Browsing and Placement of Multiresolution Images on Secondary Storage[*]

Sunil Prabhakar Divyakant Agrawal Amr El Abbadi
Ambuj Singh Terence R. Smith
Department of Computer Science
University of California
Santa Barbara, CA 93106

August 21, 1996

Abstract

With the 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 of images are being proposed for the storage organization of images. Image decomposition techniques such as *wavelets* can be used to provide multiple resolution images. The image is represented by several coefficients, one of them with visual similarity to the original image but at a lower resolution. Thus, these visually similar coefficients can be thought of as the *thumbnails* or *icons* of the original image. This paper addresses the problem of storing these multiresolution coefficients on disk(s) 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 browsing pattern of the images is used in the placement of image coefficients. Disk simulation is used to evaluate the performance of these strategies. The data used in the simulation are of two types: the texture samples from the Brodatz collection as well as 50,000 real landsat images. The results indicate that significant performance improvements can be achieved with as few as four disks by placing image coefficients based upon the browsing access patterns.

1 Introduction

With the 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

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

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 at UC Santa Barbara [SF95] 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 related to the 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 useful for users who are not interested in any particular set of images and would like to determine items of interest by viewing some or all of the images available in the system. When browsing, a user would typically view large numbers of images, most of which may not be of much interest to the user. 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 to the user. Thus 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. The Chabot project is an example of such a system. 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 four 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 the *thumbnail* or *icon* of the original image. The transformation can be carried out recursively to several levels without incurring any additional storage cost. Browsing on these thumbnails reduces the transmission of unnecessary images. Also, the original image can be obtained by carrying out image reconstruction recursively in the reverse direction. Another advantage of wavelet decomposition is that it allows progressive delivery of the images.

In order to develop and evaluate placement strategies it is necessary to have a model of the user access patterns. Since traces of actual user access patterns for image repositories are not available, we developed approximate models of the user access patterns. In a browsing scenario, we expect that users accesses will be directed by similarity between images. For example, users 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 also be a more complex relationship involving several characteristics such as image content, date of publication and geographical location. For the purposes of out 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 image processing techniques described in Section 2.

This paper addresses the problem of storing the wavelet coefficients on disk(s) 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. These strategies can be classified into two broad classes depending on whether the content of the images is used in the placement of image coefficients. Disk simulation is used to evaluate the performance of these strategies. Experiments with real disks are also conducted to verify some of the simulation results. The data used in the simulation is of two types. The first set of experiments use 1,856 images based on synthetic texture images taken from the Brodatz album [Bro66]. The second set of experiments were conducted on 50,000 landsat images from the Map and Imagery Library at UC Santa Barbara. We conducted experiments both in single user as well as in multi-user environments. The results from single user environment confirm the conventional wisdom that declustering improves performance when multiple disks are available. We demonstrate that if content based retrieval is used to access the images then content information should also be used for the placement of images on the disks.

Furthermore, declustering the coefficients of multi-resolution image data is also beneficial in this setup. However, interpreting the results of various declustering policies in a multi-user setup is more challenging and often contradicts conventional wisdom. This is particularly the case since the vast majority of data placement and data allocation schemes are not tested in a multi-user environment. Thus, the interplay between parallel disk I/O versus sequential or streamed I/O on a busy disk is not clearly evaluated. In our latter setup when multiple clients are present, we conclude that not all declustering schemes result in better performance. We show that declustering based on image content is still beneficial but declustering image coefficients results in degraded performance at higher concurrency (i.e. with large numbers of clients) and higher expansion levels. Overall we observe significant performance improvements with as few as four disks when multiple clients are present. It is important to note that although in our experiments 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 any measure of similarity based upon which user accesses are made and any image decomposition scheme that generates an icon and other components necessary for reconstruction of the original image.

The rest of the paper is organized as follows. In Section 2 we discuss the wavelet decomposition process and content based retrieval. Section 3 discusses related work on data placement on disks. Section 4 describes the placement strategies that were evaluated. Section 5 reports the experimental setup and results for the experiments with the 1,856 texture images. Section 6 presents the setup and results for the 50,000 landsat image dataset. Section 7 presents and discusses some experiments with real disks. Section 8 gives the final conclusions that we draw from the results.

2 Content Based Browsing and Retrieval

When browsing, a user would like to see many images in a short period of time. All the images seen may not be of interest to the user, hence it is not very important to show the images in full detail when displaying images for browsing. The images themselves can be quite large, and there is little to gain in spending the user's time as well as valuable resources in displaying large high resolution images that are not of interest to the user. It is better to summarize a large image into a small thumbnail which retains visual similarity to the image at a lower resolution. During a browsing session, if a user finds an image to be of interest, he/she may initiate a request to view the image at a higher resolution. While browsing, users will in general retrieve new images based on the images that have been already examined. This type of interaction can only be supported by incorporating a content-based search and retrieval mechanism. In this section, we first discuss wavelets to store multiresolution representations of images and then briefly describe some image processing techniques to summarize the contents of images.

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.

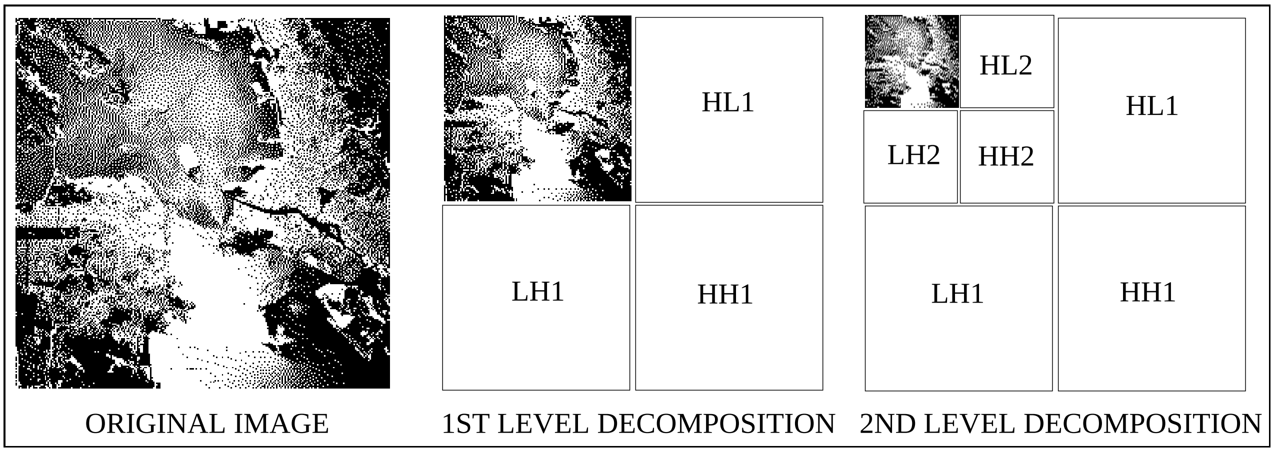

Figure 1: The Wavelet Decomposition Process

Hence the Euclidean distance between a pair of images gives a quantitative measure of the degree of similarity of their image contents [AMEM95].

3 Related Work

Data placement has been an important research topic in the context of database systems [AS95, BG88, CABK88]. The technique of striping data across multiple disks to overcome the I/O bottleneck has been extensively studied [PGK88, CP90]. The improvements in performance are essentially because large requests are serviced in parallel by multiple disks and multiple small requests are handled concurrently. Extensive studies of both the performance and fault tolerance aspects of RAID have been conducted. In [GHW90] for example, a variation of RAID where only the parity and not the data is striped across disks is shown to yield lower response times and higher throughput.

Further improvements in performance have been achieved when the structure of the data and the access patterns are well understood. This is the case in the relational database setting where the data is structured in tables and accesses are via range or partial match queries. Various schemes have been proposed to improve the performance of these queries. In [AE93] Abdel-Ghaffar and El Abbadi develop provably optimal allocation algorithms for partial match queries. In [GD90], three alternative declustering schemes are compared for shared-nothing multiprocessor database machines. Another multidimensional declustering method, called Coordinate Modulo Distribution (CMD), is developed in [LSR92].

When the structure of the data or the access patterns are not well understood, other approaches have been taken to improve I/O performance. In these cases data is generally viewed as independent files. In [AS95] a technique which dynamically copies blocks that are frequently accessed to a reserved space in the center of the disk to reduce seek delays is described. The notion of "Disk Cooling" is introduced in [SWZ94], where data is dynamically redistributed based on the access patterns to yield higher I/O performance. Data placement and declustering in highly parallel systems is studied in [CABK88]. The authors show that the static data placement problem is NP-complete. They introduce the notion of the heat of a disk and use it to develop a greedy heuristic for data allocation on disks.

The application studied in this paper does not fall into the RDBMS category since the image structure is not as well understood as the relational model, and therefore the optimizations for this model cannot be applied. However, we do know that the data has structure, since each image is decomposed into meaningful parts. The coefficients of an image are intimately related, and we have some understanding of the access patterns for browsing and multiresolution image access. Hence we can potentially do better than the more general approaches taken above. In a similar setting, Chiueh and Katz [CK93] discuss pyramidal coding schemes for multiresolution video data and a storage layout for this data 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 each disk for better performance. This work is not directly applicable outside the context of video data. The Chabot project currently underway at UC Berkeley [OS95] is also concerned with the retrieval of multiresolution images (although each image is stored at different resolution levels and thus incurs additional storage overheads when compared to wavelets). The allocation strategy followed in Chabot is to place all thumbnails and metadata about thumbnails on disk using POSTGRES [SRH90], 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. Another similar project, Query by Image Content at IBM Almaden [NBE+93, FBF+94] handles querying of images by color, shape and texture. It does not however 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.

4 Placement Strategies

The objective of our experiments was to 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 was also considered where no declustering of images is done. The thumbnails and their coefficients are placed randomly 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 would expect if no meaning is attached to the thumbnails and their three coefficients.

The two image allocation strategies listed below were evaluated.

1. *Distance based allocation*

2. *Round Robin allocation*

Distance based allocation refers to an allocation 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 should 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, it was felt that an approximation to the optimal solution would be acceptable. Hence, the following algorithm was 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.

We begin by determining which disk each thumbnail is to be allocated to. Starting with an initial unallocated node, it is arbitrarily assigned to a disk. Next, all its neighbors are assigned to the other disks in a round robin fashion, in increasing order of distance from the image. We then proceed in a breadth-first manner allocating images that have not yet been allocated. Note that since we have restricted the number of edges for each image, the graph may become disconnected. Once all the images that are connected to the initial node are allocated, we pick a new unallocated thumbnail and repeat the process until all images are allocated. In this manner, the disk on which each thumbnail is to be placed is decided. The allocation of the other three coefficients and the placement of the thumbnail and coefficients on each disk is determined by the placement strategy used.

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 of the other three coefficients and the placement of all items allocated to each disk is determined by the placement strategy used, as in the Distance based allocation strategy.

The allocation strategies determine which thumbnails are to be stored on which disk. They also determine the order in which this allocation is done. On the other hand, the placement strategies determine on which disks the remaining three coefficients of each image will be stored. Under both allocation strategies, four placement strategies were tested:

1. *Interleaved*
2. *Zonal*
3. *Separated*
4. *Bundled*

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.

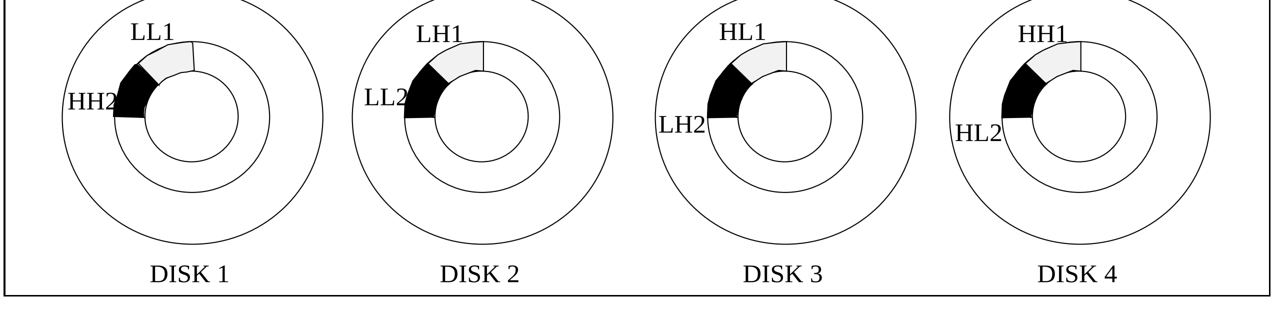

Figure 2: Interleaved Placement Strategy

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.

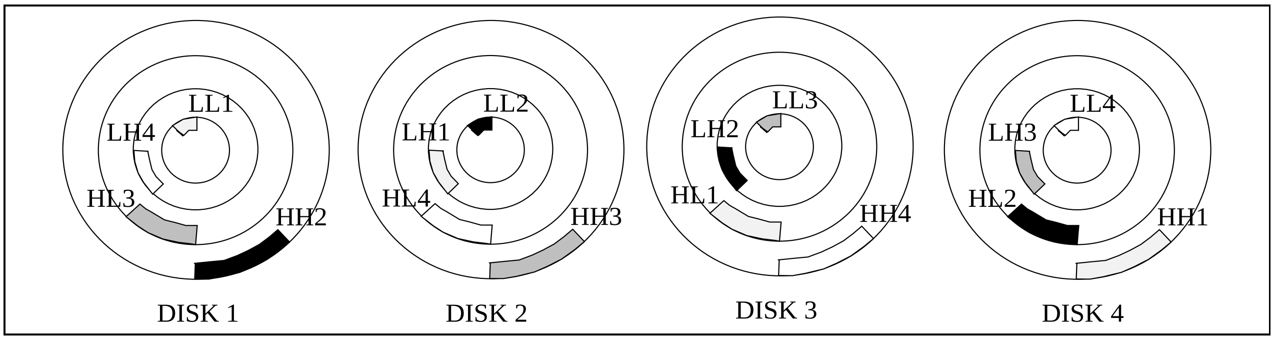

Figure 3: Zonal Placement Strategy

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 the results for this placement strategy.

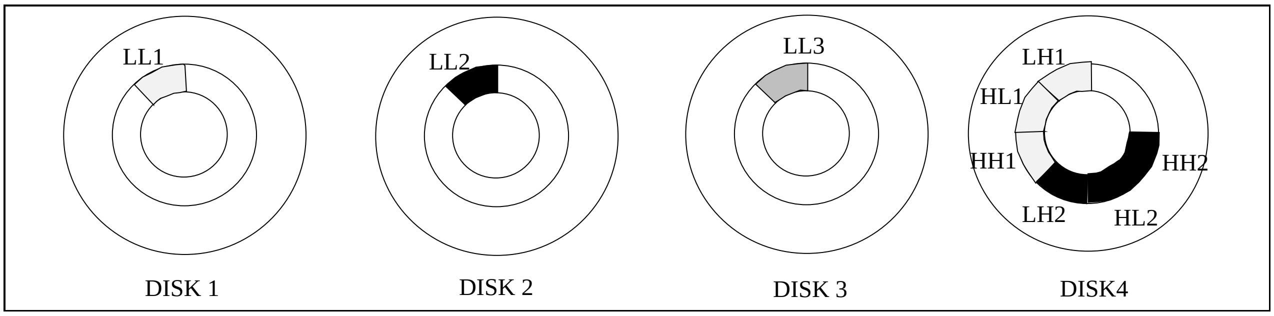

Figure 4: Separated Placement Strategy

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

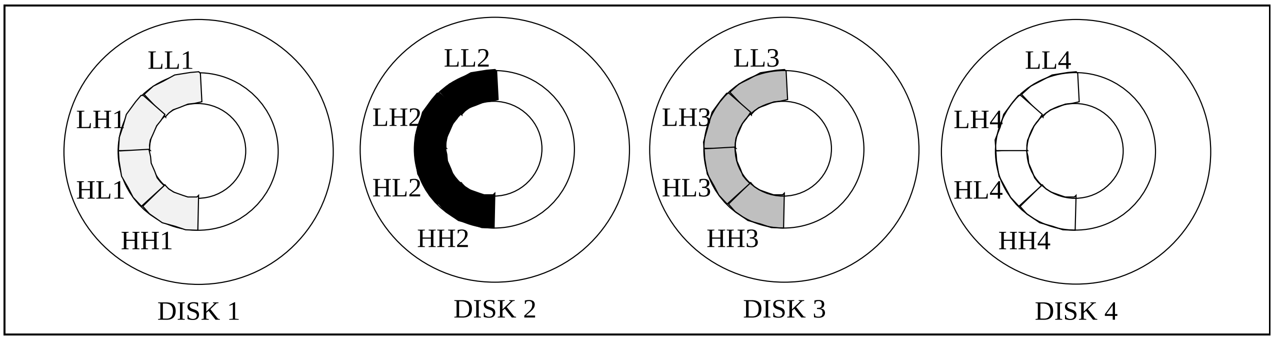

Figure 5: Bundled Placement Strategy

5 Texture Images

5.1 Experimental Setup

The concept of a digital library is modeled using a client-server architecture, in order to simulate as closely as possible the intended interactions in Alexandria. The server represents the digital library that maintains the holdings and services clients request. The client is representative of users spread across a wide area network that make requests to the server for images. In our study, the data for the images is not actually stored on disk, but rather a disk simulator is used. Use of a simulator provides much greater flexibility and also reduces development time that would be needed to write the necessary disk driver routines in order to control the placement of data on a real disk. Once placement strategies have been studied, confirmation of the results will be done using real disks. The simulator that we used 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. For the experiments conducted on the Texture images, we worked with only a single user. In reality we expect there to be multiple users which access the library but these users will be divided into groups with similar interests. The net accesses of such groups will show an overall pattern that we approximate by a Gaussian distribution over images. In other words, we expect that user groups will access some small number of images more often than others.

The client process therefore represents not a single user but rather a group of users with skewed access patterns. We approximate this behavior by using a Gaussian distribution with a standard deviation of 20 images. Furthermore, the number of accesses for the most frequently accessed image is 50. We ensure that the Gaussian distribution is not related to the sequence in which the images were ingested into the system.

The browsing access pattern used in our experiments for a typical user group can be summarized

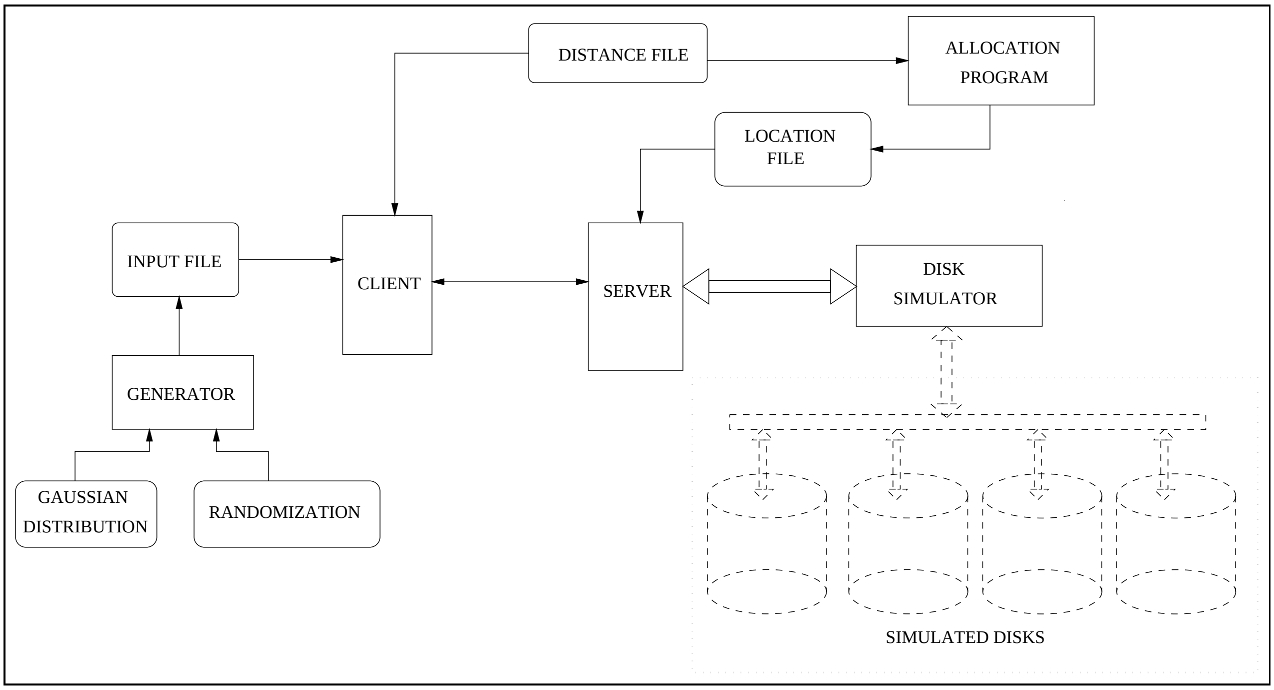

Figure 6: Schematic view of the Experimental Setup

as follows. The user first accesses an initial image according to the Gaussian distribution. Each request is sent to the server which results in the retrieval of the thumbnail of the image and is delivered to the client. Next, the user requests a set of images that are closest to the initial query image, following which the user could proceed to the next image or choose to view higher resolution images of some or all of the retrieved set. 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 disk(s).

The server receives requests from the clients and services them. It is connected to the disk simulator which evaluates the time required to access items from the various disks. In the current setup, four disks were used, all connected to the same I/O bus. The use of four disks allows us to investigate inter-disk parallelism for seeking. The modeled disk is 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 allocation programs using the various strategies that are to be evaluated. The location file has information about the placement of each coefficient of each image.

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

[a] A Short seek is incurred when the difference of cylinders (dcylinders) is less than 383.

Table 1: Table of Disk Parameters for HP 97560

5.2 Results and Analysis

The dataset for the experiment consisted of 1,856 texture images from the Brodatz album[1] [Bro66]. We decomposed each image 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 distances among these vectors were computed to determine the 20 closest neighboring images for each image. The closest 20 images were sufficient because it was felt that the user will not want to be overloaded with a large number of images while browsing.

For each strategy, the coefficients accessed were the same, i.e., the input to the client was generated only once and the same input was used for all tests on this dataset for fairness of comparison. The experiments were conducted using different levels of expansion, i.e., a fixed percentage of images in each query was expanded to full resolution. The number of neighbors accessed was 1, 2, 4, 8, 12, and 16. For each test, the average disk access time for each retrieved image was calculated.

The Random placement of images was used as a reference for comparing the performance of each placement strategy for the four expansion schemes. An aggregate figure of performance for each test was obtained by taking the average of the ratio of average access times for each item in the Random case to that for the respective strategy over all items accessed, using the same set

[1] Actually the Brodatz album consist of approximately 100 texture images. These images were supplemented with other textures to a total of 116. Each of the images was divided into 16 sub-images [MM94].

size. Therefore for each expansion scheme, each strategy and each set size, an aggregate figure was computed which gives the average ratio of access times of the Random placement strategy to the respective strategy. A ratio of 1 means that the placement strategy is as good as the Random strategy, and a higher ratio means that the strategy is better than the Random strategy for that expansion scheme and set size.

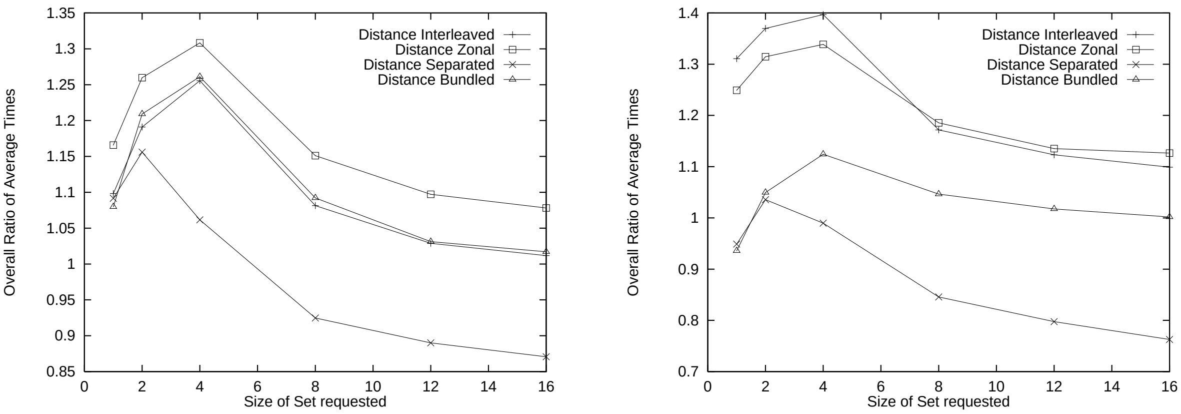

Figure 7: Distance based allocation with (a) 0% and (b)25% Expansion for 1,856 Images

First we analyze the performance of the four placement strategies, viz., Interleaved, Zonal, Separated and Bundled. Figures 7(a) and 7(b) report the results of the simulation of the four Distance based placement strategies. In each graph, the x-axis represents the query set size that was used and the y-axis gives the average performance ratio. The figure on the left shows the results for the no expansion scheme and the figure on the right shows the results under 25% expansion.

Figure 7(a) illustrates the results for no expansion for Distance based allocation. In this case, the Zonal placement strategy performs the best since only thumbnails are accessed and all thumbnails are placed on 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 others (except Separated). Hence the average seek time is much smaller which results in the better performance of the Zonal placement strategy. However, when expansion is introduced, we see that the Interleaved placement strategy begins to outperform the Zonal especially for small set sizes, as seen in Figure 7(b). This is because with expansion, the Zonal placement strategy is compelled to seek over larger distances as it retrieves all four types of items from all four zones, which is not necessary for the Interleaved placement strategy. For large query sets, though, both the Zonal and Interleaved placement strategies perform almost similarly. This is because the likelihood that at least one of the four disks in the Interleaved placement strategy will incur a large seek is now much higher, hence both placement strategies are equally likely to make large seeks. For query set sizes smaller or equal to 8, the Interleaved placement strategy performs better, and for set sizes larger than 8, the Zonal placement strategy performs marginally better. This cutoff is largely dependent on the number of disks that were used. With larger numbers of disks, we expect that the Interleaved placement strategy will outperform the Zonal placement strategy for even larger set sizes.

If we compare the Interleaved placement strategy with the Bundled placement strategy, we see that when we do not expand, the two perform almost identically as shown in Figure 7(a). 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 we do not access these coefficients, the performance is almost identical. With expansion, Bundled placement strategy performs worse than the Interleaved placement strategy (Figure 7(b)). The reason for this is that the degree to which the Interleaved placement strategy is able to exploit inter-disk parallelism is more than the Bundled placement strategy when the three coefficients are accessed for reconstructing the image. It is important to note that the Bundled placement strategy always performs worse than the Interleaved placement strategy which indicates that declustering the thumbnail and its three coefficients over the disks results in better performance.

The Separated placement strategy performs worse than the Random placement strategy for most query set sizes. For the case where we have no expansion, 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 Random case, especially for large set sizes where the Random placement strategy is able to get significant parallel I/O. With expansions, the performance is even worse for the Separated placement strategy. This is because of the above reason and also because the Separated placement strategy cannot gain from any parallel I/O for the access of the three coefficients of expanded images. The only cases where the Separated placement strategy outperforms the Random placement strategy is when there are few expansions, and the query set size is small. This is because under these circumstances the Random placement strategy is unable to achieve substantial parallelism.

From Figures 7(a) and 7(b), we see that all strategies show an increase in performance when the query set size is increased from 1 to 4. This increase is a result of the gains in parallelism that result from increased query set sizes because the items in a set are accessed together. Hence strategies which decluster images will have more opportunity to achieve inter-disk parallelism. The Random placement strategy is least likely to have good declustering, hence it gains the least which is why all the other strategies gain more relatively (the ratios are all relative to the Random placement strategy). When there is at least some expansion there is no one placement strategy that consistently outperforms all others over the entire range of the query set. The two possible candidate strategies are Distance Interleaved and Distance Zonal. The Distance Interleaved placement strategy performs better for small query set sizes (smaller than 8), whereas there is only a marginal difference between the two at larger query set sizes. The relative performance of these four strategies under different expansion levels (50% and 100%) is the same as under 25% expansion.

The relative behavior of the four placement strategies under Round Robin allocation is almost

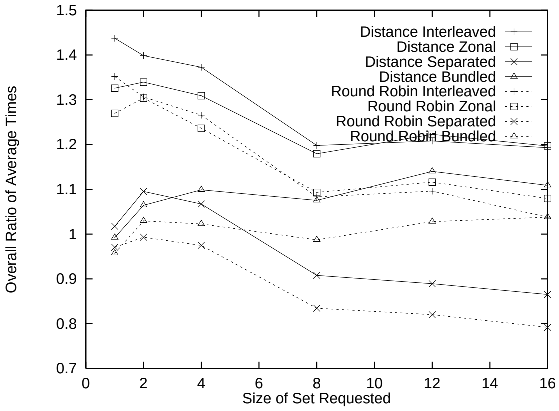

Figure 8: Results for 25% Expansion for 10,000 Images

identical to that under Distance based strategies. Due to the small number of images, the differences between Distance based and Round Robin based allocation strategies were not significant. Therefore, in order to compare the relative performance of these allocation strategies, we repeated the experiments with a larger set of images. For this purpose we used 10,000 landsat images and processed them in the similar manner as the 1,856 Brodatz images and ran the same tests. Figure 8 shows the performance of the four placement strategies under both Distance based and Round Robin allocation for 25% expansion. We see that each placement strategy performs better under Distance based allocation than under Round Robin allocation. The same results were obtained for 0%, 50% and 100% expansion, and are not shown here. These experiments were also repeated with the cache size set to the maximum size for the HP97560 disks, viz. 128 Kilobytes. The qualitative results obtained with this cache size were similar to those with a small cache size and are therefore not reported here.

6 Landsat Images in Alexandria

6.1 Experimental Setup

After evaluating the performance of the placement strategies in a single user environment, we extended our experimental setup to incorporate multiple clients. In addition to increasing the number of clients, we also made other enhancements to the setup. We increased the number of images from 1,856 to 50,000 landsat images in order to study the scalability of the placement strategies. The access pattern for each client was changed from a predetermined set of items to a dynamically determined, configurable set. Each client executed a number of queries (typically 1000), each of which consisted of requesting a thumbnail, then its nearest neighbors and finally, depending upon the expansion level, requesting the expansion of some of these thumbnails. The choice of the initial item in each query was 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 were considered to be "hot", and these items were accessed 80% of the time. The initial item for each query was determined randomly based upon this distribution. The choice of how many neighbors to request and what fraction of these to expand was made dynamically for each query based on certain probability distributions specified for each client independently.

In the extended setup, we modeled multiple disks using one disk per disk simulator process instead of multiple disks within one disk simulator process as before. 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 to the simulator, the simulator cannot be accessed until the request is completed. Thus for example, if the server dispatches a request for disk 1 and if 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 to the simulator, resulting in the same problem. The modified setup eliminates this limitation, resulting in greater parallelism. 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 queuing delays. Attempts to include the network delays were unsuccessful because this involved the use of the real system clock. The use of a real clock lead to erroneous results due to unpredictable context switches. When the server process is context switched, real time elapses, however, the server or disk processes do not progress. Hence the times reported were in error due to these context switches, and more importantly, they were unreliable, since the time that the process spends while waiting for the processor is largely dependent on the number of other processes running on the system, which was beyond our control. In order to avoid misleading conclusions, we did not use real times but rather used the times reported by the disk simulator. One has to be careful with such a setup because the network effects are in real time, however the disk simulator works in simulated time. We solved this problem by forcing the disk to delay until the real time elapsed was equal to the time that the simulator reported before sending the reply to the server. In this way, we omitted network times, but we included estimates of queuing times. The queuing 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 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 queuing model.

Although the Distance based declustering algorithm that was described in the previous setup was effective for the small image set of 1,856 images, this approach was not effective for larger image sets. In order to quantify the effectiveness of a declustering algorithm, we computed a figure of demerit. The measure that we used was a count of the total number of pairs of images, for which one of the two items was among the n nearest neighbors of the other item, and both were placed on the same disk. Here n was a parameter ranging from 1 to 10. This 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 we can find a coloring using $d_{max} + 1$ colors[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 such that it 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.

The graph corresponding to the declustering problem is complete since we can compute the distance between any two images. However, as noted earlier we can ignore most of the edges and consider only edges to nearest neighbors. The resulting graph is a directed graph since an edge from image a to image b may be present because image b is among the nearest neighbors of image a, however, image a is not among the nearest neighbors of image 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. Upon applying the above algorithm to this graph, we obtained a much greater degree of declustering than the algorithm used in section 5.1.

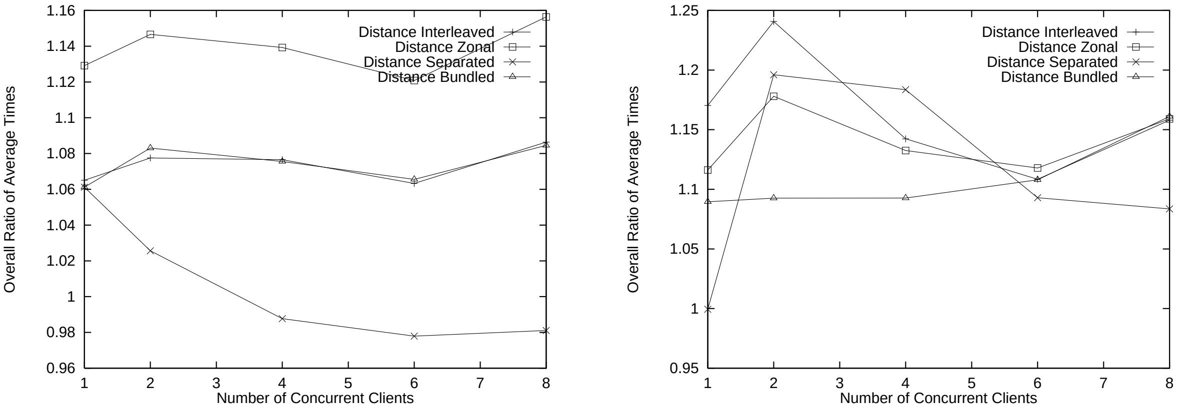

Figure 9: Distance based Allocation for 50,000 images with (a)0% and (b)25% expansion

6.2 Results and Analysis

Four sets of experiments were conducted for each placement strategy, each with a different level of expansion - 0%, 25%, 50% and 100%, as before. In order to study the performance of the strategies with different levels of user 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 picked an image at random using the "hot-cold" probability distribution. Each client in a concurrent set has a different seed for the randomization function. However, across different strategies, the clients used the same set of seeds so that the sequence of accesses would be the same for fairness of comparison. The setup allows a client to delay for some period of time after finishing one query and before starting the next. However, for the experiments reported here this delay, Think time, was set to 0. The ranges 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 [a]	0
Number of Disks	4 (8)
Thumbnail size	4KB (8KB)

[a] The time for which a client delays before sending the next query.

Table 2: Table of Parameters for Landsat experiments

As was the case with the texture images, the Round Robin allocation strategy performed worse than the Distance based strategy for each placement strategy. Hence, for reasons of brevity the results for Round Robin allocation are not presented for the landsat images. Figures 9(a) through 10(b) report the results for Distance based experiments. Figure 9(a) shows the results for the no expansion scheme, Figure 9(b) shows the results for the 25% expansion scheme and Figures 10(a) and 10(b) show the results for 50% and 100% expansion respectively.

Figure 9(a) illustrates the results for no expansion. The results for this scheme are qualitatively very similar to those for the Texture image set with no expansion as seen in Figure 7(a). In particular, the Zonal strategy performs best because we access only thumbnails, and these are located in a region of disk that is approximately one-fourth the size as compared to the other strategies. Hence Zonal incurs shorter seeks resulting in better performance. Interleaved and Bundled have very similar performance since they have the same degree of declustering of thumbnails. Their differences are due to the placement of the three coefficients of each image. Therefore, for no expansion their performance is almost identical. Separated performs poorly, even when compared with the random strategy. The reason for this is that the thumbnails are declustered across only three disks as opposed to four disks for every other strategy. Hence Separated is not able to exploit interdisk parallelism when only thumbnails are accessed and its performance degrades with increased concurrency.

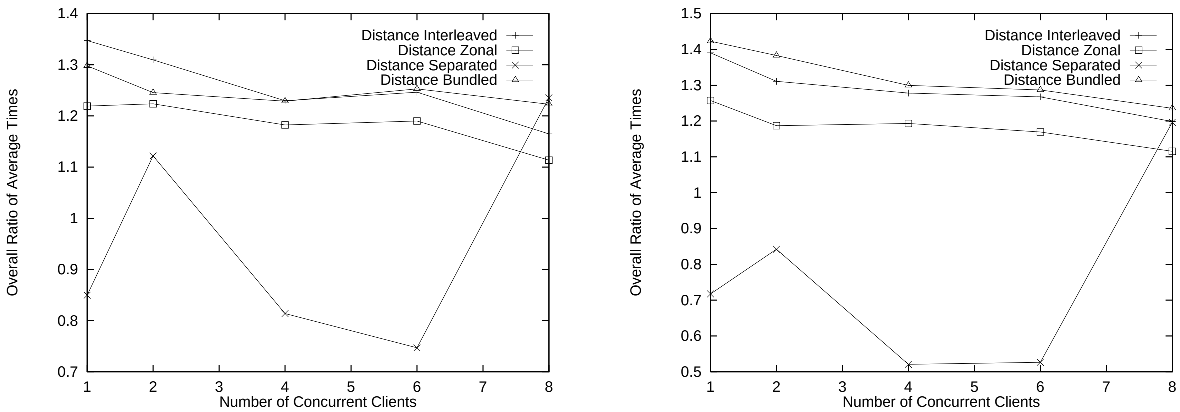

Figure 10: Distance based Allocation for 50,000 images with (a)50% and (b)100% expansion

In Figure 9(b) we see that as we begin expansion, Interleaved begins to outperform Zonal as before, 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. Interleaved also begins to outperform Bundled because the three coefficients are on three different disks as opposed to being on the same disk in Bundled. Hence, Interleaved is able to get parallelism in its accesses 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 10(a) and 10(b). We see that Bundled outperforms Interleaved at higher degrees of concurrency or greater expansion. This behavior was not seen in the single user scenario, where Interleaved clearly outperformed Bundled whenever any amount of expansion was done. 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. This behavior was not seen with the 1,856 dataset because the smaller number if images resulted in shorter seeks on the average, hence Interleaved did not lose much advantage. However, when the dataset size was increased to 10,000 images, this behavior was observed, as seen in Figure 8 where Bundled outperforms Interleaved for a set size of 4.

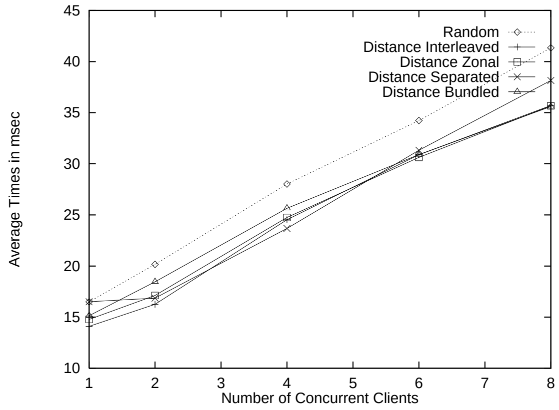

Figure 11: Actual average access times for 25% expansion

In order to understand the behavior of the Separated strategy, it is more instructive to consider Figures 10(a) and 10(b) before we try to explain the behavior seen in Figure 9(b). In both graphs we see that Separated has a jump at a concurrency of 2, and then it drops and later picks up at a concurrency of 8. This apparently anomalous behavior can be explained if we consider in detail the access patterns. In each query, we first access an initial thumbnail, then access three other thumbnails and finally we access a number of the other three coefficients, where the exact number depends on the expansion scheme. This can be seen from Figure 11 which shows the average access times in milliseconds for the five strategies. For each strategy, as we increase the number of clients, we see that there is increased 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 that we see 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.

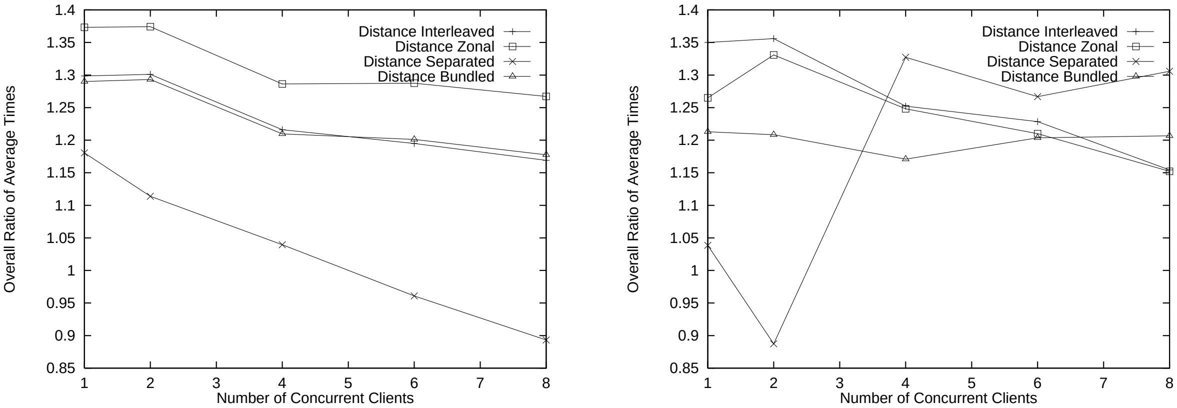

Figure 12: Distance based allocation with only two access phases with (a) 0% and (b) 25% expansion

As we move to a concurrency of 4, we see that this alternation is no longer possible and the performance of Separated drops 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 once again 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 we are required to retrieve the three coefficients for either 2 or 4 items which are closely related in terms of distance. Since these items would be placed close to each other, we will not need to incur large seeks 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 when we finish servicing the expansion request for one client and move 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.

Therefore, we observed that the performance of the various strategies was sensitive to the access patterns that the clients exhibited. In particular, we saw that 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, we conducted further experiments which are discussed in the next subsection.

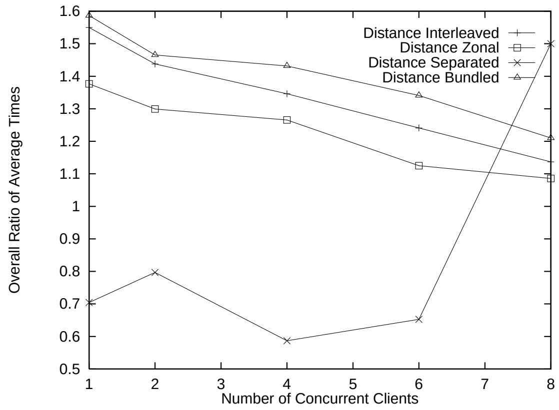

Figure 13: Distance based allocation with only two access phases with 100% expansion

6.3 Sensitivity Analysis

To investigate the influence of user access patterns, we ran another series of tests where we eliminated the initial request for a single thumbnail and combined that request 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 12(a) through 13.

Once again we see that with no expansion, the relative performance does not change. As we begin expansion, the performance of interleaved and Bundled improves relative to Zonal. With higher levels of expansion or greater degrees of concurrency, we see that 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 we expected, we see that 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 coarser request to the fourth disk. This reduces the level of contention and the average queuing delays. Similar effects were also observed by Drapeau and Katz [DK93] in striping experiments with 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. We expected that Separated should not perform better than Bundled in this situation. As expected, the crossover point where Separated performs better than Bundled shifted in the direction of larger number of clients.

To study the impact of the number of disks, we increased the number of disks used 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 six, 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 14. We see that Separated is not able to make the crossover that was earlier seen 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 on tape libraries also bore out similar conclusions.

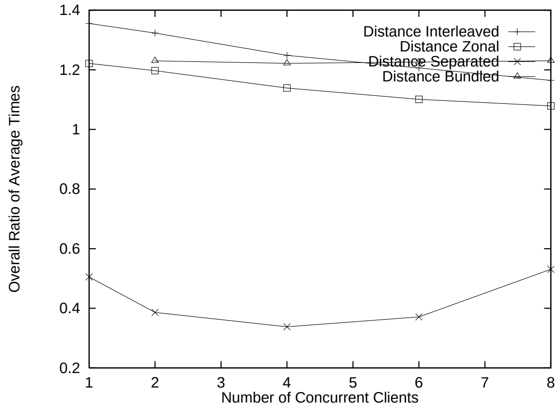

Figure 14: 100% expansion with Eight Disks

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 larges 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.

7 Results with Real Disks

In order to corroborate our findings based on simulation of disks, we conducted some representative experiments using actual disks. Due to the extent of real time involved in these tests, we were unable to repeat all our tests. We first describe the experimental setup and then the results.

7.1 Disk Setup

The experiments were conducted using four Seagate ST410800W disks, connected to a 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 filesystem 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 lseek 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 we did not disable the on-disk cache.

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.

7.2 Results

We conducted experiments 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, we do not report all the results 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 15 shows these results for the four Distance based strategies. If we compare this with Figure 9(b), we see that the two graphs show similar trends. When the number of concurrent clients is low, we see that the startegies that decluster coefficients perform better than those that do not. However as the number of concurrent clients is increased, we observe that with real disks the Separated strategy outperforms all other strategies. This is in accordance with the simulated 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 startegy shows similar behaviour in both graphs too, improving in performance as the concurrency increases.

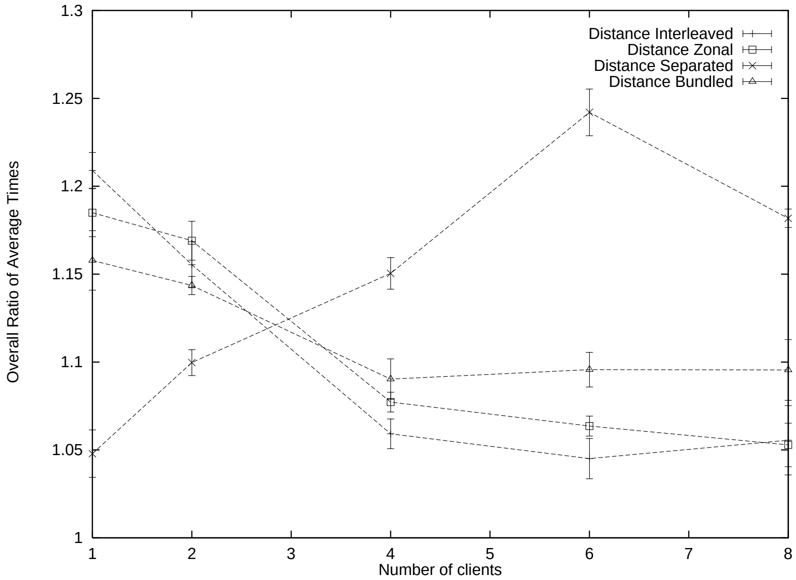

Figure 15: 95% Confidence intervals for 25% expansion with real disks

It is important to note that the confidence intervals in Figure 15 do not overlap in 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. In particular, the confidence intervals for the separated strategy have no overlaps and are much smaller than the performance differences (for example for 6 clients). 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.

8 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 single user as well as multi-user environments. We showed that when content based retrieval is used to access images then this information should be used for the placement of images on disks. We also showed that declustering the coefficients of multi-resolution image data is also beneficial in this setup. 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. Under the multi-client environment, however, the results that we obtained are illuminating in that they highlight the complicated interplay between increased client concurrency and multiple disk parallelism. We showed that as concurrency is increased, not all declustering schemes result in improved performance. In particular, we showed that Interleaved and Zonal do not perform as well as Bundled and Separated when we have more than six clients. We can therefore conclude that although declustering based upon image content is advantageous, the benefits of declustering the coefficients is largely dependent upon the the degree of concurrency and the nature of the client access patterns.

References

[AE93] Khaled A. S. Abdel-Ghaffar and Amr El Abbadi. Optimal disk allocation for partial match queries. *ACM Transactions of Database Systems*, 18(1):132–156, March 1993.

[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.

[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.

[Bro66] P. Brodatz. *Textures: A Photographic Album for Artists & Designers.* Dover, New York, New York, 1966.

[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 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.

[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] Chris Ruemmler and John 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.

[SRH90] M. Stonebraker, L. A. Rowe, and M. Hirohama. The implementation of POSTGRES. In *IEEE Trans. on Knowledge and Data Eng.*, volume 2, pages 125–142, March 1990.

[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.