Appeared in 5th Annual Workshop on I/O in Parallel and Distributed Systems, IOPADS 97, San Jose, CA

Browsing and Placement of Multiresolution Images on Parallel Disks*

Sunil Prabhakar Divyakant Agrawal Amr El Abbadi
Ambuj Singh Terence Smith
{sunilp,agrawal,amr,ambuj,smithtr}@cs.ucsb.edu
Department of Computer Science
University of California
Santa Barbara, CA 93106.

Abstract

With rapid advances in computer and communication technologies, there is an increasing demand to build and maintain large image repositories. In order to reduce the demands on I/O and network resources, multiresolution representations are being proposed for the storage organization of images. Image decomposition techniques such as *wavelets* can be used to provide these multiresolution images. The original image is represented by several coefficients, one of them with visual similarity to the original image, but at a lower resolution. These visually similar coefficients can be thought of as *thumbnails* or *icons* of the original image. This paper addresses the problem of storing these multiresolution coefficients on disks so that thumbnail browsing as well as image reconstruction can be performed efficiently. Several strategies are evaluated to store the image coefficients on parallel disks. These strategies can be classified into two broad classes depending on whether the access pattern of the images is used in the placement. Disk simulation is used to evaluate the performance of these strategies. Simulation results are validated with results from experiments with real disks and are found to be in good agreement. The results indicate that significant performance improvements can be achieved with as few as four disks by placing image coefficients based upon browsing access patterns.

1 Introduction

With rapid advances in computer and communication technologies, there is an increasing demand to build and maintain large image repositories. Two examples of such repositories are the Chabot project [OS95] and the IBM Almaden Center's QBIC project [NBE+93]. This need is particularly critical for the experts working in the fields of remote-sensing and earth sciences. The Alexandria project [SF95] at UC Santa Barbara has been initiated to build a *digital library* for maps and image data, typically stored in raster or vector format. The project aims to provide its users access to maps, aerial photographs, census data, digital line graphs and elevation models, digital orthophoto quads, SLAR, landsat images, and other geo-referenced data over the internet. Clearly, significant hurdles remain before the Alexandria library can become fully operational. In particular, one of the major problems is the efficient storage and retrieval of image data.

Other than providing access to images, libraries and image repositories also provide the ability to browse the images. Browsing is particularly useful for users who are not sure what images match their interests and would like to navigate through a set of images to determine items of interest. In this process the user would typically view large numbers of images, most of which may not be of much interest. As compared to text, images have much higher storage needs. Consequently, access to image data places a high I/O and network demand. This demand increases with increasing resolution of the images. Therefore, displaying large numbers of full resolution images consumes a large amount of system resources.

A more efficient mechanism for browsing is to display low resolution representations of large images and to provide the ability to view higher resolution images of specific items of interest. Multiresolution image retrieval seems to be the key to providing efficient browsing of image data. This retrieval model is employed both in the QBIC [NBE+93] system as well as the Chabot [OS95] project. One method of providing images at multiple resolutions is storing multiple copies of images at various resolutions, as in the Chabot project. Due to the multiple copies the storage requirements of this scheme are high. A more storage efficient scheme is to use image decomposition techniques such as *wavelets* [Cas96]. Wavelets transform an image into multiple image coefficients without incurring any additional storage overhead, and one of the coefficients has visual similarity to the original image. Thus, the visually similar coefficient can be thought of as a *thumbnail* or *icon* of the original image. Browsing on these thumbnails reduces the amount of data transmitted. The original image can be obtained by carrying out image reconstruction recursively in the reverse direction.

System performance depends upon efficient retrieval of data from secondary storage. A standard technique for improving disk performance is to control the placement of data on disks. Several data placement techniques have been used to overcome the I/O bottleneck of secondary storage [AS95,

*Work supported by a research grant from NSF/ARPA/NASA IRI9411330 and NSF instrumentation grant CDA-9421978.

BG88, CABK88, PGK88, CP90, GHW90]. Some studies such as [GD90, LSR92], have relied on the well understood data structure and access patterns of relational databases to develop placement techniques. Others have worked in more general settings where the data is viewed only as independent files (e.g. [AS95, SWZ94]). Wavelet decomposed data are not as structured as relational databases, however they are not independent files. With the knowledge of the structure of wavelet decomposed data we can potentially do better than the more general techniques. In a similar setting, Chiueh and Katz [CK93] discuss pyramidal coding schemes for multiresolution video data and a storage layout on parallel disk arrays to provide real time jitter-free video retrieval. They do not however discuss the issue of the relative placement of stripes on disk for better performance. Placement techniques for multiresolution video data based upon the RAID-II prototype have been studied in [KK95]. Multiresolution video data is obtained through subband video coding which, like wavelets for images, generates several bands (coefficients) without data replication. Both these studies, however, are not directly applicable to non video data. The allocation strategy followed in Chabot is to place all thumbnails and metadata about thumbnails on disk and all higher resolution images on tertiary storage. This results in relatively fast browsing of thumbnails, at the cost of slower retrieval of higher resolution images. Query by Image Content [NBE+93, FBF+94] does not address the issues of image placement for performance improvement. There appears to be a general lack of placement techniques for image data. Furthermore, we believe that ours is the first effort that investigates the issue of image placement on disks when wavelets are used as the underlying storage architecture for images. From a browsing point of view, we incorporate content based techniques to determine appropriate placement strategies.

This paper addresses the problem of storing wavelet coefficients on disks so that thumbnail browsing as well as image reconstruction can be performed efficiently. Several strategies for storing the image coefficients on parallel disks are evaluated. Due to the complexity of working with real disks, we evaluated our placement techniques using simulation. Some representative results are later validated with experiments using real disks. The experiments were conducted on data obtained from 50,000 landsat images from the Map and Imagery Library at UC Santa Barbara. Both single user as well as multi-user environments are evaluated. The results from single user environment confirm the conventional wisdom that declustering improves performance when multiple disks are available. However, interpreting the results of various declustering policies in the presence of multiple users is more challenging and often contradicts conventional wisdom. When multiple clients are present, we conclude that not all declustering schemes result in better performance. This is particularly interesting since the vast majority of prior work was not tested in a multi-user environment. Overall we observe significant performance improvements with as few as four disks when multiple clients are present.

The rest of the paper is organized as follows. Section 2 describes the wavelet decomposition process and image content quantification techniques used in our experiments. Section 3 describes the proposed placement strategies that are evaluated. Section 4 presents the simulation setup used for the experiments. In Section 5 we present and discuss the results from the simulations. Section 6 presents experiments with real disks and Section 7 concludes the paper.

2 Review of Image Processing Techniques

The wavelets transform [Cas96] decomposes images into progressively lower resolutions as follows. Beginning with an original image of size $N \times M$ (for simplicity, assume that N and M are powers of 2), the image is decomposed into four parts, each of size $N/2 \times M/2$, as shown in Figure 1. The four parts or coefficients are labeled LL, LH, HL and HH. The LL coefficient is a low resolution copy of the original image, and the other three coefficients store other information necessary to reproduce the original image. The original image can be reconstructed from these four coefficients without loss. The LL coefficient can be further decomposed to yield another set of coefficients and an even lower resolution image. This process is repeated until the desired decomposition is achieved. Each image is therefore decomposed into several coefficients - one thumbnail and sets of LH, HL and HH coefficients for each level. In this paper, the process of reconstructing an image by one level is referred to as *expansion* of the image.

Content based browsing refers to retrieving images based on the similarity of the visual information contained in the images [NBE+93, OS95]. For example, having seen a specific hurricane image, a user may want to see similar images of hurricanes. In order to evaluate the similarity of images, one would first quantify the image content using several possible techniques. One possibility is to have an expert in the field annotate each image with some descriptive text. This method is undesirable because of the slow speed of the annotation process and also the dependence on and limitation of the subjective interpretation of the expert. Fortunately, there exist several image processing techniques for quantifying the image content [NBE+93, OS95]. We used the Gabor Transform [MM94] to generate a set of features for each image. The features for each image are viewed as a vector in a multidimensional space. These vectors have the property that images that are similar in content will be placed closer in the multidimensional space defined by the vectors. Hence the Euclidean distance between a pair of images gives a quantitative measure of the degree of similarity of their image contents [AMEM95].

3 Placement Strategies

The objective of our study is to propose and evaluate several different schemes for the placement of wavelet decomposed image data. We decided to use multiples of four disks for this study with most of the experiments using only four disks. The decision to use four disks came from the observation that when a higher resolution image is required, it is necessary to read all the coefficients in order to reconstruct the high resolution image. Since these items will be accessed together, it would be advantageous to place them on separate disks and to exploit inter-disk parallelism. Hence, in all but one of the placement strategies studied, the four coefficients of any given image are placed on separate disks. The placement of images is subdivided into two parts. The first determines the allocation of items to disks, and the second determines the relative placement of items on each disk. In the rest of the paper, the approaches taken to solve these two parts are referred to as *allocation* strategies and *placement* strategies respectively.

As a reference, a Random allocation and placement strategy is also considered where no declustering of images is done. The thumbnails and their coefficients are placed ran-

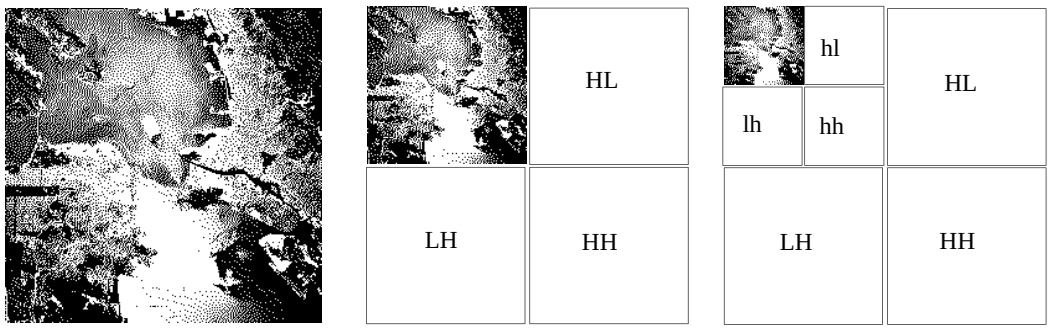

Figure 1: The Wavelet Decomposition Process

domly on any of the four disks. For fairness of comparison, these images are limited to a band of the disk that would be used by the other placement strategies. This is the placement that we expect if no meaning is attached to the thumbnails and their three coefficients. The following two image allocation strategies are evaluated.

1. *Distance based allocation*
2. *Round Robin allocation*

Distance based allocation refers to a strategy where the images that are close to each other (as measured by the distance between them, as explained above), are declustered across disks. Ideally, it is desirable that for each image the closest neighbors be allocated to a different disk. This is in general a hard problem. However, since an optimal declustering of images is not crucial for evaluation purposes, an approximation to the optimal solution is acceptable. The following algorithm is employed to decluster the thumbnails based upon distance.

The set of thumbnails can be viewed as nodes in a weighted graph, where the edges have a weight given by the distance between the connected images. Note that in any real library database, there will be hundreds of thousands of images. If we consider the distance between each pair, it will be computationally expensive working with these graphs. However, in practice, it is unlikely that we would be interested in comparing images that are very different from each other. Also in the context of browsing, we are seldom interested in getting large sets of related images. Hence, images that are sufficiently far away can be viewed as being infinitely far. In other words, we can limit the number of edges to only those with the shortest distances. This reduces the size of the graph significantly.

The declustering problem can be mapped to the graph k-colorability problem. We need to assign each item to a disk such that items that are neighbors are not assigned to the same disk, with a limited number of disks. Similarly the graph k-colorability problem is to assign colors to nodes in a graph such that no two adjacent nodes have the same color using at most k colors. This problem in general has been shown to be NP-complete [GJ79]. However, if we limit the maximum degree of each vertex to d_{max}, then a coloring using $d_{max} + 1$ colors exists [CLR90]. For such a graph a simple greedy algorithm can be used to find such a coloring. This algorithm assigns colors to nodes in a sequential traversal trying to assign the lowest possible color that is different from all colors assigned to its neighbors. It is always possible to do this because at every step, one more color is available than the maximum degree of any vertex. Note however, that such an algorithm may result in an unbalanced use of colors (or in our case disks) which is undesirable. A simple modification can remedy this problem. Instead of trying to assign the lowest color, we begin assignment with any color with equal probability.

We would like to point out that even though the determination of the distances between every pair of images requires $O(n^2)$ time, it is incurred only once. Moreover, for each image only a small number of the nearest images need to be known. If this information can be obtained without calculating the distances to every other image then the time for determining the distances can be reduced. Adding more images to the database changes the graph. This change potentially requires reallocation of images. However, since the allocation generated is not an optimal one, it is not necessary to reallocate all images for a small number of updates. For the similarity measure studied (image content), the distance between any two images is constant. However, for other similarity measures (e.g. one based upon actual user access), the distance between images could also change over time, requiring reallocation.

The graph corresponding to the declustering problem is complete since we can compute the distance between any two images. However, as noted above, we can ignore most of the edges and consider only edges to nearest neighbors. The resulting graph is directed since an edge from image a to image b may be present because b is among the nearest neighbors of a, however, a may not be among the nearest neighbors of b. To apply the above algorithm to the declustering problem, we limited the out degree of each vertex to $DISKS-1$, where $DISKS$ is the number of disks available, by considering only the edges to the nearest $DISKS - 1$ neighbors for each vertex. We did not consider any of the in-edges. The above algorithm is used to determine the assignment of image thumbnails to disks based upon distance for all the Distance based strategies.

Round Robin allocation refers to the allocation of thumbnails, randomly ordered, across the four disks in a round robin fashion. Starting with the first image, we allocate the thumbnails to disks in a round robin fashion. Therefore the first thumbnail is allocated to the first disk, the second to the second, ... , the fifth to the first disk, etc.

The allocation strategies determine which thumbnails are to be stored on which disk. The placement strategies, on the

other hand, determine on which disks the remaining three coefficients of each image will be stored. Under both allocation strategies, four placement strategies were tested: *Interleaved, Zonal, Separated* and *Bundled*. For simplicity, in this paper we limit ourselves to only a single level of decomposition. We hope to extend to greater levels in the future.

For the Interleaved strategy, all the four coefficients are placed, one after the other on each of the disks, in the order in which they are allocated to disks. Note that the three coefficients are allocated as soon as the thumbnail is allocated. Each item is placed on the next available sectors of the disk to which it has been allocated. Thus the placement of items on each disk proceeds sequentially, as shown in Figure 2. The following convention is followed for the figures shown in this section. The four coefficients are marked LL, LH, HL and HH. The number following these two letters identifies the image to which these coefficients belong. The sequence of these numbers refers to the order in which the thumbnails were allocated, as determined by the allocation strategy used. Therefore item number 1 is allocated before item number 2, etc. The orders will be different for the two allocation strategies. All four coefficients of a single image are shown using the same shading.

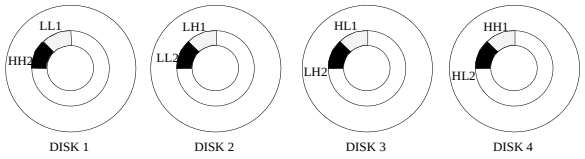

Figure 2: Interleaved Placement Strategy

The Zonal placement strategy divides each disk into four separate zones, one for each type of coefficient. Thus each type of coefficient, irrespective of what image it belongs to, is placed in the zone designated for that type. The placement within each zone proceeds in a sequential manner in the order in which items are allocated, as shown in Figure 3.

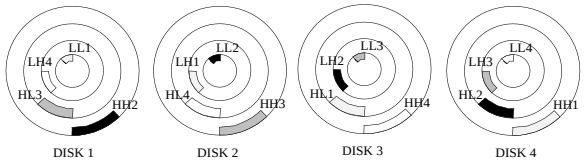

Figure 3: Zonal Placement Strategy

The Separated placement strategy differs from the others in that the thumbnails are placed on only three disks and all the other three coefficients are placed together on the fourth disk. This storage organization is similar to the placement of data in the Chabot project at UC Berkeley [OS95] where the lowest-resolution images or thumbnails are stored on disks and all higher resolution images are stored in tertiary storage (which is in contrast to the fourth disk in the Separated placement strategy). The thumbnails are placed sequentially on each of the three disks in the order of allocation, and the other coefficients are also placed sequentially on the fourth disk in the order of allocation. Figure 4 shows an example of this placement strategy.

The Bundled placement strategy does not decluster the four coefficients on different disks. Rather the thumbnail and the other three coefficients of each image are placed

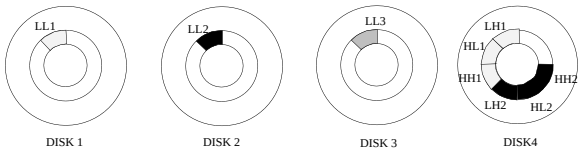

Figure 4: Separated Placement Strategy

sequentially on the same disk, as shown in Figure 5. This results in nine placement strategies: the Random strategy for reference purposes and four each under Distance based and Round Robin allocation strategies.

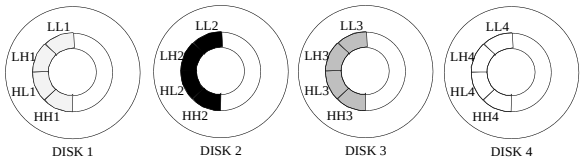

Figure 5: Bundled Placement Strategy

It should be pointed out that except for the Separated strategy, all other strategies use the four disks in a balanced manner. The Separated strategy uses three times the storage on the disk holding the non-thumbnail coefficients than the disks holding the thumbnails. This results in wastage of space on the three disks.

4 Simulation Setup

A client-server architecture is used to model our application. The server represents the image repository that maintains the holdings and services client requests. The clients are representative of users spread across a network that make requests to the server for images. For most of our study, the data for the images is not actually stored on disks, rather a disk simulator is used. Use of a simulator provides much greater flexibility. Some of the experiments are repeated with real disks. These results are discussed in Section 6. The simulator that we use has been developed by Kotz et al. [KSR94] based on the model developed by Ruemmler and Wilkes of HP laboratories [RW94]. The model is sophisticated and is capable of simulating multiple disks connected to multiple I/O buses.

Figure 6 shows the setup that was used. In order to study the effects of placement, it is necessary to have a model of the usage patterns. Since traces of actual user access patterns for image repositories are not available, we developed approximate models. In a browsing scenario, we expect that client accesses will be directed by similarity between images. For example, clients may wish to view images of a given geographical area, pictures taken by the same photographer, images with the same subject or of lakes etc. This similarity could be a more complex relationship involving several characteristics such as image content, date of publication and geographical location. For the purposes of our study we have assumed that the similarity measure is determined by image content alone. In order to quantify the similarity between two images, we use the image processing techniques described in Section 2.

The browsing access pattern used in our experiments can be summarized as follows. Each client executes a number

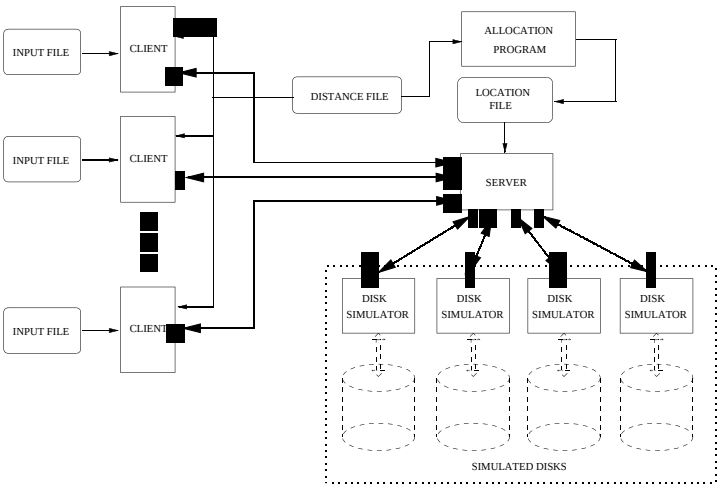

Figure 6: Schematic view of the Experimental Setup

of queries (typically 1000), each of which consists of possibly three requests to the server. The client first requests a thumbnail which is delivered by the server. Next the client requests the thumbnails of some of the nearest neighbors of this image. Following this the client could, depending upon the expansion level, choose to view higher resolution images of some or all of the retrieved thumbnails or move on to a new thumbnail. This pattern of access is close to what would transpire in a typical browsing scenario where items are retrieved by query and by content. We assume that the reconstruction of images is carried out at the clients. When an image is expanded the client already has one of the four coefficients and hence only the other three coefficients are retrieved from the disks. The choice of the initial item in each query is made based on a "hot-cold" distribution. We consider a certain fraction of the images to be "hot" images, and these are accessed more often than "cold" images. In the setup used, 20% of the images are considered to be "hot", and these items are accessed 80% of the time. The initial item for each query is determined randomly based upon this distribution. The choice of how many neighbors to request and what fraction of these to expand is made dynamically for each query based on certain probability distributions specified for each client independently.

The server receives requests from clients and services them. We model multiple disks using one disk per disk simulator process. Although the disk simulator is designed to handle multiple disks independently, it has a limitation in modeling asynchrony. The simulator interface is such that once a request is made, the simulator cannot be accessed until the request is completed. Thus for example, if the server dispatches a request for disk 1 and a request from another client arrives at the server for disk 3 the server cannot forward the request immediately to the simulator even though disk 3 is idle. Note that the simulator does provide non blocking I/O, but the synchronization is done via a blocking call, resulting in the same problem. Our setup eliminates this limitation, resulting in greater parallelism. In particular, four separate disks are used in order to investigate inter-disk parallelism for seeking. The simulator models the HP 97560 disk. Some of the parameters of these disks are given in Table 1. Further details of the disk and the model can be found in [RW94] and [KSR94]. Although the specified size of the cache for the HP 97560 is 128 Kilobytes, we limited the cache to 4 Kilobytes, in order to study the placement strategies in the absence of caching. Note that 4 Kilobytes is the size of the thumbnail and each coefficient of the images. Experiments were also conducted with larger cache sizes. The server specifies the disk, starting sector and number of sectors that are to be transferred from the disk. This information resides in a location file which is generated by placement programs using the various strategies that are to be evaluated.

Parameter	*Value*
Capacity	1.3 GB
Sector Size	512 Bytes
Sector Per Track	72
Tracks Per Cylinder	19
Rotational Speed	4002 RPM
Short seek [1]	$(3.24 + 0.4 * \sqrt{dcylinders})$ msec
Long seek	$(8.00 + 0.008 * dcylinders)$ msec
Cylinder Skew	18 sectors
Track Skew	8 sectors

Table 1: Table of Disk Parameters for HP 97560

The server and the disk processes communicate via messages. The server maintains a queue which holds outstanding requests for each disk. When the disk finishes servicing a request, it is given the next item in the queue to service. The server sends messages to clients as soon as an item for the client is received from a disk. The server is non blocking and keeps track of both client requests and disk responses. When a request is received from a client the server looks up the location file and queues the necessary requests for each disk. The queues for each disk are serviced in a first-in first-out manner.

The access times that are used to evaluate the performance reflect both the disk access time and the queueing delays. Each disk simulator had its own simulated clock. The requests from the clients arrive through the sockets in a real time fashion. The relationship between the clocks of the different disks and the arrival of the messages from the clients was not captured by this setup. We solved this problem by forcing each disk to actually observe the service that it reported for servicing requests. Thus when servicing a request, the disk processes delay until the real time elapsed is equal to the time that the simulator reported before sending the reply to the server. In this way, we synchronized the simulated clocks and were able to mix simulated times and real times. The significant differences in performance of the different strategies would largely depend upon the time spent waiting for the disks, i.e. the queueing times. The queueing time for any item waiting in the disk queue was estimated as the sum of the disk times for all items that precede the item in the queue plus half the disk time for the item that was being serviced by the disk (if any) at the time that the item joined the queue. The reasoning for taking half the time is that we expect that new items may arrive at any time between the time that the item at the head of the queue is sent to the disk and the time that the disk send the response for this item. Hence on the average, we expect that a new item will have incurred a delay equal to half the time that the disk spent servicing the request at the head of the queue. For simplicity, we used a closed queueing model.

5 Simulation Results and Analysis

The dataset for the experiment consists of 50,000 landsat images. Each image was considered to have been decomposed by one level using the wavelets technique described earlier to yield a thumbnail and three coefficients, each of size 4 Kilobytes. The images were also processed using the Gabor filters to produce feature vectors each with 24 dimensions. The Euclidean distance among these vectors were computed to determine the 20 closest neighboring images for each image. The closest 20 images were sufficient because in practice the client will not want to be overloaded with a large number of images while browsing. Further, 20 thumbnails can appear on a screen reasonably.

5.1 Basic Experiments

Four sets of experiments were conducted for each placement strategy, each with a different level of expansion - 0%, 25%, 50% and 100%. In order to study the performance of the strategies with different levels of client concurrency, we varied the number of clients from 1 to 8 for each set of experiments. Each client executed 1000 queries. For each query, the client picks an image at random using the "hot-cold" probability distribution. Each client in a concurrent set has a different seed for the randomization function. Across different strategies, the clients use the same set of seeds so that the sequence of accesses would be the same for fairness of comparison. The range of values of the various parameters used in these sets of experiments are summarized in Table 2.

Parameter	*Range of Values*
Number of Clients	1 - 8
Number of queries/client	1000
Number of neighbors	3
Amount of Expansion	0%, 25%, 50% and 100% of retrieved thumbnails
Dataset size	50,000 images
Maximum degree (d_{max})	3 (7)
Hot-Cold Distribution	20% hot images accessed 80% of the time
Think time [2]	0
Number of Disks	4 (8)
Thumbnail size	4KB (8KB)

Table 2: Table of Parameters for Landsat experiments

The Random placement of images was used as a reference for comparing the performance of each placement strategy for the four expansion schemes. For each combination of expansion level, number of clients and placement strategy, an average access time, over all concurrent clients, was computed. This time included the disk time and queueing time for each request. The results presented in the paper show the ratio of this average time for each strategy to the average time for the Random strategy. Therefore, a ratio of 1.0 means that for the given expansion scheme and number of concurrent clients, the strategy is as good as the Random strategy, a higher ratio means that the strategy is better than the Random strategy.

In all experiments conducted, it was found that all placement strategies based upon the Round Robin allocation strategy performed worse than the corresponding Distance based

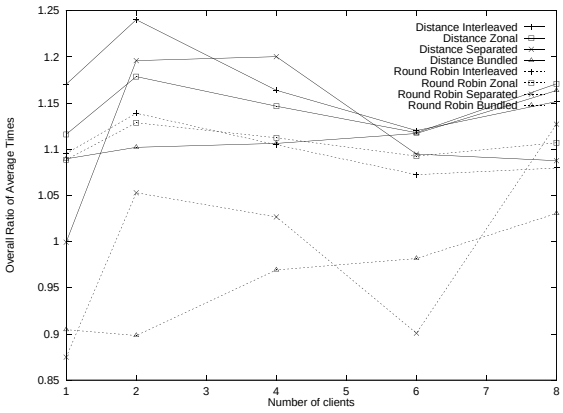

Figure 7: Comparison of Distance Based and Round Robin allocation for 25% expansion

placement strategies. Hence, for reasons of brevity the results for Round Robin allocation are not presented in the rest of the paper. In Figure 7 we present a representative example for 25% expansion which shows that it is always beneficial to use Distance based allocation as opposed to Round Robin allocation.

Figures 8 and 9 report the results for Distance based experiments with no expansion, 25%, 50% and 100% expansion respectively. In each graph, the x-axis represents the number of concurrent clients and the y-axis gives the average performance ratio.

Figure 8(a) illustrates the results for no expansion. In this case the Zonal placement strategy performs best since only thumbnails are accessed and all thumbnails are placed in the same zone on each disk. Due to the break-up into zones, the range over which thumbnails can be located on any disk is one-fourth for the Zonal case as compared to Bundled and Interleaved. Hence the average seek time is much smaller which results in the better performance. The Interleaved and Bundled strategies perform almost identically when there is no expansion. This is because the only difference between the two is in the location at which the three coefficients of each image are stored (on the same disk as the thumbnail or on the other three disks), and since these coefficients are not accessed, the performance is almost identical. The Separated placement strategy performs worse than the others. This poor performance is caused by the fact that the Separated placement strategy has only three disks over which the thumbnails are declustered resulting in less inter-disk parallelism than the others. This effect is even greater as concurrency increases because there exist more opportunities for parallel I/O for all the strategies.

When expansion is introduced, as seen in Figure 8(b), Interleaved begins to outperform Zonal, since Zonal is now compelled to make long seeks for expansion in order to retrieve all three types of coefficients from all three zones, whereas Interleaved is more likely to find all three coefficients close to the corresponding thumbnails. From Figures 9(a) and 9(b) it can be observed that as the amount of expansion increases even further, Interleaved outperforms Zonal for all concurrency levels. Interleaved also begins to outperform Bundled because the three coefficients are on three different disks as opposed to being on the same disk as in Bundled. Hence, Interleaved is able to get parallelism in its accesses

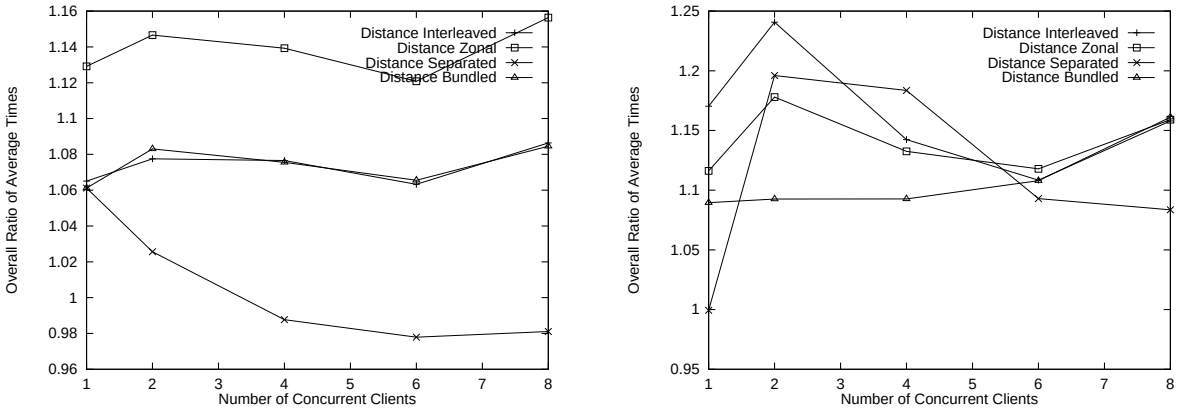

Figure 8: Distance based Allocation for 50,000 images with (a)0% and (b)25% expansion

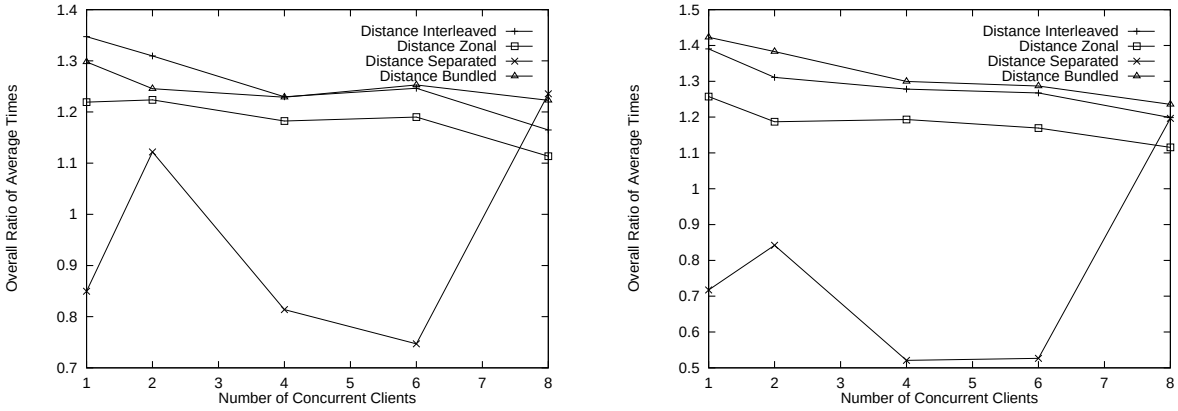

Figure 9: Distance based Allocation for 50,000 images with (a)50% and (b)100% expansion

whereas Bundled must retrieve all three items from the same disk. Note however, that the three items are located contiguously on the same disk, hence Bundled will not incur additional seeks. This advantage of Bundled pays off when the level of expansion or degree of concurrency is increased, as seen in Figures 9(a) and 9(b). Bundled outperforms Interleaved at higher degrees of concurrency or greater expansion. The explanation for this behavior is that when there are a lot of requests for expansion, either due to a large number of clients or a greater degree of expansion, Interleaved incurs more seeks on the average than Bundled. Hence the gain from having all three coefficients adjacent on the same disk pays off. We therefore conclude that declustering the image coefficients is not always beneficial. In particular for the same level of expansion, image declustering losses its advantage when concurrency is high.

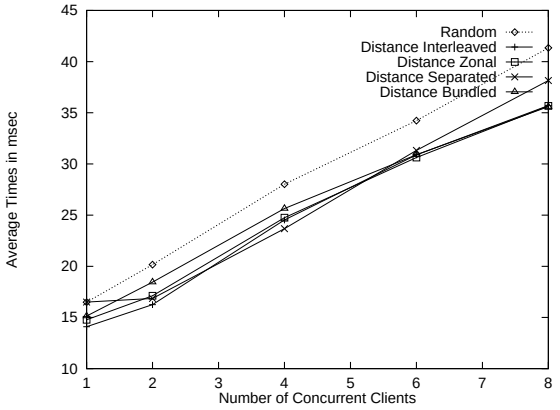

Figure 10: Actual average access times for 25% expansion

In order to understand the behavior of the Separated strategy, it is more instructive to analyze Figure 9 before trying to explain the behavior seen in Figure 8(b). In both graphs Separated has a jump at a concurrency of 2, it then drops and later picks up at a concurrency of 8. This apparently anomalous behavior can be explained by examining the access patterns in detail. In each query, a client first accesses an initial thumbnail, then accesses three other thumbnails and finally a number of the other three coefficients, where the exact number depends on the expansion scheme. Let us consider the actual average access times instead of their ratios to the Random strategy. These can be seen in Figure 10 which shows the average access times in milliseconds for the five strategies. For each strategy, as the number of clients is increased, there is greater contention for the disks and the average access times increase almost linearly. Separated however, has a constant average time for both one and two concurrent clients. This is the cause for the anomalous peak as noted in the ratio graphs. The reason for no increase in access times is that under the Separated strategy, the thumbnails are placed on three disks, and all the other three coefficients are placed on the fourth disk. This means that requests for thumbnails, irrespective of the image, are directed to the three disks and requests for expansion are directed to the fourth disk. Hence, in the situation where there is only one client, these two sets of disks are in use alternatively. When the request for thumbnails is being processed the fourth disk is idle and when an expansion request is being processed only the fourth disk is in use. Hence there are large periods during which either set of disks is idle. When the concurrency is increased to two, these idle times can be used to service requests for the other client, without introducing any significant additional delays. In other words, when the thumbnail request for client 1 are being processed by the three disks with thumbnails, an expansion request for client 2 can be serviced by the fourth disk. Therefore each client does not have to wait for the other, resulting in little or no increase in the average time.

When concurrency is increased to 4, this alternation is no longer possible and the performance of Separated drops. This is because the fourth disk is a bottleneck as it must service all the requests for expansion and also the requests for thumbnails can be shared only by three disks as opposed to four disks for the other strategies. Surprisingly however, Separated begins to perform at a level comparable to the other strategies when the concurrency is 8. This occurs due to the performance of the fourth disk which holds the three coefficients for all the images. Expansion requests that are received from clients get queued for this disk in the order in which they are received. Hence the three coefficients for either 2 or 4 items which are closely related in terms of distance are retrieved. Since these items would be placed close to each other, large seeks will not be incurred for retrieving them. Also only one seek is required for each image followed by the transfer of three consecutively placed coefficients. A large seek is needed only after servicing the expansion request for one client and moving to the request for another client or a different request for the same client. The other strategies however, have the requests for thumbnails and expansions intermingled on each disk and therefore we expect that there will be more seeking on these disks.

We observed that the performance of the various strategies was sensitive to the access patterns that the clients exhibited. In particular, the Bundled and Separated strategies are able to exploit the contiguous placement of coefficients when there are large requests. In order to study the effect of variations in access patterns and other experiment parameters on the performance of the strategies, further experiments were conducted. These are discussed in the next subsection.

5.2 Effect of Varying the System Parameters

To investigate the influence of client access patterns, we ran another series of tests where the initial request for a single thumbnail was combined with the next request for the three closest neighbors yielding an access pattern that has only two phases for each query - one requesting four thumbnails and another requesting some number of expansions. Tests were run with the same expansion schemes as before. The results are shown in Figures 11 and 12.

Once again with no expansion, the relative performance of the various strategies does not change. With expansion, the performance of Interleaved and Bundled improves relative to Zonal. With higher levels of expansion or greater degrees of concurrency, both Bundled and Separated improve in performance relative to the other strategies. Separated still shows the peak at a concurrency of 2. All these results are consistent with the earlier results. However, as expected, with 8 clients, Separated performs much better than Bundled. Separated performs better than Bundled here because of the effect of concurrency and expansion on contention for the disks. Instead of generating four independent requests to the disks in the second phase, Separated generates a single

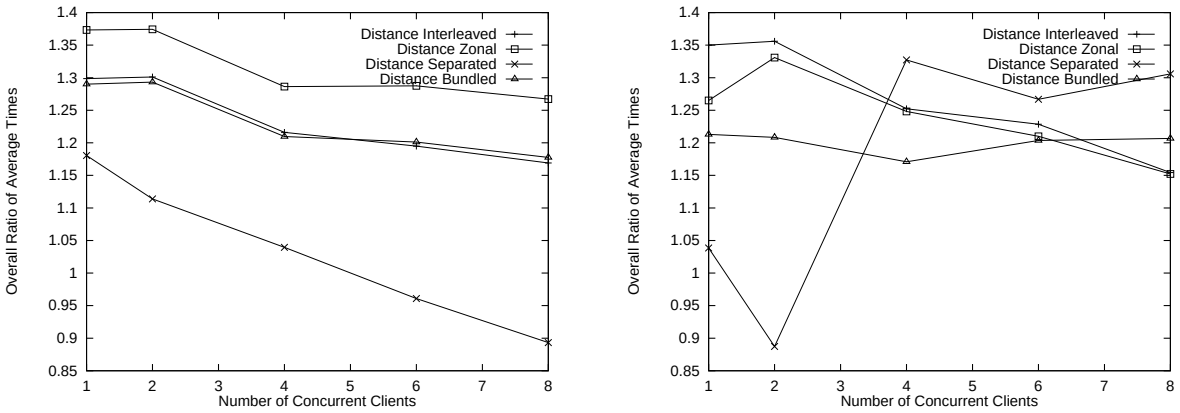

Figure 11: Distance based allocation with only two access phases with (a) 0% and (b) 25% expansion

coarser request to the fourth disk. This reduces the level of contention and the average queueing delays. Similar effects were also observed by Drapeau and Katz [DK93] in their experiments with striping in tape libraries. To test this hypothesis, we ran tests where the expansion of the thumbnails was requested one image at a time rather than as a set. As expected, the crossover point where Separated performs better than Bundled shifted in the direction of larger number of clients. Further details can be found in [Ano96].

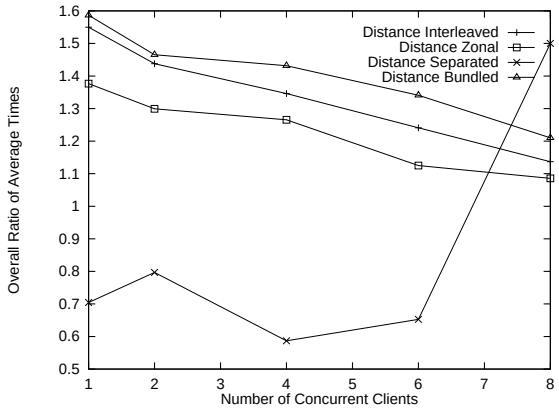

Figure 12: Distance based allocation with only two access phases with 100% expansion

To study the impact of the number of disks, the number of disks to was increased to eight. Also, the size of each thumbnail (and coefficient) was doubled so that the amount of disk used on each disk was the same as in the setup with four disks. The location files for each strategy were regenerated. The strategies used were natural extensions of those used for four disks. Therefore the maximum degree used in the declustering algorithm, d_{max} was seven instead of three, and two of the eight disks were reserved for the three coefficients of images under Separated placement instead of one disk. Similar experiments were run with different levels of expansion. The results for 100% expansion are shown in Figure 13. It is seen that Separated is not able to make the crossover that was earlier observed for 8 clients. Increasing the number of disks reduced the level of contention and limited the gains of the Separated strategy. This implies that the number of disks should be scaled with the degree of concurrency. The experiments of Drapeau and Katz [DK93] on tape libraries also bore out similar conclusions.

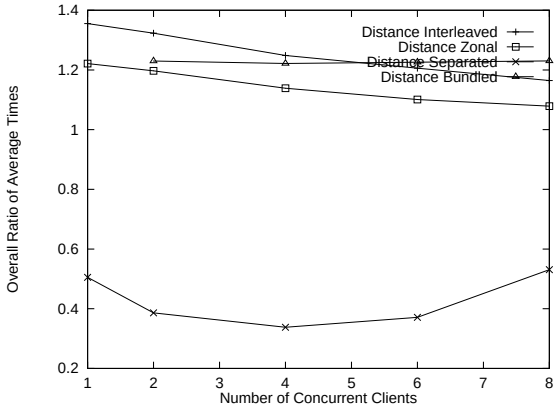

Figure 13: 100% expansion with Eight Disks

The influence of disk caching was also investigated by increasing the disk cache size to 128 Kilobytes - the maximum size for the disks simulated. Tests were run with an expansion factor of 25% using both the two phase and original three phase query modes. The results show that Bundled and Separated perform better than the other two placement strategies. These strategies perform better with larger caches because the disks that were simulated have read ahead and therefore, most of the items that are to be retrieved are very close to each other, hence they are read into the cache and are available when they are needed. Zonal and Interleaved do not have as many cache hits because the location of items that are accessed by any one client are usually on different disks, hence read ahead is not effective. Further details of these experiments can be found in [Ano96].

5.3 Discussion

In all the experiments, we observe that the Distance based strategies perform better than the corresponding Round Robin strategies (not shown for clarity). From this we conclude that it is always beneficial to decluster the images based upon their content. The performance of the four placement

strategies suggests that there is no clear winner across all expansion levels and degrees of concurrency. For low concurrency, the strategies that decluster the image coefficients perform better and for high concurrency the strategies that do not decluster coefficients perform better. It is interesting to note that Keeton and Katz [KK95] observed similar results for multiresolution video data. In particular, they found that declustering coefficients or subbands for video resulted in better performance only for low or moderate concurrencies.

8 Results with Real Disks

In order to validate the simulation results, some representative experiments using actual disks were conducted. We first describe the experimental setup and then the results.

8.1 Disk Setup

The experiments were conducted using four Seagate ST410800W disks, connected to an SGI Challenge server. On each of the four disks, a physical partition that consisted of the outer half of the disk was used to hold the data for the experiments. For the purpose of our experiments, it was necessary to control the placement of the data on the disks. This is not possible if the normal Unix file system is used to store the data on the disks. Therefore, in order to control the placement of the data, the disks were accessed through the raw device interface provided in Unix. The raw device is accessed by opening the file that corresponds to the raw partition. These files normally reside in the /dev/rdsk/ directory. Operations of read, write and seek when performed on such a file cause the disk driver to perform the corresponding operations on the physical disk. Another advantage of accessing the disk through the raw device interface is that operating system buffering is bypassed and I/O calls on the file result in actual physical operations on the disks. Note however that the on-disk cache was not disabled.

One drawback of our setup was that the disks were not exclusively available for our use. Although half of each disk was reserved for our data, one of the four disks was also used to hold other data. Consequently, access to this data during the same period in which our tests were running would affect our measurements since one of the disks would have to service requests that were not part of our experiments, resulting in extra seek times. Though we could not completely eliminate this problem, we attempted to minimize such effects by running the tests during periods of low activity from other users. Also, several sets of experiments were conducted and confidence intervals were generated to ensure that external factors did not affect our conclusions.

8.2 Results

The experiments were conducted using the 50,000 Landsat image data described in the previous section. Experiments with different levels of expansion were conducted. All experiments were found to be in close agreement with the simulation results. For reasons of brevity, all the results are not reported here. We describe only the results for 25% expansion. Due to the likelihood of random interference from other users, four sets of experiments with the same access patterns were conducted. These results were used to construct 95% confidence intervals. Figure 14 shows these results for the four Distance based strategies. It is important to note that

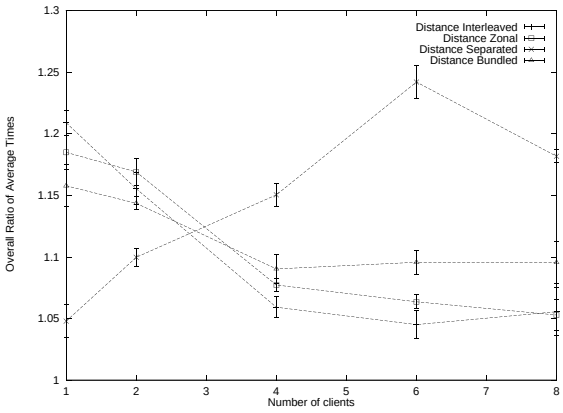

Figure 14: 95% Confidence intervals for 25% expansion with real disks

the confidence intervals do not overlap for most data points. The overlaps occur only when there is a crossover of the corresponding lines. From this we conclude that the measurements are reasonably reliable. Comparing Figure 14 with Figure 8(b), it can be seen that the two sets of graphs show similar trends. When the number of concurrent clients is low, the strategies that decluster coefficients perform better than those that do not. However as the number of concurrent clients is increased, the Separated strategy outperforms all other strategies with real disks. This is in accordance with the simulation results except that the crossover points are different. The drop in performance of the Separated strategy for even higher degrees of concurrency is also observed in both graphs, although to different degrees. The Bundled strategy shows similar behavior in both graphs too, improving in performance as the concurrency increases. In summary we note that the results of the tests with real disks support the earlier results that for low concurrency declustering of the coefficients is beneficial, however for high concurrency, placing the coefficients together results in better performance.

The differences between the simulation and the real disk experiments can be attributed to several factors. First, as mentioned earlier, the influence of other users and system activity on the disks could not be eliminated for the real disk experiments. Second, disk scheduling has been neglected in the simulations. In the simulation, requests are serviced in the order they are received at the server. However, the real disks do have some scheduling which affects the performance. Third, in the simulation we take an estimate of the time spent by each request waiting for the disk to service requests that are earlier in the queue. A request is assumed to incur a queueing delay equal to the sum of the disk times for requests that are ahead in the queue and are serviced after the given request is queued. For requests that were being serviced by the disk at the time the given request is queued, we take the waiting time to be half the disk time for the request. This estimate, though good, is approximate. Finally, in the simulation, we have a mix of simulated time (disk time) and real time (network time). In order to match these two times, we delay the simulated disk process by the time that the disk would have taken to process the request.

7 Concluding Remarks

We have evaluated the performance of nine placement strategies for wavelet decomposed multiresolution image datasets of various sizes, in the context of content based browsing under various levels of user concurrency. In all our experiments, the Distance based strategies consistently outperformed the Round Robin strategies which supports the conclusion that when content based retrieval is used to access images this information should be used for the placement of images on disks. In particular our results show that although declustering based upon image content is advantageous, the benefits of declustering the coefficients are largely dependent upon the degree of concurrency and the nature of the client access patterns. We showed that declustering the coefficients of multiresolution image data is beneficial under low concurrency (in particular for single users). Significant performance improvements were achieved with as few as four disks. These results are in agreement with conventional wisdom on declustering and striping of data. For higher levels of concurrency, however, the results that illuminating in that they highlight the complicated interplay between increased client concurrency and multiple disk parallelism. As concurrency is increased, not all declustering schemes result in improved performance. In particular, when it comes to coefficient placement strategies, there is no single strategy that outperforms all others over the entire range of expansion schemes and concurrency levels. The best strategy is dependent on the nature of access. For low concurrency and low expansion levels, Interleaved is the best. For high concurrency and high expansion levels, Separated is the best strategy. The Zonal strategy is only of interest if no expansion takes place, in which case only thumbnails need to be stored. Bundled is the strategy of choice for high expansion levels and low concurrency. We can therefore conclude that although declustering based upon image content is advantageous, the benefits of declustering the coefficients is largely dependent upon the degree of concurrency and the nature of the client access patterns. It is important to note that although we studied content based retrieval and image decomposition using wavelets, our placement techniques are applicable in a much wider domain. In particular, our techniques will result in gains for applications where access is made based upon any measure of similarity and objects are stored as collections of sub-objects.

In this study we have limited ourselves to a single level of wavelet decomposition. There is a need to extend the placement strategies to more levels and also to evaluate the performance of these strategies. In our future work, we hope to explore these issues and to develop more general placement strategies for wavelet decomposed images.

References

[AMEM95] A. D. Alexandrov, W. Y. Ma, A. El Abbadi, and B. S. Manjunath. Adaptive filtering and indexing for image databases. In *Proc. of the SPIE Int. Conf. on Storage and Retrieval for Image and Video Databases - III*, pages 12–23, San Jose, CA, February 1995.

[Ano96] Anonymous. Browsing and placement of multiresolution images on secondary storage. Technical Report TRCS96-22, Dept. of Computer Science (A brief abstract of this paper will appear in the proceedings of an IEEE conference. Details omitted to facilitate blind review.), 1996. http://www.cs.ucsb.edu/TRs/TRCS96-22.ps.

[AS95] S. Akyurek and K. Salem. Adaptive block rearrangement. *ACM Trans. on Comp. Systems*, 13(2):89–121, May 1995.

[BG88] D. Bitton and J. Gray. Disk shadowing. In *Proceedings of the Int. Conf. on Very Large Data Bases*, pages 331–338, Los Angeles CA., September 1988.

[CABK88] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data placement in Bubba. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 99–108, Chicago, 1988.

[Cas96] K. R. Castleman. *Digital Image Processing.* Prentice-Hall, Inc., 1996.

[CK93] T. Chiueh and R. H. Katz. Multi-resolution video representation for parallel disk arrays. *ACM Transaction on Multimedia*, pages 401–409, 1993.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill MIT Press, 1990.

[CP90] P. M. Chen and D. A. Patterson. Maximizing performance in a striped disk-array. In *Proc. of the 17th Int. Sym. on Comp. Architecture*, pages 322–331, 1990.

[DK93] A. L. Drapeau and R. H. Katz. Striping in large tape libraries. In *Proc. of Supercomputing*, pages 378–387, Portland, Oregon, 1993. ACM.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. In *Journal of Intelligent Information Systems*, volume 3, pages 231–262, 1994.

[GD90] S. Ghandeharizadeh and D. J. DeWitt. A multiuser performance analysis of alternative declustering strategies. In *Proc. Int. Conf. Data Engineering*, pages 466–475, Los Angeles, California., February 1990.

[GHW90] J. Gray, B. Horst, and M. Walker. Parity striping of disc arrays: Low-cost reliable storage with acceptable throughput. In *Proceedings of the Int. Conf. on Very Large Data Bases*, pages 148–161, Washington DC., August 1990.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[KK95] K. Keeton and R. H. Katz. Evaluating video layout strategies for a high-performance storage server. *Multimedia Systems*, 3:43 – 52, 1995.

[KSR94] D. Kotz, T. B. Song, and S. Radhakrishnan. A detailed simulation of the HP 97560 disk drive. Technical Report TR94-220, Comp. Sci. Dept., Dartmouth College., 1994.

[LSR92] J. Li, J. Srivastava, and D. Rotem. CMD: a multidimensional declustering method for parallel database systems. In *Proceedings of the Int. Conf. on Very Large Data Bases*, pages 3–14, Vancouver, Canada, August 1992.

[MM94] W. Y. Ma and B. S. Manjunath. Pictorial queries: Combining feature extraction with database search. Technical Report CIPR 94-18, Univ. of California, Santa Barbara, November 1994.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, and P. Yanker. The QBIC project: Querying images by content using color, texture and shape. In *Proc. of the SPIE Conf. 1908 on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–187, February 1993.

[OS95] V. E. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, pages 41–48, September 1995.

[PGK88] D. A. Patterson, G. Gibson, and R. H. Katz. A case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 109–116, Chicago, 1988.

[RW94] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, March 1994.

[SF95] T. R. Smith and J. Frew. Alexandria digital library. *Communications of the ACM*, 38(4):61–62, April 1995.

[SWZ94] P. Scheuermann, G. Weikum, and P. Zabback. "Disk Cooling" in parallel disk systems. *Bulletin of the Technical Committee on Data Engineering*, 17(3):29–40, September 1994.