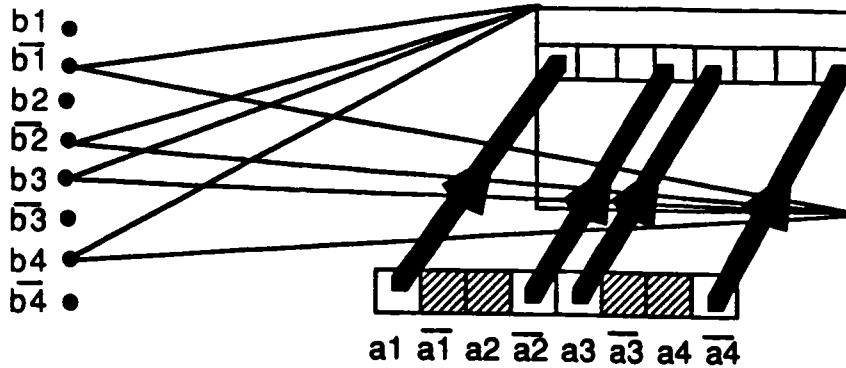


Fig. 6 Sakaguchi's Associative Holographic Memory

The light beams that make up $A(x)$ and $B(x)$ come from the true or complement bit places, according to whether the corresponding bits are in states "1" or "0". As shown in Figure 6, suppose that (A_1, A_2, A_3, A_4) is $(1, 0, 1, 0)$ and (B_1, B_2, B_3, B_4) is $(0, 0, 1, 1)$. Then, $A(x)$ is a set of coherent beams coming from $a_1, \bar{a}_2, a_3, \bar{a}_4$, and $B(x)$ is a set of diverging beams coming from $\bar{b}_1, \bar{b}_2, b_3, b_4$. In this way the hologram is recorded with A being recorded as a spot sequence $a_1, \bar{a}_2, a_3, \bar{a}_4$ over the entire page in a word space, while B is contained in every spot of $A(x)$.

The holograms are read by interrogating them with a search argument $C = (C_1, C_2, C_3, \dots, C_n)$, represented by beam $C(x)$. Beam $C(x)$ consists of $2n$ coherent light beams, $c_1, \bar{c}_1, c_2, \bar{c}_2, c_3, \bar{c}_3, \dots, c_n, \bar{c}_n$. A match is detected by the Exclusive Or principle; that is a match occurs if the i th interrogation signal bit and the i th interrogated bit are different. So, in actuality the beam which does the interrogation, $C'(x)$, is the complement of beam $C(x)$. The corresponding c_i or \bar{c}_i beams simultaneously illuminate the corresponding a_i or \bar{a}_i bits of all the words subjected to interrogation. Some of the C_i 's can be DON'T CARES, as such they do not participate in the interrogation so no light is emitted from the corresponding c_i or \bar{c}_i bit. To illustrate the point the previous values of A and B are used, where $A = (1, 0, 1, 0)$ and $B = (0, 0, 1, 1)$. If the search argument (C_1, C_2, C_3, C_4) is $(1, 0, dc, 0)$, where dc denotes a DON'T CARE bit, the corresponding interrogation beam $C'(x)$ is actually $(0, 1, dc, 1)$ that is, it consists of three parallel beams coming out of \bar{c}_1, c_2 and c_4 positions, with no light coming from c_3 or \bar{c}_3 , because the third bit is a don't care. We find that the interrogation signal $C = (1, 0, dc, 0)$ matches the interrogated signal $A = (1, 0, 1, 0)$, in reality it is the EXOR between $C' = (0, 1, dc, 1)$ and $A = (1, 0, 1, 0)$, which determines whether a match has occurred or not. The match is detected by a photodetector array; wherever the detector detects a null then a match has occurred. If all the bits for a word are detected as null then, the word matches the search argument.

Knight (1974) extended Sakaguchi's work by recording a page hologram consisting of 100 by 100 pages each containing 10^4 bits/page, for a total of 10^8 bits. Each page was very small; 1mm^2 . In recording the 10,000 pages the $A(x, y)$ and $B(x, y)$ beams were kept stationary while the reference beam was repositioned for each page. The page composer was loaded with new $A(x, y)$ and $B(x, y)$ data for each page.

What makes this suitable for implementing a database index is that each page has a code word, A , associated with it; and the data, B , written in a particular page can be identified by using the code word. In other words, the entire memory plane can be searched with a code word represented by beam $A(x, y)$ in parallel with one flash of light. When a match is detected by the photo-array, the data, represented by beam $B(x, y)$, corresponding to that page is read out. Next, an example will be presented that illustrates how this technique can be applied in implementing an

index scheme.

In this example we use an inverted list with indirect addressing as the index technique. This particular index was chosen over the more conventional inverted list index, due to the fact that whenever a reorganization of the database occurs (the physical addresses of the records change), all of the addresses in all the holograms would have to be changed. In conventional systems this is not an easy task, and in a medium such as holography it would be even more difficult. However, by using indirect addressing only the top level index would have to be changed.

Part of the example database is shown in Figure 7. The database is an employee database, and indexes are formed on SS#, Sex, Job, Dept# as shown in Figure 8.

| Database | | | | | | |
|-----------|--------|-----|-----|-------|--------|-----------|
| SS# | Name | Sex | Job | Dept# | Salary | Seniority |
| 012345678 | Mary | F | SQ | 120 | 1000 | 73 |
| 123456789 | Gary | M | FI | 110 | 2000 | 64 |
| 345678901 | John | M | EN | 110 | 1750 | 76 |
| 456789010 | Peter | M | JN | 130 | 900 | 84 |
| 567890123 | Eva | F | FI | 120 | 2400 | 79 |
| 678901234 | Teresa | F | EN | 120 | 1750 | 81 |
| 789012345 | Sophie | F | JN | 120 | 1000 | 83 |
| 890123456 | Mark | M | SQ | 130 | 1200 | 77 |
| 901234567 | Laura | F | FI | 110 | 2200 | 76 |
| 946801234 | Hector | M | EN | 130 | 2000 | 72 |
| 956801234 | Bruce | M | EN | 110 | 3000 | 70 |
| 968012345 | Luke | M | FI | 120 | 2400 | 74 |
| 979123456 | Dean | M | SQ | 130 | 1300 | 74 |
| 980123456 | Sue | F | EN | 120 | 2000 | 82 |
| 991245678 | Cindy | F | JN | 130 | 1100 | 85 |
| 999234567 | Anna | F | JN | 120 | 1200 | 83 |
| 999379135 | Sam | M | EN | 130 | 2300 | 80 |
| 999387913 | Lucy | F | FI | 110 | 2200 | 80 |
| 999581470 | Ken | M | SQ | 130 | 1300 | 72 |
| 999820048 | Henry | M | FI | 110 | 2400 | 70 |

Fig. 7 Database Excerpt

| A1 Index | | | A2 Index | | A3 Index | | A4 Index | |
|----------|-----------|---------|----------|---------|----------|---------|----------|---------|
| Serial# | SS# | Address | Sex | Serial# | Job | Serial# | Dept# | Serial# |
| 1 | 012345678 | 32 | M | 2 | SQ | 1 | 110 | 2 |
| 2 | 123456789 | 23 | | 3 | | 8 | | 3 |
| 3 | 345678901 | 14 | | 4 | | 13 | | 9 |
| 4 | 456789010 | 43 | | 8 | | 19 | | 11 |
| 5 | 567890123 | 16 | | 10 | FI | 2 | 120 | 18 |
| 6 | 678901234 | 10 | | 11 | | 5 | | 20 |
| 7 | 789012345 | 6 | | 12 | | 9 | | 1 |
| 8 | 890123456 | 19 | | 13 | | 12 | | 5 |
| 9 | 901234567 | 22 | | 17 | | 18 | | 6 |
| 10 | 946801234 | 13 | | 19 | | 20 | | 7 |
| 11 | 956801234 | 45 | 20 | EN | 3 | 130 | 12 | |
| 12 | 968012345 | 4 | F | | 1 | | 6 | 14 |
| 13 | 979123456 | 12 | | | 5 | | 10 | 16 |
| 14 | 980123456 | 21 | | | 6 | | 11 | |
| 15 | 991245678 | 30 | | 7 | 14 | 4 | | |
| 16 | 999234567 | 36 | | 9 | 17 | 8 | | |
| 17 | 999379135 | 40 | | 14 | JN | 4 | 10 | |
| 18 | 999387913 | 25 | | 15 | | 7 | 13 | |
| 19 | 999581470 | 18 | | 16 | | 15 | 15 | |
| 20 | 999820048 | 7 | | 18 | | 16 | 17 | |
| | | | | | | | | 19 |

Fig. 8 Indexes

The question now arises as to how we apply Knight's work to the above database index scheme. The pages represent the attributes we have indexed on. For example looking at the Department index we see that there are 3 different departments (110, 120, 130). With each department number we have associated a list of serial #'s. In recording the data we place the department number on the A(x,y) beam and the list of serial #'s associated with that particular department number on the B(x,y) beam. The representation is stored as shown in Figure 9.

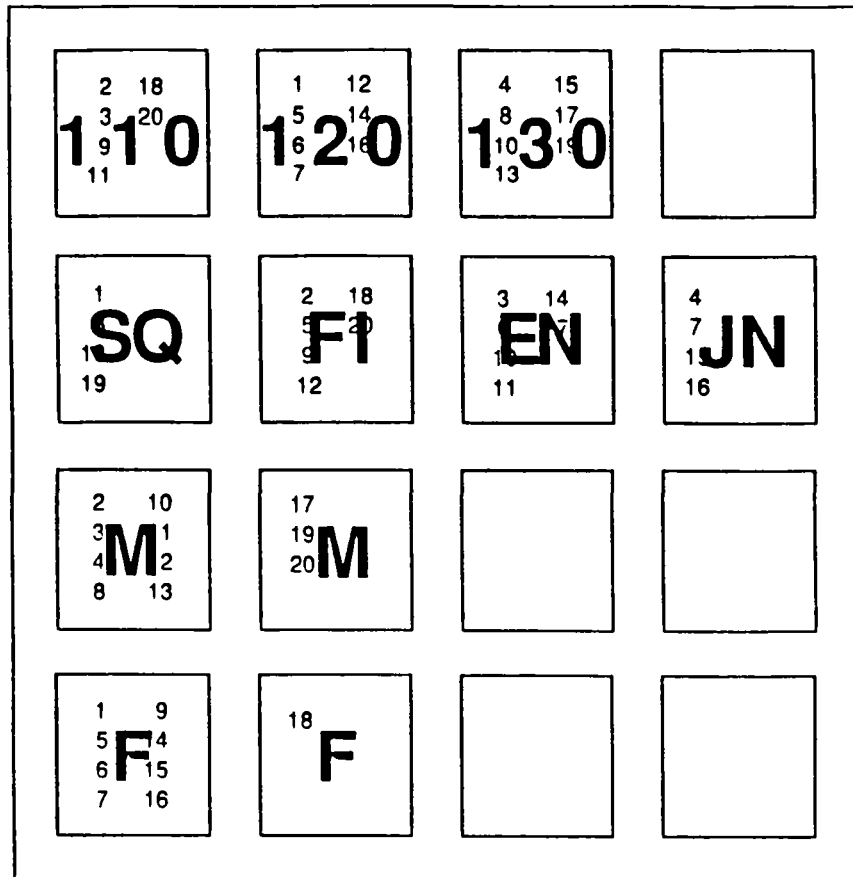


Fig. 9 OCAM Representation

In order to illustrate how queries are performed suppose information is needed on those persons working in department 120. In this case "120" is placed on beam C(x,y), the interrogation beam. The interrogation signal is flashed onto the memory plane, and the match is detected by the photo-detector array. The contents of the page are thus read in parallel and at the speed of light. The situation may exist that the data concerning department 120 may span several pages. In that case each of the pages must be read in turn to get the serial #'s. The serial #'s are now used to obtain the data records.

CONCLUSION

In this paper we have tried to illustrate how optical content addressable memories can be used in the management of indexes to very large data/knowledge bases. OCAMs have many attributes, large size, almost instantaneous access and parallel readout, that are important to solving the problems associated with managing an index for very large data/knowledge bases. However, it does take a relatively long time to form a hologram and they are difficult to change once formed. We are continuing our research into many of the issues concerning this marriage and will be reporting on our work from time to time.

REFERENCES

- [CAULFIELD70] H.J. Caulfield and Sun Lu, *The Applications of Holography*, Wiley-Interscience, Division of John Wiley & Sons, 1970
- [COLLIER71] R.J. Collier, C.B. Burckhardt, L.H. Lin, *Optical Holography*, Academic Press Inc., 1971
- [GABOR69] D. Gabor, "Associative Holographic Memories", *IBM Research Journal*, March 1969
- [GAYLORD79] Thomas K. Gaylord, *Handbook of Optical Holography*, Chapter on Applications, Academic Press Inc. 1979
- [KNIGHT74] Gordon R. Knight, "Page-Oriented Associative Holographic Memory", *Applied Optics*, Vol. 13, No. 4, April 1974
- [LAMACCHIA68] J.T. LaMacchia and D.L. White, "Coded Multiple Exposure Holograms", *Applied Optics*, Vol. 7, No. 1, January 1968.
- [MEZRICH70] R.S. Mezrich, "Magnetic Holography", *Applied Optics*, Vol. 9, No. 10, October 1970
- [RAJCHMAN70] J.A. Rajchman, "An Optical Read-Write Mass Memory", *Applied Optics*, Vol. 9, No. 10, October 1970
- [SAKAGUCHI70] M. Sakaguchi, N. Nishida, T. Nemoto, "A New Associative Memory System Utilizing Holography", *IEEE Transactions on Computers*, Vol. C-19, No. 12, December 1970
- [SMITH69] Howard M. Smith, *Principles of Holography*, Wiley-Interscience 1969
- [TRICOLES87] G. Tricoles, "Computer Generated Holograms: a historical overview", *Applied Optics*, Vol. 25, No. 20, October 1987

Optical Mass Storage Systems and their Performance

S. Christodoulakis, K. Elliott, D. Ford, K. Hatzilemonias, E. Ledoux, M. Leitch, R. Ng

Department of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1 Canada

1 Introduction

The MINOS project at the University of Waterloo aims at the development of multimedia data base management systems. These systems regularly have as a major resource, information which is not formatted (text, graphics, bitmaps, voice, animation, etc.)

Due to the very large storage requirements of these data types and the archival nature of the applications involved, optical discs are currently the most appropriate storage devices. As part of the MINOS project we study performance aspects of various optical disk based architectures, and we are designing and implementing a high performance server based on optical disk technology. In this paper we summarize our current research activities.

2 ODAS: An Optical Disk Based Archival System

WORM (Write Once Read Many times) storage is secondary memory in which registration of information is permanent (i.e. done once), but retrieval can be done an unlimited number of times. It differs from CD ROM, another, currently popular, type of permanent optical disk storage, in that it has greater storage capacity (atleast twice) and that information is written on the disk by the user, not by the manufacturer during pressing as is the case for CD ROM. It can also differ in the format of the recorded tracks, WORM disks can be either CAV (Constant Angular Velocity, concentric rings) or CLV (Constant Linear Velocity, single spiral track). CD ROM is only available in the CLV format.

The low cost per bit of stored information available when using WORM optical disks, their large storage capacity, cartridge removability, and long archival life are projected to be the dominant

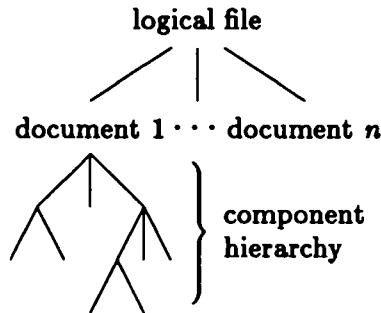


Figure 1: ODAS Logical File Organization

storage technology for many archival applications, examples of which can be found in [11,2,12]. Presently, existing data management systems either do not support WORM storage devices or do not meet the demanding functionality and performance requirements of a multimedia management system such as MINOS [3,8].

ODAS is a system for archival of structured documents. WORM disk cartridges are used for archival, and magnetic disk is used for intermediate document buffering. Once archived, a document may not be modified in place, but new versions of the document may be created with controlled physical data duplication. Facilities are provided to customize physical data placement based on anticipated access patterns to improve retrieval performance.

A document is a graph or hierarchy of components, which may be shared physically among documents. A component consists of user data, uninterpreted by ODAS. Every archived component has an identifier which is unique among all optical disks. A logical file is a collection of logically related archived documents (see Figure 1). No further organization is imposed on documents within a logical file.

A logical file may occupy many optical disk cartridges, and logical files may share cartridges. In ODAS, cartridges are self-contained so that they can be removed from one installation and transported to another. The archival system provides functionality for archival, retrieval, deletion of documents. Additionally, index maintenance packages may be interfaced with ODAS to implement

content addressability.

The design of ODAS accommodates the peculiarities of the WORM storage and exploits its properties in order to provide a high performance system. The span access capability of optical disks increases the importance of physically clustering data which will be frequently accessed together. Furthermore, if more than one optical disk drive is available, concurrent retrieval can enhance efficiency.

A declarative, high-level script language can be used by application designers to specify rules for data clustering based on properties of document components. Using this facility, information about known access patterns can be utilized to improve retrieval performance for specific applications [10].

Since the retrieval performance of magnetic disks is generally better than that of optical disks, the use of magnetic disk space for temporary buffering of incoming documents improves retrieval speed for documents which tend to be retrieved often soon after they are archived. Periodic flushing of document buffers in batches onto optical disk is done when buffers become full, or during system idle times (to alleviate contention and minimize seeks).

A first version of ODAS was completed at the University of Waterloo [9]. The software consists of approximately 50,000 lines of C source code, developed on Sun 3 workstations (also ported to a VAX 8650). The current implementation of ODAS includes a hardware driver for an Alcatel-Thomson Gigadisc WORM drive [1]. Additionally, two WORM disk simulators of varying complexity and properties have been implemented. Simulated disk drives may be intermixed transparently with real hardware.

3 The Distributed Testbed

The *Distributed Testbed* is a client-server based distributed system. The system features a high performance optical disk based multimedia document server and is intended to support a distributed version of MINOS. More importantly, it provides a testing ground to explore the specific requirements of supporting optical disks, large multimedia documents, and interactive users.

The testbed distributes the basic functionality of *ODAS*, an optical disk based document filing

system, across a set of workstations interconnected by a local area network. The testbed consists of two major software subsystems, the *Storage and Retrieval Subsystem* and the *Client Subsystem* (Figure 2). Each subsystem consists of an administrator process and several specialized worker processes [13]. Within a subsystem, the administrator is typically concerned with overall scheduling while the workers perform device and communication management.

The *Storage and Retrieval Subsystem* is the testbed's document server. The server receives requests from clients and replies to them. To retrieve a document, a client sends a retrieval request specifying a document to the server, the request is processed and the document is then returned to the client. To archive a document, the client sends an archival request accompanied by the document. The server batches the document on magnetic disk and the client is free to proceed. The decision to actually archive the document on the optical disk is at the discretion of the server.

The server has two specialized worker processes to manage the optical and magnetic disk storage. The optical disk storage manager interacts with *ODAS* to perform all optical disk requests. The magnetic disk storage manager provides a set of magnetic disk storage caches. The magnetic disk is available for the batching of documents to be archived. It is also available for the prefetching of documents to be retrieved.

The *Client Subsystem* manages user interaction with the testbed server. Each user in the system corresponds to a single instance of the client subsystem, with the potential for multiple clients per workstation. Client workstations may be optionally equipped with magnetic disk storage. A user interacts with the *Client Subsystem* through a set of exported operations through which requests are generated while architecture and communication details are obscured.

Communication within the testbed has severe performance and design implications and may be broken down into communication within subsystems and communication between subsystems. Within a subsystem, a set of reliable interprocess communication primitives is used. They are based upon the blocking send-receive-reply model and are used to relay all information between the processes.

Communication between subsystems may be divided into control messages and data messages.

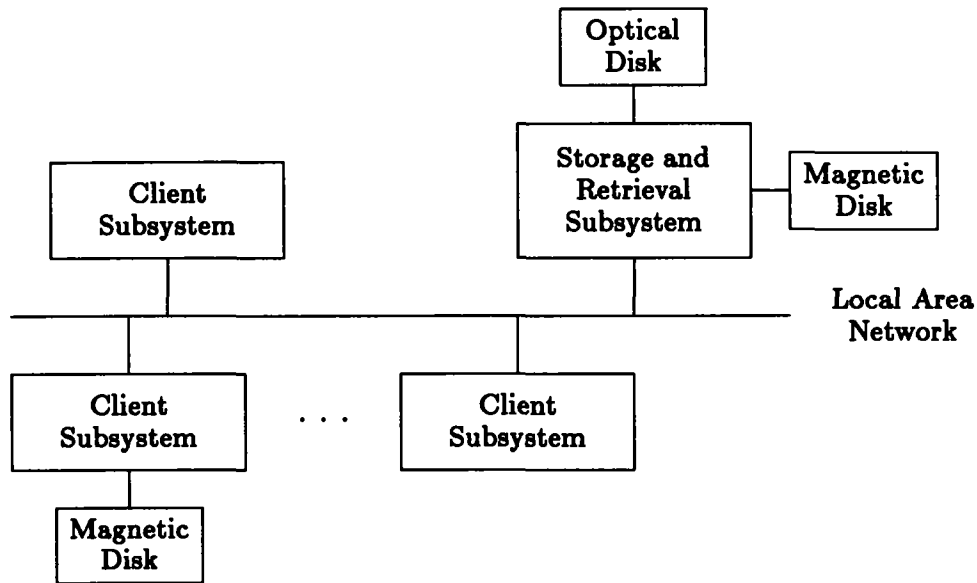


Figure 2: The Distributed Testbed System Architecture

Messages containing control information such as user requests and high level acknowledgements are communicated between subsystems using the reliable interprocess communication primitives. All control information sent between subsystems passes through a set of transporter processes. Each subsystem has a receive transporter and a reply transporter. A receive transporter receives a message sent to the subsystem and relays it to the administrator. The reply transporter receives a reply from the administrator and relays it to a specified destination subsystem. Through transporters, an administrator may communicate with other subsystems without blocking.

Multimedia data messages move between subsystems using an end-to-end window based communication protocol[14]. This approach eliminates the overhead associated with the interprocess communication primitives and also bypasses the intermediary transporter processes. This optimized communication path is critical to system performance due to the large amounts of data that can be associated with multimedia documents. This path is made possible by setting up an end-to-end connection between subsystems, analogous to a high-bandwidth bus.

Although interaction is required between the server and its clients, a significant effort has been made to abstract the data structures and algorithms of the testbed. This so-called testbed approach supports experimentation within the system. An abstraction may be altered or replaced without

affecting other areas of the testbed.

A major area of experimentation is the scheduling problems associated with implementing an optical disk based multimedia document server. To this end, optical disk requests flow through a hierarchy of three schedulers. The long term scheduler receives client requests and enters them into the intermediate term scheduler. The intermediate term scheduler maps a client request to a set of individual page requests on a particular device to be entered into a short term schedule. The short term schedule consists of the current set of optical disk operations to be performed.

The testbed approach also supports the embedding of instrumentation for the purpose of performance monitoring. Through performance monitoring, scheduling policies and other design decisions may be evaluated. The current implementation provides a response time monitoring facility providing a metric for overall system performance.

An enhancement, is the development of a query processing facility based upon signature techniques. This requires the introduction of a *Query Evaluation Subsystem* that searches signature files retrieved from the server. The user will be returned identifiers for documents meeting conditions specified in the query. Document miniatures may also be returned to enable the user to quickly browse through a set of documents and select a set of documents according to some criteria. Query processing further implies the support of user interrupts to terminate queries.

The first version of the *Distributed Testbed* has been completed at the University of Waterloo. The system involves a set of SUN 3 workstations communicating over an Ethernet local area network. The document server runs on a workstation equipped with an Alcatel-Thomson WORM optical disk drive and Fujitsu magnetic disks.

The multimedia document management subsystem of MINOS is going to be ported as an application on top of the system. The functionality of the testbed is expected to grow over time as greater performance and user requirements are imposed.

4 Optical Disk Simulators and Emulators

Presently, both WORM and CD ROM optical disks are not rewritable storage media, and in the case of CD ROMs, the mastering process is lengthy and expensive.

To ensure optimal retrieval performance in a certain application environment, careful planning is required during the application design stages, regarding the placement of data on the optical disk and thereafter, the scheduling of queries.

There are two methods to aid the process of examining performance implications of alternative data placement and query scheduling algorithms. The first, employs analytical tools to produce exact results and at times can be difficult and/or prove expensive. In such cases, approximations may provide more efficient but sometimes, due to the necessary simplifications involved of the model under examination, unreliable results. In these situations, the only means for reliable experimentation is simulation.

The **Optical Disk Simulator (ODS)** package [7] runs on SUN workstations, and uses the storage capacity of one gigabyte of magnetic disk to provide full scale simulation of various WORM and CD ROM disk architectures, as well as simulation of expensive storage devices such as single drive and multidrive jukeboxes of WORMs and CD ROMs.

Simulated optical disk drive architectures include one of two types of reading servomechanisms. Those having an objective lens only, at the end of the laser beam path, and those that also have an adjustable reflecting mirror.

During sector retrievals, as dictated by document retrieval queries under various query scheduling algorithms, estimated delays include jukebox operation costs, long and short seek costs, track address identification costs, rotational delays and transfer costs. Furthermore, simulation of multidrive jukeboxes allows parallelism during reading, resulting in reduced transfer times.

The ODS system has been extensively tested and is presently validated against the Hitachi CDR-1503S CD ROM drive and the Alcatel-Thomson GIGADISC WORM drive.

To date, the ODS system has been used for experimentation related to the retrieval performance of multimedia documents with different storage requirements, for experimentation with various

query scheduling algorithms, for investigation of the performance characteristics of each of the two reading servomechanisms, for investigation of the various implications the span size of the access mechanism has on performance and for deriving important implications based on the property differences of WORM and CD ROM optical disks. The ODS system has also been used to validate exact and approximate analytical results in the above areas.

Optical Disk Emulator System (ODES) emulates the placement and collation of actual or randomly generated multimedia documents on simulated WORM and CD ROM optical disk and drive architectures, and enforces the time delays involved (jukebox operation costs, seek costs, address identification costs, rotational delay costs, transfer costs) when retrieval queries from different clients in a distributed environment are posed on the data.

The ODES system incorporates a number of software facilities, developed by members of the Minos group. The client management and data prefetching facilities of the Minos Distributed Testbed, the various placement and retrieval facilities of the WORM Optical Disk Based Document Filing System (ODAS) and the query scheduling, CD ROM and WORM drive, disk and jukebox simulation and retrieval cost prediction facilities of the Optical Disk Simulator package (ODS).

5 Analysis

Optical disk technology takes two basic forms, that of *Constant Angular Velocity* (CAV) disks of which WORM type disks are an example, and *Constant Linear Velocity* (CLV) disks of which CD ROM is an example. CAV disks rotate at a constant rate but have a variable storage density, while CLV disks rotate at a variable rate and have a constant storage density. The two forms embody trade-offs of speed and storage. CAV disks have faster access times but typically lower storage capacities than their counterpart CLV disks.

Some of the work performed at the University of Waterloo has been to develop basic models and analytic tools for studying the retrieval performance of both types of optical disks [6]. These tools are expected to prove useful for data base design decisions as well as for performance predictions of various file organizations. They may also be of value in the development of query optimization

schemes.

The retrieval performance of optical disks is generally determined by the time required to move the access mechanism (the seek time). A typical delay for a CAV WORM drive ranges from 100msec to 300msec, compared to that of a magnetic disk of 30msec. The issue is complicated slightly however, by the fact that some optical disk drives are capable of reading from more than one track without moving the access mechanism (and incurring the seek delay). The access mechanisms of such optical disk drives are equipped with an adjustable mirror that allows slight deflections of the beam of laser light used to read the data over a small set of tracks immediately beneath the access mechanism. This set of tracks is called a *span* and the number in the set is called the *span size*. Typical values of the span size are 10, 20 and 40 tracks. A span size of 40 tracks corresponds roughly to about 1 megabyte of storage on a CAV WORM disk. This capability can be likened to cylinders on magnetic disk packs, except that spans on an optical disk drive can be overlapping.

Our analysis concentrates upon determining the expected number of *spans* required to access a given number of objects for drives with or without a deflecting mirror in the access mechanism.

5.1 Models

The models used in the analysis of both types of optical disks are similar. The distinction between the two is made when considering the impact of track capacities on retrieval performance. The track capacity of a CAV type disk is constant. Because of the spiral recording scheme used, the capacity of tracks on a CLV type disk varies depending upon their position on the disk surface, tracks at the outer edge hold more data than those at the inner edge.

An optical disk is a device composed of T ordered tracks, an access mechanism and a viewing mechanism. A track is represented by its sequence number i within the tracks of the device, $i = 1, 2, \dots, T$ (to avoid discussion about boundary conditions we assume that track numbers are extended above T and below 1). Each track is composed of a number of sectors (or blocks). The access mechanism can be positioned at any track. When the access mechanism is positioned at a certain track i , the device can read data which completely exist within Q consecutive tracks (track i is one of them). In order to do that, the viewing mechanism *focuses* to a particular track with

qualifying data (within the Q tracks). We call the Q consecutive tracks a *span* and this capability of optical disks, *span access capability*. An anchor point a of a span, is the smallest track number within a span. The largest track number within this span is $a + Q - 1$. The anchor point of a span completely defines the tracks of the span. A span can therefore be described by its anchor point.

A number N of objects (or records) may qualify in a query. N is called the object selectivity (or record selectivity) of the query. In the case of optical disks the number of times that the access mechanism has to be moved for accessing the data which qualifies in a query is called the *span selectivity* of the query. Therefore, a first approximation of the cost of evaluating a query is given by the span selectivity of the query.

When the access mechanism is moved a seek cost and a rotation delay cost are incurred as in the case of magnetic disks. The seek cost depends on the distance travelled by the access mechanism. Transferring a track of data from the device involves a track transfer cost as is also the case in magnetic disks. When more than one track within a given span has to be transferred to main memory, the access mechanism does not have to be moved because of the deflecting mirror. However, there is a small additional delay (of the order of one millisecond) involved for focusing the viewing mechanism on each additional track (within a span) that has to be read. We call this delay a viewing cost and will ignore it for the remaining part of this paper. This model reduces to a model of magnetic disks when the number of tracks in a span is one and the track size is equal to the cylinder size.

A spiral scheme is generally used on CLV optical disks for the storage of data rather than the concentric rings found on CAV disks. The spiral consists of one long physical track of data recordings that begins at a distance from the centre of the disk called the *principal radius*, and continues until close to the outer edge of the disk where it terminates.

A track for our purposes starts from the intersection of the spiral with the radial line that begins at the centre of the disk and passes through the very beginning of the physical spiral. The track ends at the next intersection with this radial line. Since sectors are of fixed length, the radial line may intersect the body of a sector. By convention, the first sector of a track is the first whole

sector encountered after the intersection of the radial line and the track.

5.2 Analytic Results

We have succeeded in deriving results for predicting the retrieval performance of both types of disks. We have produced both exact and approximate analyses for a variety of object (record) sizes and configurations.

For CAV type optical disks, we derived estimates of the number of spans (movements of the access mechanism) required to retrieve small and large objects. For small objects, the two cases of records being allowed and not allowed to cross sector boundaries were examined. For large objects such as bit maps and digitized audio, which could require several hundred kilobytes or even megabytes of storage, we have produced results for their retrieval performance for the case where only large objects exist in a file and also for the case where small objects may exist.

For CLV type optical disks, similar exact and approximate results were derived. It was shown that for the case of small objects, that the corresponding CAV results were an excellent approximation. Such was not the case for larger objects however, but we were able to derive two results, one based upon the expected distribution of objects among the variable capacity tracks and another based upon multiple CAV solutions [6].

5.3 File Placement

We also examined the implications of the variable track capacities found on CLV disks, on the placement of files on the disk surface (i.e. into sets of different capacity tracks). We found that files which are expected to experience the greatest frequency of random access should be placed closer to the outer edge (larger capacity tracks) of the disk than those files that are expected to experience fewer random accesses.

References

- [1] *GC 1001 OEM Manual*. ATG Gigadisc, April 1987.
- [2] A. Bell. Critical issues in high density magnetic and optical storage. In *Proc. of the SPIE*, January 1983.

- [3] S. Christodoulakis. Issues in the architecture of a document archiver using optical disk technology. In *Proc. of ACM SIGMOD '85*, pages 34–50, May 1985.
- [4] S. Christodoulakis. Query processing in optical disk based multimedia information systems. *IEEE Database Engineering*, October 1987.
- [5] S. Christodoulakis and C. Faloutsos. Design considerations for an optical-disk based multimedia object server. *IEEE Computer*, December 1986.
- [6] S. Christodoulakis and D. Ford. Performance analysis and fundamental performance trade-offs for CLV optical disks. December 1988. Technical Report.
- [7] S. Christodoulakis and K. Hatzilemonias. Optical Disk Simulation package (ODS). *Technical Report, University of Waterloo*, November 1987.
- [8] S. Christodoulakis, F. Ho, and M. Theodoridou. The multimedia object presentation manager of MINOS: a symmetric approach. In *Proc. of ACM SIGMOD '86*, pages 295–310, May 1986.
- [9] S. Christodoulakis, E. Ledoux, and R. Ng. *A WORM Optical Disk Based Document Filing System*. Technical Report, University of Waterloo, December 1987.
- [10] S. Christodoulakis and T. Velissaropoulos. Architectural aspects of MINOS: a distributed multimedia information system. *IEEE Office Knowledge Engineering*, 1(1):18–27, February 1987.
- [11] G. Copeland. What if mass storage were free? *IEEE Computer*, 27–35, July 1982.
- [12] L. Fujitani. Laser optical disk: the coming revolution in on-line storage. *CACM*, 27(6):546–554, June 1984.
- [13] W.M. Gentleman. Message passing between sequential processes: the reply primitive and the administrator concept. *Software - Practice and Experience*, 11, 1981.
- [14] J.H. Saltzer, D.P. Reed, and D.D. Clark. End-to-end arguments in system design. *ACM Trans. on Computer Systems*, November 1984.

THE IMPACT OF DISK STRIPING ON RELIABILITY

Hector Garcia-Molina

Kenneth Salem

Department of Computer Science
Princeton University
Princeton, N.J. 08544

ABSTRACT

A group of disks is striped if each data block is multiplexed across all the disks. Since each sub-block is in a different device, input and output can proceed in parallel, achieving greater bandwidth. One possible drawback of striping is decreased reliability: if a file is striped across n disks, and one or more disk failures can make the file unavailable. In this paper we study this problem and show that in some cases it may not be significant. For those cases where it is a problem, we discuss how it can be effectively addressed with data redundancy.

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and by the Office of Naval Research under Contracts Nos. N00014-85-C-0456 and N00014-85-K-0465, and by the National Science Foundation under Cooperative Agreement No. DCR-8420948. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

January 18, 1988

THE IMPACT OF DISK STRIPING ON RELIABILITY

Hector Garcia-Molina
Kenneth Salem

Department of Computer Science
Princeton University
Princeton, N.J. 08544

1. Introduction

A group of disk units is *striped* if each data block is stored, not on one of the disks, but across all of the disks. That is, if there are n disks, each block is split into n sub-blocks and a sub-block is placed on each disk.

The goal of striping is to improve IO bandwidth. Ideally, if a block is spread across n disks, then it can be read (or written) in parallel, cutting access time by a factor of $1/n$. Unfortunately, this beneficial effect may be counterbalanced by a higher CPU overhead and larger rotational and arm movement delays. Another possible drawback is lowered reliability: if a file is split across several disk drives, then the probability that one of the units, and hence the entire file, is unavailable may be higher.

In this paper we focus on the reliability issue, studying whether striping does reduce reliability or not, and discussing several options for improving striped reliability. However, before discussing reliability (Section 3) it is important to understand the performance of striping, as it will play a role in the reliability evaluation. Hence, in Section 2 we very briefly summarize some performance results that appeared in [SALE86].

It is important to note that striping is by no means a new idea. Although we have been unable to find references in the published literature, it appears that some early versions of Unix gave users the option of striping their files [HONE84]. Striping has also been proposed for multiprocessors and database machines [SIEG79, BORA83]. We also suspect that some type of striping is used in many high performance commercial applications, for instance, in sorting packages. On some parallel-readout disk models, striping has been done for many years within the disk drive itself. These units have multiple read/write heads mounted on a single arm, and parallel data transfers to and from the heads are possible. However, as disk densities increase, it becomes more difficult to align the different heads simultaneously during I/O. This makes parallel-readout disks much more expensive and limited to relatively small

number of striping channels. The alternative, striping from separate disk drives, is becoming more effective, and is thus the only one we consider in this paper.

Even though striping is not new, there has been renewed interest in it in the context of very high performance “supercomputers” that must move huge amounts of data. As a matter of fact, several supercomputer vendors like Cray[†] and Convex now provide support for striped disks [MASO84, CONV86], and report significant performance improvements [WALL86]. M. Y. Kim et al [KIM87b] report on the use of striping for very large fast fourier transforms.

At the same time, with rapidly increasing memory densities, there has recently been interest in memory resident database systems [GRAY83, DEWI84, GARC84]. Here again, striping has been suggested as a means to improve bandwidth to disks. For example, striping can reduce the time it takes to load the database into memory after a crash, or it can reduce delays in writing the log.

2. Striping Performance

There are a number of ways to study the performance of disk striping. The types of disks which are to be striped and the characteristics of the target applications have major effects on striping’s effectiveness. Our model concentrates on striping off-the-shelf, unsynchronized disks as part of a batch-type, back-end system. Other models are certainly possible and important. For example, M. Y. Kim considers the striping of synchronized disks [KIM84, KIM86] and unsynchronized disks [KIM87a] in a multi-programmed system. Livny et al [LIVN85] present another model for multi-programmed systems. Models such as ours have the virtue of simplicity; they allow us to understand striping by viewing it in a less complicated environment. In addition, a number of systems in common use today, such as supercomputer systems, are used as back-end systems for large batch jobs. Such systems can benefit tremendously from better IO performance, so our model is an important one.

The model is quite detailed [SALE86], so it is not possible to fully present it here. Suffice it to say that it includes the CPU overhead of initiating IO operations (proportional to the number of disks involved); the seek, latency and rotational times; the location of the blocks on the disk; and the amount of main memory available for buffering. There are also several enhancements or variations to striping that can be considered. For example, one option is to start reading a block off disk at any point where it is found (as opposed to waiting

[†] As far as we know, the Cray system was the first to use the term “striping” that we have adopted here.

for the beginning to come under the read head). If a block is as large as a track, then the latency delay is eliminated with this enhancement. Another option is to locate the sub-blocks for a striped block at the same position on their corresponding disks. This can eliminate the variability of the different seeks involved in fetching a block.

For the results we present here we do not consider any of the enhancements. To obtain these results we have set the parameters to model DEC's RA81 disk drives [DEC82]. The CPU takes $10 + n100$ microseconds to initiate an IO operation, where n is the number of disks that are involved.

The effectiveness of striping for a particular task is the percentage of speedup obtainable by striping the task's data set across n disks rather than storing it on a single disk. This is measured by comparing the task completion time in the striped system to the task completion time in the other. For example, consider a situation in which a task can be completed by a single-disk system in 2.5 seconds, while the same task is finished in 1.8 seconds by a system with a 3-disk striped group. We compute the effectiveness of the striped group by:

$$effectiveness = \left[(2.5 - 1.8)/2.5 \right] 100 = 28\%$$

Under this definition, the maximum possible speedup is 100%. Negative effectiveness indicates that the task completion time for the striped group is greater than that for the single-disk system.

Let us first consider a simple task like reading a single block off disk. Figure 1 shows the effectiveness of striping for this task with block sizes ranging from 1K to 256K bytes. This figure illustrates the basic behavior of striping: As the file is striped over more disks (n grows), the transfer time of a block decreases and effectiveness improves. However, as n increases, the CPU overhead increases. At the same time, it takes longer to reconstruct the block in memory because the variability of each sub-block seek is compounded. The net effect is that effectiveness tapers off surprisingly rapidly. At least for the parameters we have selected, it does not pay off to stripe more than approximately four ways ($n = 4$).

It is also clear that striping is more effective for larger block sizes, although significant savings can be obtained for block sizes as small as 1K bytes. That large block sizes provide an ideal environment for striping is to be expected since transfer time is exactly the component of the task completion time that striping works to reduce.

To fully evaluate striping one must look at an entire application, not just a single block transfer. Consider for instance balanced 2-way merge sort [KNUT73]. Assume that we begin with an unsorted database on an n -disk striped group. The task is to sort the database and

store the sorted version on the same striped disks.

Suppose that there are 1.5 million records each of size 100 bytes. Also, say that sorting a k -record buffer takes $50k\log(k)$ microseconds. Similarly, to merge two k -record buffers into one of twice the length takes $20k$ microseconds. The file block size is not fixed; we use the one that produces best performance for each scenario.

Figure 2 shows the effectiveness of striping on the merge-sort task, for various main memory buffer sizes. (The algorithm uses two buffers of the indicated size.) Increasing the buffer size allows the use of larger disk blocks. As was seen in previous tasks, larger block sizes provide a better environment for striping and result in higher effectiveness. Although it is not shown in the figure, the results are not very sensitive to the processing speed assumptions we made for sorting and merging.

Finally, recall that the results we have shown are for “current” hardware. However, note that processors are becoming faster, memories larger, but disk speeds are not changing very much. At the same time, data requirements are growing in many applications. All of these factors make striping more effective, so it is interesting to briefly consider a future scenario for striping. Consider for instance a file processing task where the records in a file must be read and processed in memory. Suppose that the file size is 2 gigabytes, the main memory buffer is half a megabyte, and the processing time is one nanosecond per byte (This is a fast processor. Disk parameters are unchanged.) Figure 3 presents the striping effectiveness for this task. (The curve begins at $n = 5$ since we need five of our disks to hold the 2 gigabyte file.) For comparison, we also present the graph for a “current scenario,” a 200 megabyte file with a 32 K buffer, and a one microsecond processing time per byte. In the new scenario, striped disks outperform non-striped disks by a factor of 3 to 1. The trend is clear: striping promises to become much more effective in the near future.

3. Striping Reliability

As mentioned in the introduction, there is a potential problem with disk striping: loss of a single disk can make an entire striped group unavailable. In this section we will study this problem, arguing that (1) it is important to distinguish between problems caused by multiple disks and those caused by *striping* multiple disks, (2) there are different measures of reliability and the impact of striping can be quite different, and (3) in those cases where failures are a problem, reasonable steps can be taken to alleviate them.

Many applications require more than a single disk for their data to satisfy either space or access requirements. If an application task requires access to data stored on several disks, the

failure of any of the disks will halt the task, *whether those disks are striped are not*. Increasing the number of disks increases the likelihood of a disk failure, however those disks are organized. We do not mean to imply that such higher failure rates are not a problem, in fact we shall describe a means of error correction for collections of disks later in this section. However, unless data can be *partitioned* in such a way that not all tasks need to access all of the disks, the distinction between striped and independent disks is not relevant to the issue of failures and failure recovery.

With this in mind, imagine an application whose data set can be partitioned so that, if the disks are independent rather than striped, different jobs access different disks. In the case of a disk failure, those jobs which do not access the failed disk can run to completion in the independent case, while all jobs would be halted if the disks were striped. Thus it would seem that the independently-organized disks have greater *availability* than striped disks.

This aspect of reliability, roughly the fraction of jobs that are runnable when submitted, is a very useful one for some types of applications. For example, in a transaction processing system it is important that the system be up as much of the time as possible since downtime translates directly into lost transactions, lost customers and lost revenue.

In other situations, other reliability/performance measures may be more appropriate [BEAU78, CAST80]. These other measures take into account, not just the availability, but the ability of the system to perform useful work when it is up. For example, the computation reliability is the probability that at a given time the system is operational and can perform a task of duration T . One can also compute the mean computation before failure (MCBF) which is the expected amount of computation before a failure. These types of metrics are more appropriate for studying striping because they take into account the fact that tasks may finish faster and be vulnerable to failures over shorter periods.

To illustrate, let us consider a simple example. (We have no justification for the numbers we are about to present other than that they are reasonable and they illustrate our point.) Suppose that we want to execute a task that takes T seconds and uses a file during this time. (Say the file is discarded after the task.) Assume that with probability $p = 0.99$, a single disk unit will not fail during the T seconds. If the file is not striped and fits on a single disk, the task will successfully complete with probability $P = p = 0.99$. This is the computation reliability defined in [BEAU78]. If the file is striped over 4 disks, then $P = p^4 = 0.96$ (assuming failure independence). So striping appears to have lower computation reliability.

However, a more detailed analysis shows that this may not be so. With striped disks, the task completes faster, say in $T/2$ seconds. (See figure 2, $n = 4$, for example). The probability

that a disk does not fail in this interval is approximately $p' = 0.995$ (the probability that it fails is roughly half of the previous amount). Hence, the striped task will be successful with probability $P = (p')^4 = 0.98$.

The computation reliability is still lower than for the non-striped case. However, we have ignored the CPU. Processors fail more often than disks, so it is reasonable to say that the CPU does not fail in T seconds with probability $q = 0.97$, in $T/2$ seconds with probability $q' = 1 - (1-q)/2 = 0.985$. Now, the non-striped task is successful with probability $P = qp = 0.960$, the striped one with probability $P = q'(p')^4 = 0.965$. Of course, this is just a simple example (e.g., q could be larger or smaller), but it does illustrate that striping does not *necessarily* lower reliability.

As the number of disks in a striped group increases, the reliability measures deteriorate. However, as noted in the previous section, most of the performance improvement that can be gained from striping can be achieved with fairly small striped groups. Thus, it may not be necessary to worry about the reliability of large groups since they are ineffective from a performance viewpoint anyway. In summary, for the small striping groups of interest, it is possible to argue that *in some cases* reliability may be better, or at least not much worse than with no striping.

There are of course, other cases where striping can cause a significant loss in reliability. In these cases, redundant information can be used to decrease the failure rate of a collection of disks. In the following we will speak of striped disk groups, however the technique that we will discuss can be applied to groups of independent disks as well. With striped disks, though, the addition of redundant hardware and error detection and correction capabilities meshes more smoothly with the normal access procedure for the disks.

The basic idea that we will present is to store redundant information that will permit a group of striped disks to tolerate the failure of one (or more) of its members. For example, suppose we want to cope with the failure of a single disk of the striped group. We can use an extra disk with parity bits. For each stored block, there will be a corresponding sub-block in the extra disk. The first bit in the sub-block will be the parity bit for the set of bits that constitute the first bit of the other sub-blocks. The second bit will be the parity for the second set of bits, and so on. This is illustrated in the diagram below, for 4-way striping:

| Disk 1 | Disk2 | Disk3 | Disk4 | Parity Disk |
|----------|----------|----------|----------|--|
| a_1 | b_1 | c_1 | d_1 | $a_1 \oplus b_1 \oplus c_1 \oplus d_1$ |
| a_2 | b_2 | c_2 | d_2 | $a_2 \oplus b_2 \oplus c_2 \oplus d_2$ |
| a_3 | b_3 | c_3 | d_3 | $a_3 \oplus b_3 \oplus c_3 \oplus d_3$ |
| a_4 | b_4 | c_4 | d_4 | $a_4 \oplus b_4 \oplus c_4 \oplus d_4$ |
| \vdots | \vdots | \vdots | \vdots | \vdots |

When a failure occurs, the error detection codes internal to each disk will be used to identify the unit and sub-block that has been lost. Then the parity sub-block can be used to reconstruct the missing sub-block. If having delays in the correction phase is tolerable, then the parity block does not have to be read until an error is detected. To correct more than a single failure, this example can be generalized using well known erasure correcting codes [BERL68]. Approaches similar to this one are discussed in [KIM85, PARK86, PATT87].

This strategy obviously has a cost: an extra disk is required, $1/n$ extra storage is used, and writes are more expensive (the parity sub-blocks have to be computed and written). However, not only have we improved reliability, we have probably surpassed the no-striping case. In our earlier example, assuming the task takes the full T seconds to complete in all cases, the task completes with probability $p = 0.990$ with no striping. With redundant striping, the task completes as long as 4 out of 5 disks do not fail, i.e., with probability $p^5 + 5(1-p)p^4 = 0.999$.

It is interesting to note that as n , the number of striped disks, increases, the redundant storage and the reliability decrease. With $p = 0.99$ and $n = 15$, only 6% extra storage is needed and the reliability is 0.990, the same as that of the non-striped case. Beyond $n = 15$ (an unlikely case as we pointed out earlier), we would require a 2 or more error correction strategy to make reliability higher.

If an extra disk is not available for storing the parity information, it could also be stored on the striped disks. The following diagram illustrates this for $n = 4$ [†]:

[†] In an actual implementation, parity bits would be stored at the end of each disk block instead of immediately following their associated data. They are shown immediately following their data in the diagram to make the error correction concept clear.

| Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| a_1 | b_1 | c_1 | d_1 |
| a_2 | b_2 | c_2 | d_2 |
| a_3 | b_3 | c_3 | d_3 |
| $b_3 \oplus c_3 \oplus d_3$ | $c_2 \oplus d_2 \oplus a_3$ | $d_1 \oplus a_2 \oplus b_2$ | $a_1 \oplus b_1 \oplus c_1$ |
| \vdots | \vdots | \vdots | \vdots |

The parity bit for a group of $n-1$ data bits is stored on the remaining disk. This way, the parity bit is always available in the case of a failure. (The strategy can also be generalized for more than 1 error detection).

This strategy is slightly more expensive in terms of redundant storage (it is $1/(n-1)$ as opposed to $1/n$) and write overhead (writes of parity information tie up the disks where the data is stored). However, there is no need for an extra disk unit, and the reliability is improved even more. In our running example, with this strategy the task will complete with probability $p^4 + 4(1-p)p^3 = 0.9994$. The extra disk strategy had reliability 0.9990 and the no-striping strategy 0.9900.

4. Conclusions

In summary, striping can be evaluated along two dimensions: performance and reliability. Along the performance dimension, it is clear that striping can improve efficiency, especially when large files and fast processors are involved.

The conclusions are not as clear along the reliability dimension. However, we have argued that for many applications, striped disks may actually provide greater reliability than unstriped disks. If striping can produce a significant reduction in the completion time for a task, this shorter “failure window” may counterbalance the negative aspect of striping disks reliably: that the failure of a single disk can make the entire group unavailable. Comparing the reliability of striped and unstriped disk groups also depends on the application programs that will use them. Large (greater than the size of a disk), monolithic data sets will not be any more reliable on unstriped disks than striped ones; any disk failure makes the data unavailable.

If reliable striping is a problem for an application, reliability can be boosted through the use of error correcting codes stored across the disks as well as within each disk. Extra disks can be used to store the error correcting information, or it can be stored on the same disks that hold the data. Reliability can be brought to whatever level is desired by increasing the

amount of error correcting information that is stored. As pointed out in [PATT87], the mean time to failure of modern disks is measured in tens of thousands of hours.[†] This means that for small striped groups (less than 10 disks), the mean time to failure of the group can easily be made greater than the expected useful life of the technology [PATT87].

Bibliography

- [Beau78] Beaudry, M.D., "Performance Related Reliability Measures for Computing Systems," *IEEE Transactions on Computers*, Vol. C-27, No. 6, June 1978, pp. 540-547.
- [BERL68] Berlekamp, E. "*Algebraic Coding Theory*," McGraw-Hill, 1968, pp. 229-231.
- [BORA83] Boral, H., DeWitt, D.J., "Database Machines: An Idea Whose Time Has Passed? A Critique of the Future of Database Machines," *Database Machines*, Leilich, H.-O., Missikoff, M., ed., Springer-Verlag, 1983, pp. 166-187.
- [CAST80] Castillo, X. and Siewiorek, D.P., "A Performance-Reliability Model for Computing Systems," *Proceedings Tenth International Symposium on Fault-Tolerant Computing*, Kyoto, Japan, October 1980, pp. 187-192.
- [CONV86] "*Convex System Managers Guide*," Doc. #710-000330-000, Rev. 3.0, Convex Computer Corp, 1986.
- [DEC82] "*RA81 Disk Drive User Guide*," Digital Equipment Corporation, 1982.
- [DEWI84] Dewitt, D.J., Katz, R.H., Olken, F., Shapiro, L.D., Stonebraker, M.R., Wood, D., "Implementation Techniques for Main Memory Database Systems," *SIGMOD Record*, Vol. 14, #2, 1984.
- [GARC84] Garcia-Molina, H., Lipton, R.J., Valdes, J., "A Massive Memory Machine," *IEEE Transactions on Computers*, Vol. C33, No. 5, May 1984, pp. 391-399.
- [GRAY83] Gray, J., "What Difficulties are Left in Implementing Database Systems," Invited Talk at *SIGMOD Conference*, San Jose, CA., May 1983.
- [HONE84] Honeyman, P., personal communication.
- [KIM84] Kim, M.Y., "Parallel Operation of Magnetic Disk Storage Devices: Synchronized Disk Interleaving," *Proc. of the Fourth International Workshop on Database Machines*, March, 1985, pp. 299-329.

Of course, these numbers refer to failures of the mechanical mechanism of the disk drive. Other factors, such as software errors and operator interference, reduce the numbers significantly.

- [KIM85] Kim, M.Y., "Error-Correcting Codes for Interleaved Disks with Minimal Redundancy," IBM Technical Report RC 11185 (#50403), May 1985.
- [KIM86] Kim, M.Y., "Synchronized Disk Interleaving," *IEEE Transactions on Computers*, Vol. C-35, No. 11, November 1986, pp. 978-988.
- [KIM87a] Kim, M.Y. and Tantawi, A. N., "Asynchronous Disk Interleaving," IBM Technical Report RC 12497 (#56190), January 1987.
- [KIM87b] Kim, M.Y., Nigam, A., Paul, G., and Flynn, R.J., "Disk Interleaving and Very Large Fast Fourier Transforms," *International Journal of Supercomputer Applications*, Vol. 1, No. 3, Fall 1987, pp. 75-96.
- [KNUT73] Knuth,D.E., "*The Art of Computer Programming*," Vol. 3, Addison-Wesley, 1973, pp.361-371,471-479.
- [LIVN85] Livny, M., Khoshafian, S., and Boral, H., "Multi-Disk Management Algorithms," MCC Technical Report No. DB-146-85, 1985. (Abstract appeared in *Proceedings ACM IEEE International Workshop on High Performance Transaction Systems*, Asilomar, California, September 1985.
- [MASO84] Mason, D., Cray Research Inc., personal communication.
- [PARK86] Park, A. and Balasubramanian, K., "Providing Fault Tolerance in Parallel Secondary Storage Systems," Technical Report CS-TR-057-86, Computer Science Department, Princeton University, November 1986.
- [PATT87] Patterson, D.A., Gibson, G., and Katz, R., "A Case for Redundant Arrays of Inexpensive Disks (RAID)," Technical Report UCB/CSD 87/391, Computer Science Division (EECS), University of California Berkeley, December 1987.
- [SALE86] Salem, K., and Garcia-Molina, H., "Disk Striping," *Proceedings Second Data Engineering Conference*, 1986.
- [SIEG79] Siegel,H.J.,Kemmerer,F.,Washburn,M.,"Parallel Memory System for a Partitionable SIMD/MIMD Machine," *Proc. 1979 International Conference on Parallel Processing*, pp. 212-221.
- [WALL86] Wallach, S., Vice-President, Technology, Convex Computer Corp., personal communication.

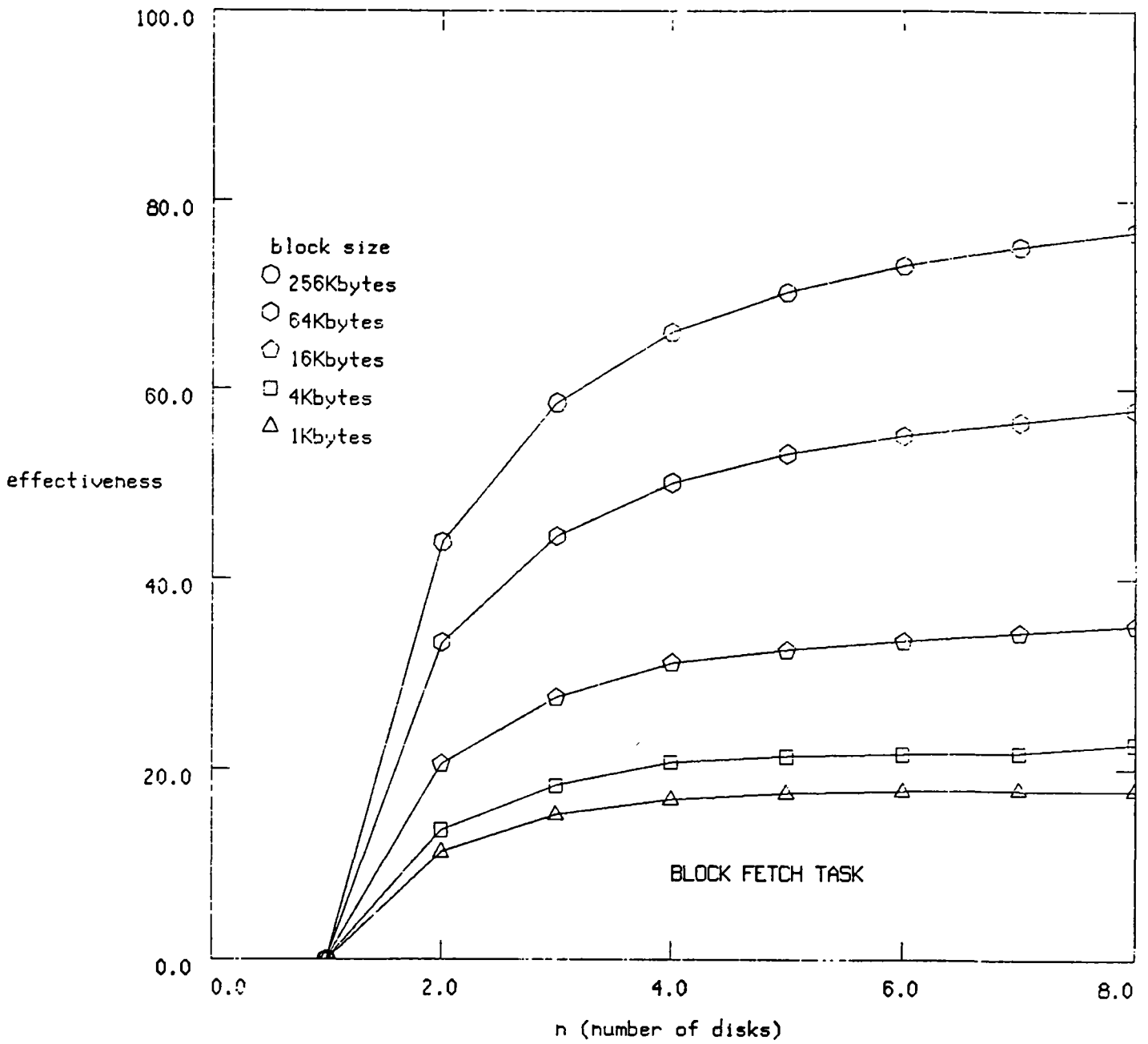


FIGURE 1

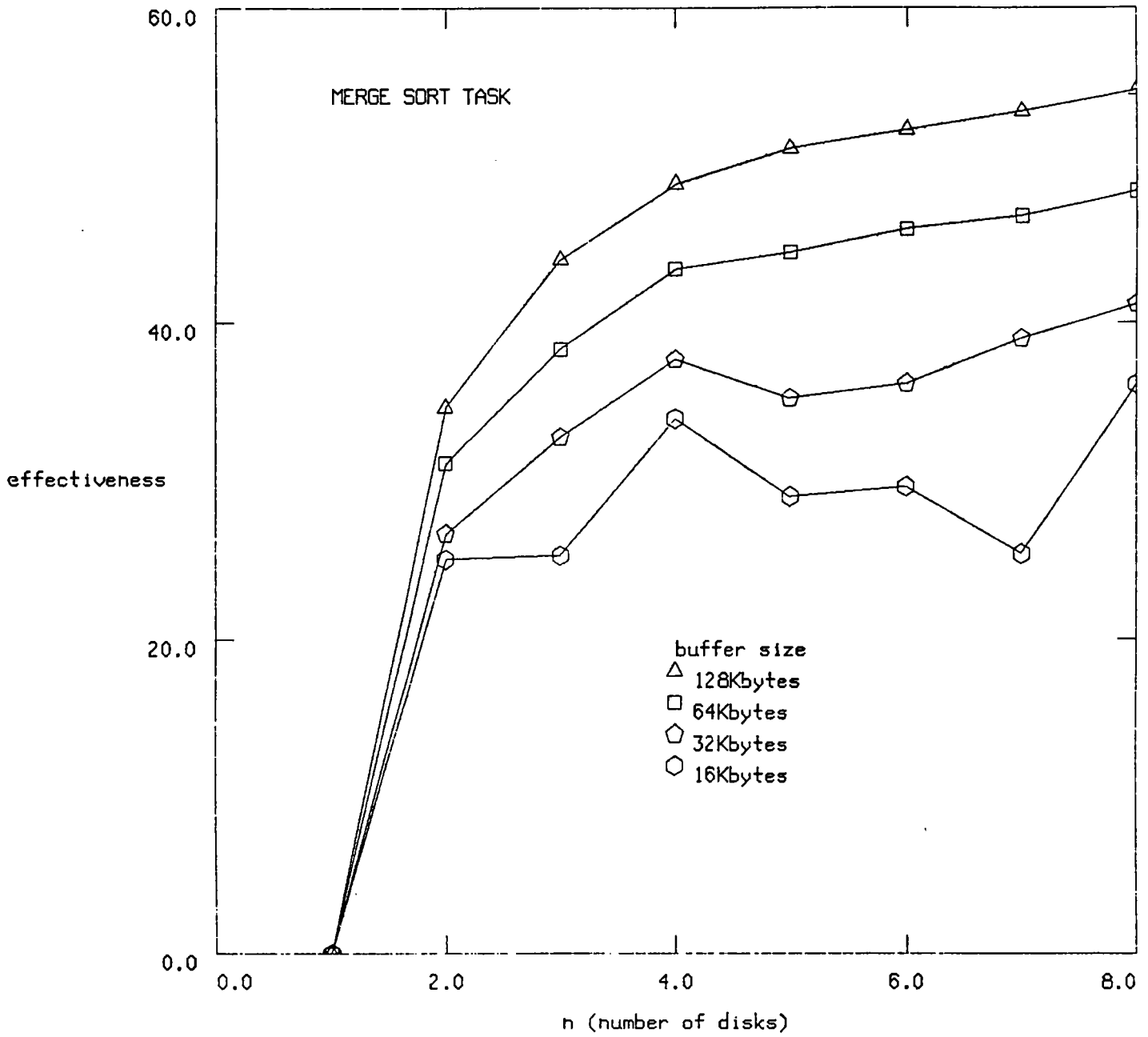


FIGURE 2

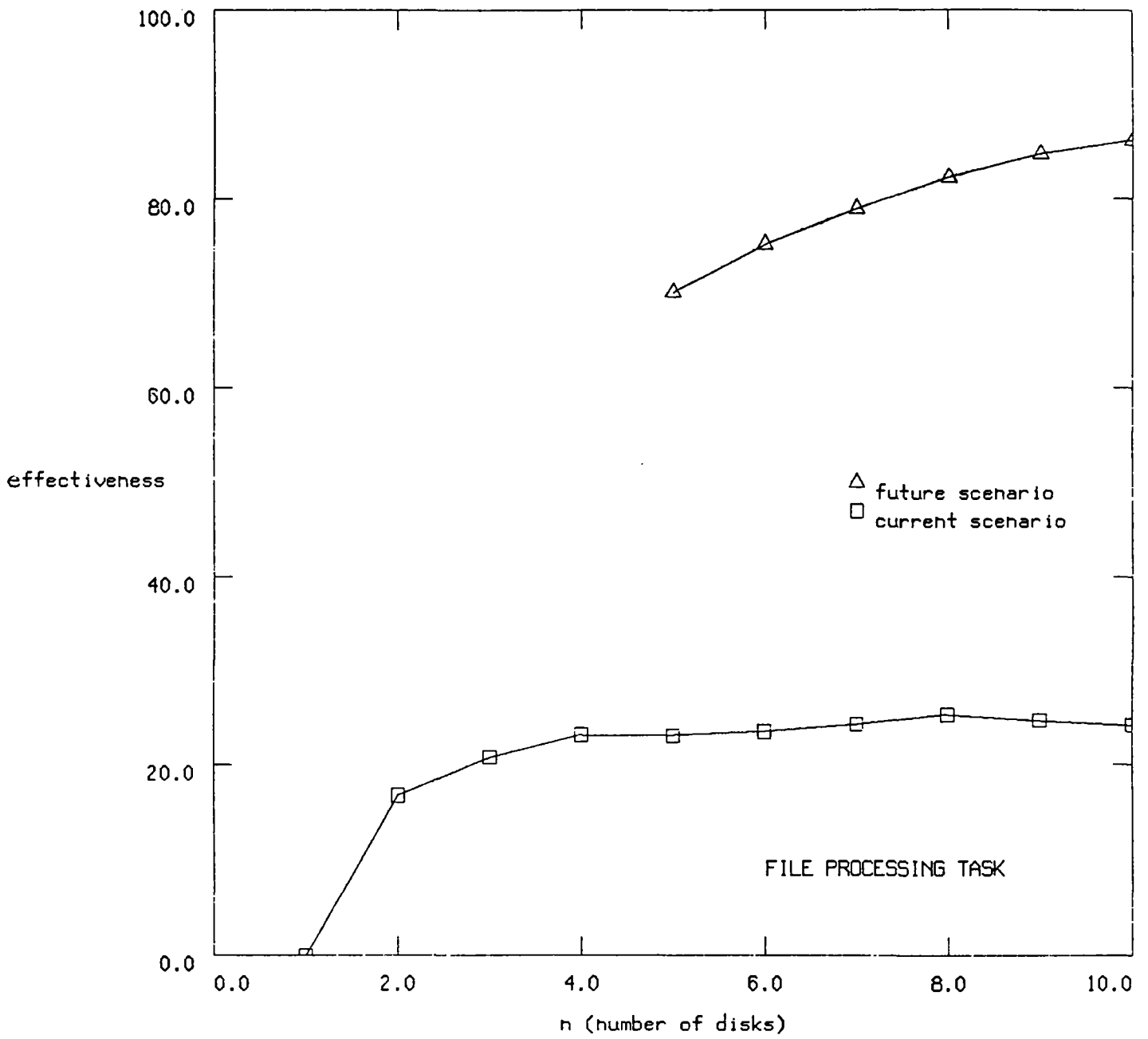


FIGURE 3

A Project on High Performance I/O Subsystems

*Randy H. Katz, John K. Ousterhout, David A. Patterson,
Michael R. Stonebraker*

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720

1. Introduction and Overview

Computing is seeing an unprecedented improvement in performance; over the last five years there has been an order-of-magnitude improvement in the speeds of workstation CPUs. At least another order of magnitude seems likely in the next five years, to machines with 100 MIPs or more. Within the architecture research community, there is talk of even larger, more powerful machines, executing as many as 10^{12} operations per second. Unfortunately, we have seen no comparable breakthroughs in I/O performance; the speeds of I/O devices and the hardware and software architectures for managing them have not changed substantially in many years.

What will unbalanced improvements in performance mean? Twenty years ago, Gene Amdahl was asked to comment about the Illiac-IV. He noted that while the vector portion of programs might run much faster, a major portion of the programs would run essentially at the same speed. In what has come to be known as Amdahl's Law, he observed that no matter how much faster one piece of the program would go over traditional computers, overall performance improvement is limited by the part of the program that is not improved. This equation formalizes his criticism:

$$Speedup = \frac{1}{(1-f) + f/k}$$

where f is the fraction of time spent in the fast mode, and k is the speedup of fast mode over the slow mode.

Thus a computer that runs compute-intensive benchmarks 10 times faster and tries a program that spends just 10% of its time in I/O will only improve performance by a factor of 5. Making this CPU go 10 times faster again--100 times current computers--will only double the speed of that program. Without major increases in I/O performance and reliability we think that transaction processing systems, supercomputers, and high-performance workstations will be unable to achieve their true potential.

Our research group is pursuing a program of research to develop hardware and software I/O architectures capable of supporting the kinds of shared memory multiprocessors, workstations, and supercomputers that will appear in the early 1990s. The project has three overall goals:

High Performance. We are developing new I/O architectures and a prototype system that can scale to achieve significant factors of improvement in I/O performance, relative to today's commercially-available I/O systems, for the same cost. We believe this speedup can be achieved using a combination of arrays of inexpensive personal computer disks coupled with a file system that can accommodate both striped and partitioned file organizations.

High Reliability. To support the high-performance computing of the mid-1990's, I/O systems cannot just be fast; they must also be large. The unreliability of current disks requires nightly backups on magnetic tape. This process is expensive and unmanageable even with our current systems, and will not scale to meet the needs of the 1990s. By using just 15% extra disks in our disk array to act as standby spares or error recovery disks, we can increase the MTBF of the magnetic media to over 100 years--obviously to the point where media reliability is no longer an issue. With write-once optical disks for user-requested archiving, we hope to eliminate magnetic tapes from the next generation of I/O systems. Our goal is to permit much larger file systems with increased file reliability and decreased human intervention.

Scalable, Multipurpose System. In the past, designers of mainframes and supercomputers have built special-purpose I/O processors that were useful only for one machine, and would need to be redesigned for each new generation, not to mention for each manufacturer. We are investigating both tightly and loosely coupled I/O architectures. For the former, we are coupling a disk array to the memory system of a shared memory multiprocessor of our own design, called SPUR. Our long term intention, however, is to structure the I/O architecture around very-high-speed networks and high-performance file servers in a way that makes diskless mainframes and supercomputers practical. Architects of new computers will be able to guarantee themselves high-performance I/O merely by providing an efficient network interface; they will not need to implement a large collection of controllers and channels for different I/O devices. As demands for I/O increase, the designer has the option of adding more I/O systems to a network if there is enough bandwidth available, or adding extra network interfaces if it is not. Moreover, since the I/O systems will be built from commercially-available computers, they can act either as I/O processors for supercomputers or as "super" file servers on the network. Our goal is to build a multipurpose next-generation I/O system from off-the-shelf disks and CPUs.

The expected physical artifacts that will evolve from this work are the following:

- (1) *A prototype rack-mounted disk array with VLSI disk controllers of our own design:* a loosely coupled version of the array will be available over a high-performance local-area network, while a more tightly coupled version will be connected to the memory system of SPUR.

- (2) *A new file system to support the disk array and additional optical disks:* this will be implemented in the Sprite network operating system.
- (3) *A data management system specifically designed to exploit shared memory multiprocessors and disk arrays:* this will be based on POSTGRES [Stonebraker 86].

2. Characterization of I/O Intensive Applications

The range and diversity of I/O intensive applications can be represented by three different kinds of environments: *Transaction Processing*, *Network File Services*, and *Supercomputer I/O*. These run the gamut from processing large numbers of simple tasks that touch small amounts of data (transaction processing) to small numbers of bulk I/Os that must be handled with minimum delay (supercomputer I/O). An important design challenge is to develop an I/O system that can handle the performance needs of these diverse applications.

Transaction processing is characterized by enormous amounts of random data access, relatively small units of work, and a demand for moderate latency with very high throughputs. The best systems today can process at most a couple of hundred transactions per second, with a thousand transactions per second an ambitious goal.

The workloads presented to network file servers are more highly sequential in nature, and typically demand lower latencies but more moderate throughputs than transaction processing. A typical client may be sequentially processing a small number of files at the same time. Good measures of file server performance are the number of opens/closes per second and the number of bytes transferred per second. A measure of the effectiveness of the server is the number of clients that can be supported per server. The best SUN-3 based servers can support 50 clients running on SUN-3 hardware.

Supercomputer I/O is characterized almost entirely by sequential I/O. Typically, computation parameters are moved in bulk from disk to in-memory data structures, and results are periodically written back to disk. These applications demand minimum latency with low throughputs.

3. Hardware Platform for High Performance I/O

3.1. High Performance Via Disk Arrays

The large-scale economics of the personal computer and workstation marketplaces, in conjunction with advances in technology, have yielded high-density hard disk assemblies at very low costs. This allows the bandwidth of a disk system to be increased enormously by replacing a small number of large disks with very large numbers of inexpensive small disks [Patterson 87]. Consider a triple-density IBM 3380 disk drive, which can store 7.5 gigabytes in about 24 cubic feet, using about 10,000 watts at a cost of \$100,000. Within a year, a 100-MByte 3.5" disk cartridge (containing media, controller, buffer memory, and interface), taking up only 0.1 cubic feet, using at most 8 watts, will be available for less than \$1000.

Note that the seek times, rotational delays, and transfer rates do not differ by more than a factor of two between these two kinds of disks. Replacing an IBM 3380 by 100 of the smaller disks will yield a comparably priced system with higher density (10 GBytes), reduced volume (10 cubic feet), and lower power (1000 watts.) However, the greatest advantage of the array is the enormous increase in bandwidth (to 100 Mbytes/sec. or more) that can be achieved by operating all the disks concurrently.

We see three possible ways to structure a disk array: *independent drives*, *synchronous arrays* [Kim 86], and *asynchronous arrays* [Kim 87]. The simplest alternative is the independent-drive approach, where the drives are plugged into one or more servers using off-the-shelf controllers; it would be up to the operating system or application level software to find ways to operate all the drives in parallel. For application domains with large numbers of small independent requests, such as on-line transaction processing, this may well be the best approach. However, it does not necessarily improve the performance of any single application. For example, a program reading a single large file, or a program reading many small files sequentially, may be limited by the latency and bandwidth of a single drive.

The second approach is to build the array so that all disks spin, seek, and transfer in synchrony, a so-called *synchronous array*. Such an organization assumes that data is interleaved across the disks by bit or by byte. Synchronous arrays do not reduce seek time or rotational latency, but they provide high bandwidth by transferring to/from all disks simultaneously. Thus, for a single large application reading or writing a large file, this approach can provide a substantial speedup. The disadvantage of synchronous arrays is that they are difficult and expensive to build, and may not scale readily to arrays with tens or hundreds of disks.

The third approach, an *asynchronous array*, simulates the behavior of a synchronous array using unmodified off-the-shelf disk drives in conjunction with a new controller architecture. A read request is implemented by independent seeks and transfers from each of the underlying disks to a shared buffer in the controller. The controller then interleaves the data returned by the disk drives. This approach yields the performance benefits of a synchronous array with little additional hardware cost.

These three array structures are well matched to different I/O domains. A reconfigurable array is desirable, especially since the underlying drives are actually asynchronous anyway, and much of the differentiation can be left to file system software. For example, it may make sense to organize the array into several independent clusters each of which is itself a small asynchronous array. The ability to choose the degree of synchronization at run-time leads to even greater flexibility.

3.2. High Reliability Via Disk Arrays

One of the most interesting challenges and opportunities for disk arrays is to make them fault-tolerant [Park 87]. While one might believe that personal computer disks are not as rugged as larger disks, the opposite appears to be true. The

drive we mentioned above has a mean time between failure of 3.5 years and comes with a 5 year warranty. (The Fujitsu Eagle drive comes only with a 90 day warranty.) Nevertheless, an array of 100 disks will experience a failure 100 times more frequently than any individual disk (about every 300 hours for the arrays we envision); a straightforward design might interleave files across the array, so that a head crash anywhere in the array could make every file inaccessible.

However, arrays of small disks also have a potential to increase reliability at very low cost. We can organize the array into separate groups of disks for recovery purposes [Patterson 87] (see Figure 1).

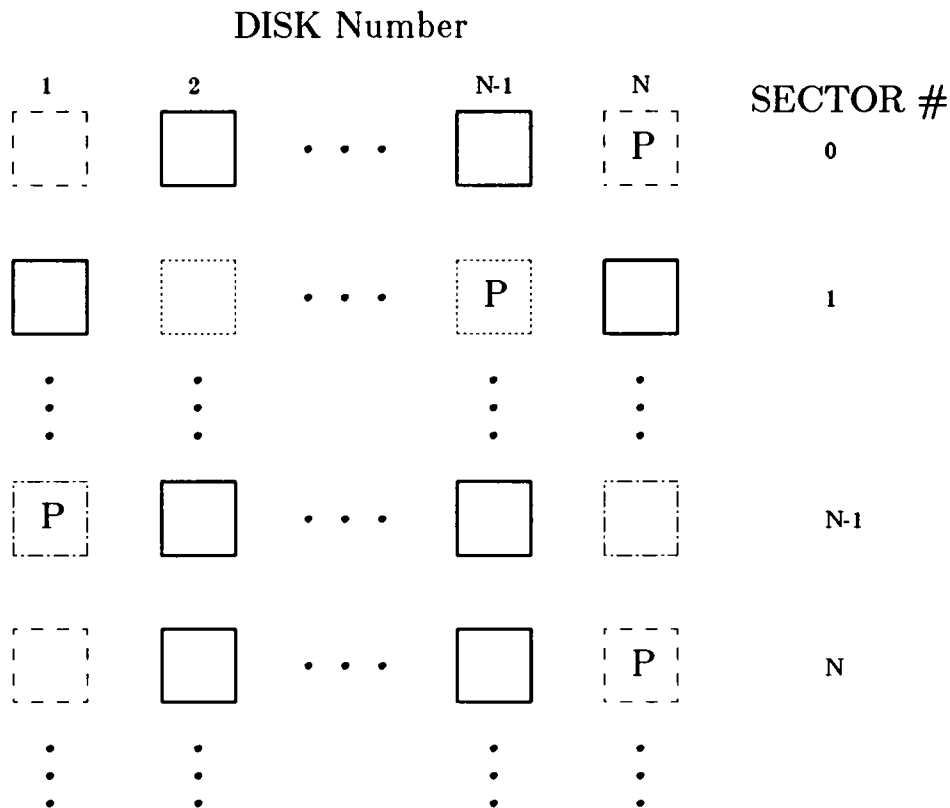


Figure 1: Interleaved Parity Sectors in an N Disk Recovery Group

Alternative rows of the array have their parity blocks on different disks within the group. Up to $N/2$ writes can be serviced at the same time, since one data sector and one parity sector are touched for each logical write.

Since disks include codes that can determine if any disk read has failed, we can recover information from a failed disk just by adding one extra disk per group to calculate the parity for the group. Thus the cost of the redundancy is just $1/G$, where G is the number of disks in a group. If we add a small pool of standby spares to the array, we can almost arbitrarily increase reliability; the question then becomes what are the chances of many reliable components failing within a very small period.

One potential area of weakness might be the power supply, but much lower power requirements of the disk arrays offers new options. A surge-protected, uninterruptible power supply with battery backup could provide the time to let all disks shut themselves down in the event of a power failure.

Note that such redundancy techniques are only plausible when building a single system from hundreds of components. The only economical solution to large-disk reliability is to double the number of disks. Because of our use of large numbers of inexpensive disks, we expect the parity disks and standby spares to increase the disks in the system by less than 15%.

3.3. High Capacity Via Optical Storage

The database and supercomputer needs of the 1990's will require not just high-performance file systems, but also file systems with extremely high capacity. Although the capacity per dollar of magnetic disks has been improving dramatically, optical storage appears to offer much greater capacity per dollar, particular if jukeboxes are used. We plan to address the issue of how to include optical disks in the storage hierarchy. At the very least, optical disks should provide a less bulky, more efficient, and more convenient archiving mechanism than magnetic tape (e.g., no operator assistance should be needed to retrieve archived data from an optical disk in an on-line jukebox).

4. Operating and File System Issues in High Performance I/O

4.1. Multipurpose Via Networking

For the non-transaction processing realm, a key to separating I/O architecture from CPU architecture is the availability of networks and latencies equivalent to high-performance backplanes. Such networks will be available within a few years. The technical challenge is to provide controller and system architectures that allow very high speed networks to be fully utilized. Our experience with the Sprite operating system [Nelson 87] is that copying costs are the major obstacle to high protocol bandwidth. For example, when data arrives at a node, the network controller just places it in a buffer; invariably, the operating system ends up copying it from the buffer to its desired destination (e.g., a cache block). Sprite's current protocols can utilize about 2 Mbits/second of burst network bandwidth per MIP of CPU power, but we believe that this can be increased substantially, so a single 100 MIP file server can provide a supercomputer with I/O bandwidth of a Gigabyte/second or more. The most promising approach appears to be a redesign of network controllers to reduce copying. For example, if packets are tagged with very simple type information interpreted by the controller, it could subdivide incoming packets and place various header and data fields in separate areas indicated by the packet.

4.2. High Performance Via Caching

Our previous work on Sprite has already demonstrated the benefits of caching on a small scale: Sprite eliminates most of the performance impact of remote

file access by caching recently-accessed file blocks in the main memories of client and server machines [Nelson 87]. We are continuing to explore and extend the Sprite caching techniques by measuring the effects of even larger cache sizes and tuning the Sprite caching mechanisms for higher performance machines.

Caching is effective in reducing network and disk accesses for reads, but it still leaves open the issue of writes. A write-through file cache has serious performance implications, but a write-back cache has reliability problems in the face of system crashes. One possible solution is to use a special disk or disk array as a cache backup, and continuously update this disk with the contents of the cache. Information could be written sequentially to the disk in cache order, so that there would be only a single seek per scan of the cache: whereas writing through to files would require at least one seek per file. A disadvantage of this approach is that it might require a backup disk for each cache.

5. Database System Issues in High Performance I/O

5.1. Flexibility via Partitioned and Striped File Allocation

Recent research [Livny 85, Salem 86] has suggested striping files across the entire collection of disks within a disk array. In this way, large sequential I/Os can be processed by reading all disks in parallel, and very high bandwidth can be achieved. However, an alternative approach, well suited to random accesses, is to partition data among the drives based on key ranges, as is often done in distributed database machines. Clearly both allocation strategies are of use for high performance I/O, and we are building a *two dimensional* file system that can support both.

Each disk extent can be described in terms of the following 4 tuple:

D_i : the drive number on which the extent starts

T_i : the track number on which the extent starts

S_i : the height of the extent in tracks

W_i : the width of the extent in disks

Each extent is a physically contiguous collection of tracks beginning at T_i and continuing to $T_i + S_i$ on each of disks D_i through $D_i + W_i$. Addressing within the rectangle is striped. We assume that if a file is striped across a disk that has crashed (and is not repaired as described in Section 3), then that file becomes temporarily unavailable for sequential accesses.

What is the appropriate choice of W_i for different applications? Consider first large object retrieval where high availability is not a consideration (e.g., supercomputer access to a database). In this case, W should be chosen to be the full width of the array, to provide maximum bandwidth to the application. If availability is an issue, then the system administrator must match the width of the extent to the desired bandwidth of the application. In a transaction processing environment with only small transactions, one should choose W as large as possible to minimize hot spots. Finally, for ad-hoc queries where query parallelism is desired, the DBMS should partition each relation into a collection of files. There is no advantage to spreading such files across more than a few disks due to

query plan saturation.

6. Summary

In this short paper we have described a coordinated hardware-software attack on the I/O bottleneck. Relying on new ideas in operating system for file caching and I/O buffering, and exploiting the arrival of low-cost disks, we plan to demonstrate significant factors of improvement in performance and reliability over current commercially-available systems. Over the next three years, we expect to demonstrate these claims via hardware-software prototypes.

The transaction processing portion of this work is currently funded by the National Science Foundation under grant # MIP-8715235.

7. References:

M. D. Hill, et. al., "Design Decisions in SPUR: A VLSI Multiprocessor Workstation," *I.E.E.E. Computer Magazine*, V 19, N 11, (November 1986).

M. Y. Kim, "Synchronized Disk Interleaving," *I.E.E.E. Transactions on Computers*, V C-35, N 11, (November 1986).

M. Y. Kim, A. N. Tantawi, "Asynchronous Disk Interleaving", IBM T. J. Watson Research Center TR RC 12497 (#56190), Yorktown Heights, NY, (February 1987).

M. Livny, S. Khoshafian, H. Boral, "Multi-Disk Management Algorithms", MCC Technical Report DB-146-85, Austin, TX, 1985.

M. Nelson, B. Welch, J. Ousterhout, "Caching in the Sprite Network File System," A.C.M. Symposium on Operating System Principles, Austin, TX, (November 1987). To appear in *A.C.M. Transactions on Computer Systems*.

A. Park, K. Balasubramanian, "Providing Fault Tolerance In Parallel Secondary Storage Systems," Computer Science TR 057-86, Princeton University, (November 1986).

D. A. Patterson, G. Gibson, R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," Report No. UCB/CSD 87/391, U. C. Berkeley Computer Science Division, (December 1987). Submitted to A.C.M. SIGMOD Conference, Chicago, IL, (May 1988).

K. Salem, H. Garcia-Molina, "Disk Striping", Computer Science Technical Report, Princeton University, 1986.

M. R. Stonebraker, L. A. Rowe, "The Design of POSTGRES," A.C.M. SIGMOD Conference, Washington, D.C., (May 1986).

Disk Performance and Access Patterns for Mixed Database Workloads

Carolyn Turbyfill
Tandem Computers, Cupertino, California

1. INTRODUCTION

Disk performance is characterized in terms of random and sequential access times. Algorithms used in database systems where the primary copy of the data resides on disk are designed to minimize the number of accesses to disk and the total amount of data retrieved. Disk access patterns depend on the placement of data on and between disks, and on the database workload. Typically, On Line Transaction Processing (OLTP) requires efficient random access to data on disk, while batch processing and ad hoc queries require efficient sequential access to data. However, the trend toward the use of large main memories is making sequential access more important for both workloads [BITTON, SALEM, TURBYFILL].

As relational database technology has matured, user expectations of the functionality and performance of database systems have risen. In particular, users want a single database that supports a mixed workload of OLTP, batch reporting, and ad hoc queries while maintaining acceptable transaction throughput and response times. This is referred to as a "single database strategy" in contrast to the "dual database strategy" which is prevalent today. The dual database strategy uses two databases, one for OLTP and the other for ad hoc queries and batch reporting. The database used for ad hoc queries and batch reporting is periodically updated (or refreshed) using the OLTP database. In this paper, we will summarize the factors that determine a disk's performance, describe database workloads in terms of disk accesses, and discuss approaches to supporting a single database strategy.

2. DISK PERFORMANCE

Over the past decade, processor speed has been steadily improving while disk technology has remained relatively constant at a maximum of 20 to 30 random I/O's per second. Typical characteristics for today's high end disks are exemplified by the IBM 3380 which has a track to track seek time of 3 seconds, an average random seek time of 16.7 milliseconds, and a maximum seek time of 30 milliseconds. The IBM 3380 has a raw data rate of 3 megabytes per second. The average seek, latency, and transfer time for a 4K page

is 23 milliseconds [MACKERT]. It has has 47,476 (formatted) bytes per track, 885 tracks per platter, and 15 platters.

The maximum data rate achievable on a particular disk is a function of circumferential bit density and rotational speed. On most disks, 1 revolution requires 16.7 milliseconds:

$$.01666... \text{ sec/rev} = 3600 \text{ revolutions per minute}$$

The upper limit for a disk's transfer rate is the number of bits that can be transferred in 1 revolution. The instantaneous (or raw, or unformatted) data rate for a disk is equal to:

$$\text{Raw data rate} = (\text{unformatted bytes/track}) / (\text{sec/revolution})$$

For instance,

$$3 \text{ Mb/second} = (52,428 \text{ bytes/track}) / (.0167 \text{ sec/revolution})$$

The raw data rate cannot be sustained over cylinder boundaries as data transfer is interrupted when the disk arm is moved to another track. Furthermore, the raw data rate is not even sustained on a single track. Tracks are formatted in sectors, typically 512 bytes/sector. Between each sector is an intersector gap where data is usually not stored. The utilization of a track is the fraction of bytes in a track where data is actually stored. The formatted data rate of a disk is:

$$\text{Formatted data rate} = (\text{raw data rate}) \times (\text{track utilization})$$

For instance,

$$2,848,603 \text{ Mb/sec} = (3 \text{ Mb/sec})(.905) = (3 \text{ Mb/sec})(47476/52428)$$

In sequential access, data is read from contiguous tracks. In previous work, we found that the IBM 3380 disk could deliver 3.46 megabytes of data in 2.47 seconds for a formatted access rate of 1.4 megabytes per second [TURBYFILL].

A disk technology trend is toward higher circumferential bit density that will enable higher raw data rates. IBM is expected to announce a controller and channel with data rates of 6 megabytes per second this year [BOZMAN]. Circumferential bit densities are also expected double soon [BAJOREK].

For random access, parameters of importance for disks are the average seek and latency. Assuming that data required

for 1 I/O is always contained in 1 track, at 60 revolutions per second, the maximum number of I/O's that could be accomplished by a single actuator is 60. In the best case, one could be scanning a file where 1 track is read for each I/O and the subsequent I/O is on an adjacent track. If the sectors are perfectly aligned, so that the only delay between reads is a track to track seek with no latency, then each I/O would require 20 milliseconds (3 millisecond seek plus 16.7 milliseconds/revolution). This would permit 50 I/O's per second. For random I/O, a minimum of 1 revolution of a disk is generally missed due to a random seek and latency, 30 random I/O's per actuator is an upper limit in practice. Controller contention due to channel contention or queuing of I/O requests may cause additional missed revolutions.

Choices of hardware and software parameters are dependent on the relative importance of sequential and random access. In terms of disk technology, sequential access favors higher bit densities, whereas random access favors a shorter seek time, usually accomplished through increasing the number of actuators relative to the number of tracks. In terms of disk formatting and software parameters, sequential access favors larger sector and block sizes. (A block consists of one or more sectors and is the minimum unit of I/O from disks.) Random access favors smaller sector and block sizes, except when spatial locality of reference can be exploited. In addition, disk striping [SALEM], where a data block is multiplexed across a group of disks so that it can be read or written in parallel, and other parallel or distributed approaches [LINDSAY] can be used to increase the effective I/O bandwidth of a system.

3. DEFINITION OF A MIXED DATABASE WORKLOAD

We group data management workloads into 3 basic types: OLTP, ad hoc queries, and batch.

OLTP: The Debit-Credit benchmark [ANON] that has become a standard for OLTP uses a transaction profile that consists of 3 single record updates and 1 single record insert. Typically, transactions that are executed online require a response time of 1 or 2 seconds. This limits the "size" of an online transaction to 10 or fewer I/O's. Of course, in transaction processing, larger transactions also occur. For instance, in an inventory database, entering and filling an order has two parts. Entering the order consists of the sequential insertion of a set of parts, usually keyed on (order #, part #, and customer #), into an orders table. One order may include hundreds of different parts. This part of the transaction can usually be completed online as the

hundreds of records are clustered on a few pages and can be written in 1 or 2 bulk I/O's. However, filling the order involves modifying the count for each part using (hundreds of) random I/O's in an inventory table. Typically, a large transaction involving hundreds of random I/O's is executed offline as batch transaction because its response time will be unacceptable for online execution.

AD HOC: An ad hoc query is an unplanned query that may range in complexity from the retrieval of a single record to a join and aggregate function that requires sorting, merging, and scanning multiple tables. Ad hoc queries can cause performance problems partially because the database may not be optimally designed for efficient query execution, or because the query inherently accesses a large amount of data (gigabytes). We expect that most ad hoc queries will be retrieval queries, that is, that they will not change any records in the database, although the result may be written to a temporary database table or to a system file. Ad hoc queries typically arise when a database is used for decision support.

BATCH: A batch workload results from programs that are executed on a regular basis. Batch workloads typically read and write large amounts of data. An example of a batch program is the daily updating of a master file by merging a daily activity file with the master file. Because batch workloads are planned and repeated, performance can be optimized through database design and system configuration.

Table 1 contrasts these three workloads in terms of data access: random or sequential, whether data is read/written from/to shared database tables, whether data is written to temporary database tables or other system files, and whether the workload characteristics are known ahead of time.

Table 1
Disk Access Characteristics by Workload Type

| TYPE | Random Access | Seq. Access | Read DB | Write DB | Write Other | Planned |
|--------|------------------|----------------|------------|-------------|----------------|---------|
| OLTP | Often | Often | Often | Often | Yes | Yes |
| Ad Hoc | Often | Often | Often | Seldom | Yes | No |
| Batch | Some | Often | Often | Often | Yes | Yes |

One entry in the above table that may appear counterintuitive is that sequential access occurs often in OLTP [SALEM]. In fact, sequential access occurs frequently in writing the transaction log and in performing checkpoints [BORR, LINDSAY]. Writing the transaction log involves sequential access with no seeks when the log is written on a dedicated disk, because the disk head is usually never moved from its position at the end of the log. A transaction is considered committed once a commit record is written in the transaction log on disk. All database pages changed by the transaction need not be written to disk since the log contains the necessary information to undo or redo a transaction in the event of a system crash. Checkpoints are used to limit the amount of the log that must be scanned for a restart. In a checkpoint, dirty pages that have been in memory for more than 1 checkpoint are flushed from memory to disk. Checkpoints can also be optimized to use sequential access. By writing the dirty pages out in sorted order by disk address, writes to disk will take place sequentially, although not all dirty pages will be contiguous and some seeks will occur.

It is also useful to compare OLTP, ad hoc queries, and batch processing in terms of relevant performance metrics. In OLTP, both throughput, in terms of transactions per second, and response time, in terms of seconds per transaction, are important. Response time, in terms of seconds, minutes, or hours, is the most relevant metric for ad hoc queries. For batch workloads, throughput, in terms of bytes per second, is the commonly used metric. Performance enhancements that reduce the total amount of work performed benefit both throughput and response time. For instance, in a "group commit" [BORR], commit records for many transactions are written to the log on disk in a single I/O. This reduces both CPU and disk utilization compared to writing each commit record with a separate I/O. On the other hand, parallelism reduces response time for all workloads and increases the throughput of data for batch processing, but does not necessarily reduce the total amount of work done in terms of CPU time and disk I/O's. In transaction processing, parallelism may be used to make online processing of "large" transactions (transactions involving more than 10 I/O's) feasible.

A "mixed workload" consists of OLTP, ad hoc, and batch queries. Reasonable expectations of the performance of a mixed workload include the following two requirements: 1) OLTP response times are not degraded by batch or ad hoc queries. 2) Batch or ad hoc queries have reasonable response times, particularly when the OLTP workload is low or halted.

4. APPROACHES TO EXECUTING A MIXED WORKLOAD

There are 2 basic approaches to executing a mixed workload. The first approach consists of executing OLTP, ad hoc queries and batch programs at the same time. From a user's perspective, this is the most convenient approach. It requires sophisticated load balancing in the operating system. From a performance perspective, there will be contention between OLTP and batch processing for locks. Ad hoc queries, as long as they are executed with "browse" access (nonrepeatable read, read only access), should not interfere with OLTP and batch programs. On the other hand, if a disk used for OLTP is also being accessed sequentially for batch processing or an ad hoc query, the sequential access will effectively become "interrupted sequential access", that is, the scan of the disk will be interrupted by random seeks to read or write data for a transaction, given that OLTP accesses are usually highest priority.

The second approach consists of using the same database, but executing batch programs and ad hoc queries in a fixed window of time, such as at night, when no OLTP is occurring. In this case, special techniques can be used to optimize OLTP and batch workloads separately. For instance, indices can be dropped while a batch job is updating the database, and then rebuilt before OLTP begins. This solution simplifies scheduling for an operating system.

5. SUMMARY

OLTP and ad hoc workloads require efficient random and sequential access. Batch workloads require efficient sequential access. There are 2 different approaches to supporting mixed workloads consisting of OLTP, ad hoc queries and batch programs on a single database: executing all three workloads simultaneously, and executing OLTP in a separate window in time from batch programs and ad hoc queries. Of these two approaches, executing all three workloads simultaneously is the most appealing alternative, effectively increasing the availability of the data by making it accessible at all times of the day for OLTP, batch processing, and ad hoc queries. For this strategy, careful physical design based on the relative frequency of retrievals and updates is required. Scheduling and load balancing are also important to this approach. Finally, for very large databases, parallel and/or distributed query processing will be required to meet user expectations of functionality and performance.

6. ACKNOWLEDGEMENTS

I am particularly grateful to Frank Clugage, Todd Sprenkle, Jim Gray, Andrea Borr, Franco Putzolu, and Jim Enright for sharing their experience and expertise with me on this topic.

7. REFERENCES

[ANON] Anon et al., "A Measure of Transaction Processing Power," Datamation, 1985.

[BAJOREK] Bajorek C., "Magnetic and Optical Devices to Quench Data-storage Quest," Electronic Design, January 1988.

[BITTON] Bitton D., et al., "Performance of Complex Queries in Main Memory Database Systems," Proc. 3rd Int. Conf. on Data Engineering, Los Angeles, 1987.

[BORR] Borr A., "High Performance SQL Through Low Level System Integration", to appear in Proc. of SIGMOD, Chicago, 1988.

[BOZMAN] Bozman J., "Ex-IBM exec sees '88 barrage," ComputerWorld, October 5, 1987.

[LINDSAY] Lindsay B., et al., "Notes on Distributed Databases", Research Report RJ2571, IBM San Jose, 1979.

[MACKERT] Mackert L. and Lohman G., "R* Optimizer Validation and Performance Evaluation for Distributed Queries," IBM Research Report RJ5050, IBM San Jose, 1985.

[SALEM] Salem K. and Garcia-Molina H., "Disk Striping," Proc. Int. Conf. on Data Engineering, Los Angeles, 1986.

[TURBYFILL] Turbyfill C., "Comparative Benchmarking of Relational Database Systems," Ph.D. Thesis, Technical Report 87-871, September, 1987.

International Symposium on Databases in Parallel and Distributed Systems

December 5-7, 1988
Austin, Texas



Symposium General Chair
Joseph E. Urban
University of Miami

Co-Program Chairs
Sushil Jajodia
NSF
Won Kim
MCC
Avi Silberschatz
UT-Austin

Program Committee
Rakesh Agrawal, *AT&T Bell Labs*
Francois Bancilhon, *INRIA*
John Carlis, *U. of Minnesota*
Doug DeGroot, *TI*
C. Ellis, *Duke U.*
Shinya Fushimi, *Japan*
H. Garcia-Molina, *Princeton U.*
Theo Haerder, *Germany*
Yahiko Kambayashi, *Japan*
Gerald Karam, *Carleton U.*
Michael Kifer, *SUNY-Stony Brook*
Roger King, *U. of Colorado*
Hank Korth, *UT-Austin*
Ravi Krishnamurthy, *MCC*
Duncan Lawrie, *U. of Illinois*
Edward T. Lee, *U. of Miami*
Eliot Moss, *U. of Mass.*
Anil Nigam, *IBM Yorktown Heights*
N. Roussopoulos, *U of MD*
Sunil Sarin, *CCA*
Y. Sagiv, *Hebrew U.*
Jim Smith, *ONR*
Ralph F. Wachter, *ONR*
Ouri Wolfson, *Technion*
Clement Yu, *UI-Chicago*
Stan Zdonik, *Brown U.*

Local Arrangement
Hong-Tai Chou, *MCC*

Publicity
Ahmed K. Elmagarmid, *Penn State*

Finance Chairman
Edward T. Lee, *U. of Miami*

Sponsored by:

IEEE Computer Society Technical Committee on Data Engineering & ACM
Special Interest Group on Computer Architecture (Approval Pending)

In Cooperation With:

IEEE Computer Society Technical Committee on Distributed Processing

Symposium Objectives

The objective of this symposium is to provide a forum for database researchers and practitioners to increase their awareness of the impacts on data models and database system architecture of parallel and distributed systems and new programming paradigms designed for parallelism. A number of general purpose parallel computers are now commercially available, and to better exploit their capabilities, a number of programming languages are currently being designed based on the logic, functional, and/or object-oriented paradigm. Further, research into homogeneous distributed databases has matured and resulted in a number of recently announced commercial distributed database systems. However, there are still major open research issues in heterogeneous distributed databases; the impacts of the new programming paradigms on data model and database system architecture are not well understood; and considerable research remains to be done to exploit the capabilities of parallel computing systems for database applications.

We invite authors to submit original technical papers describing recent and novel research or engineering developments in all areas relevant to the theme of this symposium. Topics include, but are not limited to,

- Parallelism in data-intensive applications, both traditional (such as Transaction Processing) and non-traditional (such as Knowledge-Based)
- Parallel computer architecture for database applications
- Concurrent programming languages
- Database issues in integrated database technology with the logic, functional, or object oriented paradigm
- Performance, consistency, and architectural aspects of distributed databases

The length of each paper should be limited to 25 double-spaced typed pages (or about 5000 words). Four copies of completed papers should be sent before May 1, 1988 to:

Won Kim, MCC, 3500 West Balcones Center Dr., Austin, TX 78759
(512) 338-3439, kim@mcc.com

Papers due: May 1, 1988
Notification of Acceptance: July 15, 1988
Camera-ready copy due: August 30, 1988



THE COMPUTER SOCIETY

1730 Massachusetts Avenue, N W
Washington, DC 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398