Proceedings of the 1996 NASA Conference on Mass Storage Systems, College Park, Maryland, September 1996.

Processing Satellite Images on Tertiary Storage: A Study of the Impact of Tile Size on Performance[i]

JieBing Yu and David J. DeWitt
Computer Sciences Department
University of Wisconsin-Madison
{jiebing, dewitt}@cs.wisc.edu

1. Introduction

In July 1997, NASA will begin to launch a series of 10 satellites as part of its *Mission to Planet Earth*, more popularly known as EOSDIS (for Earth Observing System, Data Information System). When fully deployed, these satellites will have an aggregate data rate of about 2 megabytes a second. While this rate is, in itself, not that impressive, it adds up to a couple of terabytes a day and 10 petabytes over the 10 year lifetime of the satellites [KB91]. Given today's mass storage technology, the data almost certainly will be stored on tape. The latest tape technology offers media that is both very dense and reliable, as well as drives with transfer rates in the same range magnetic disk drives. For example, Quantum's DLT-4000 drive has a transfer rate of about 3.0 MB/sec (compressed). The cartridges for this drive have a capacity of 40 GB (compressed), a shelf life of 10 years, and are rated for 500,000 of passes [QUA95]**.** However, since tertiary storage systems are much better suited for sequential access, their use as the primary medium for database storage is limited. Efficiently processing data on tape presents a number of challenges [CHL93]. While the cost/capacity gap [GRA94] between tapes and disks has narrowed, there is still about a factor of 2 in density between the best commodity tape technology (20 gigabytes uncompressed) and the best commodity disk technology (10 gigabytes uncompressed) and at least a factor of 4 in total cost ($2,000 for a 10 GB disk and $10,000 for a 200 GB tape library).

Raw data from a satellite is termed level 0 data. Before the data can be used by a scientist it must first undergo a number of processing steps including basic processing (turning the electrical voltage measured for each pixel in a image into an digital value (typically 8 or 16 bits), cleansing, and geo-registration (satellites tend to drift slightly between passes over the "same" area). The end result is a level 3 data product consisting of a series of geo-registrated images that an earth scientist can use for his/her research. Processing actually expands the volume of data collected by a factor of 2 or 3 and the original data received from the satellite is never deleted. Thus, the processing and storage requirements actually exceed the 2 terabytes/day figure cited above. As part of the EOSDIS project, NASA has contracted with Hughes to build such a system.

[i] This work is supported by NASA under contracts #USRA-5555-17, #NAGW-3895, and #NAGW-4229, ARPA through ARPA Order number 018 monitored by the U.S. Army Research Laboratory under contract DAAB07-92-C-Q508, IBM, Intel, Sun Microsystems, Microsoft, and Legato.

Once processed, the data is ready for analysis by an earth scientist. Analysis involves applying a series of algorithms (typically developed by the earth scientists themselves) to a large number of images in a data set. Frequently a scientist will be interested in a certain type of images for a particular region of the earth's surface over an extended period of time.

The focus of our research is how best to handle images stored on tape. In a random access environment, the tile size (tape block size) depends largely on the transfer speed and access latency (seek speed). For the current DLT technology, the tile size (tape block size) is expected to be in the range of 100 MB in order to achieve good average performance. However, we believe using tile size at such a big size may not be beneficial for certain kind of applications. For the purposes of this paper we make the following three assumptions:

1. Images that are of interest to a scientist are stored on the same tape.
2. Images are accessed and processed in the order they are stored on tape.
3. The analysis requires access to only a portion of each image and not the entire image. We term this portion of the image its ***clip region***. Typically, the analysis will clip the same portion of each image.

With regard to the first assumption, while the images from a single sensor will undoubtedly span multiple tapes, it makes little sense to mix images from different sensors on the same tape. Analysis requiring access to multiple tapes (for data from either the same or different sensors) can use the techniques described in [SUN94] to minimize tape switches in combination with the techniques described below. The second assumption requires that the reference pattern to the images be known in advance so that it can be sorted into "tape order." In some cases, this order can be determined by examining the meta data associated with the data set. In a companion paper [YD96] we show how a new tape processing technique that we call "query pre-execution" can be used to automatically and accurately determine the reference pattern. The third assumption is based on the fact that satellite images are quite large (the size of an AVHRR image is about 40 megabytes) and scientists are frequently interested in only a small region of a large number of images and not each image in its entirety. The EOSDIS test bed [EKD95] has also put a strong emphasis on providing real-time dynamic subsetting of AVHRR images.

There are two alternative approaches for handling tape-based data sets. The first is to use a *Hierarchical Storage Manager (HSM)* such as the one marketed by EMASS [EMA95]. Such systems almost always operate at the granularity of a file. That is, a whole file is the unit of migration from tertiary storage (i.e. tape) to secondary storage (disk) or memory. When such a system is used to store satellite images, typically each image is stored in a separate file. Before an image can be processed, it must be transferred in its entirety from tape to disk or memory. While this approach will work well for certain applications, when only a portion of each image is needed, it wastes tape bandwidth and staging disk capacity transferring entire images.

An alternative to the use of an HSM is to add tertiary storage as an additional storage level to the database system. This approach is being pursued by the Sequoia [DFG94] and Paradise [DKL94] projects. Such an integrated approach extends tertiary storage beyond its normal role as an archive mechanism. With an integrated approach, the database query optimizer can be used to optimize accesses to tape so that complicated, ad-hoc requests for data on tertiary storage can be executed efficiently. In addition, the task of applying a complicated analysis to a particular region of interest on a large number of satellite images can be performed as a single query [SFG93].

Integrating tertiary storage into a database system requires the use of a block-based scheme to move data between different layers of the storage hierarchy in the process of executing a query. While 8 Kbytes is a typical block size for moving data between memory and disk, it is too small to use as a unit of transfer between tape and memory or disk, especially when dealing with large raster images. The approach used instead by Postgres [SUN94] and Paradise [DKL94] is to partition each satellite image into a set of *tiles*. Tiles become the unit of transfer between tape and memory or disk while a smaller disk block (e.g. 8K bytes) is used to transfer data between disk and memory (i.e. the database system buffer pool). When a query references a portion of an image residing on tape, the meta data associated with the image is used to determine the minimum number of tiles necessary to satisfy the request. These tiles are first moved from tape to disk in tile-sized units and then from disk to memory in units of 8K bytes (or whatever page size the database system uses).[ii]

This paper examines the impact of tile size on the time required to retrieve a series of partial (i.e. clipped) images residing on tape. The evaluation includes a simplified analytical model, a simple simulation study to verify the analytical model, and a performance evaluation in the Paradise database system, extended to include support for tertiary storage [YD96]. Our results indicate that the careful selection of tile size can reduce the time required to clip a series of images residing on a single tape. In particular, we show for the state of the art tape drives such as the Quantum DLT-4000, a smaller tile size in the range between 32 KB and 1024 KB gives reasonable performance gain over larger tiles sizes for a variety of image and clip region sizes.

The remainder of this paper is organized as follows. Section 2 describes the problem and the derivation of the analytical model. Section 3 describes the simulation experiments and analyzes their results. Section 4 examines the impact of tile size under a variety of experiments using Paradise as a test vehicle. Section 5 contains our conclusions and discusses future work.

2. Analytical Model

In this section we describe a simplified analytical model to compute the time to clip a single raster image by a rectangular region. We assume that the tape head is positioned at the beginning of the image. This model is then used to study the effect of tile size on the time to clip a raster image.

2.1 Problem Description

As discussed in Section 1, *tiling* partitions an image into smaller, rectangular pieces that preserve the spatial locality among adjacent pixels. Figure 1 depicts three alternative partitioning strategies of a single image and their corresponding linear layout on tape. The top left rectangle represents an un-tiled image. The middle and right rectangles in the top row show the same image partitioned into 4 tiles and 16 tiles, respectively. Once an image has

[ii] Actually data cannot be moved directly between two mechanical devices such as tape and disk without first passing through main memory. Thus, a tile is first read by the tape controller into memory and then written to a tape block cache on disk. While one could read tiles directly into the database buffer pool, this approach tends to flood the buffer pool with large amounts of information, forcing more valuable information out.

been tiled, it is written onto tape one tile after another. The bottom portion of Figure 1 depicts how each of the three tiling alternatives is laid out on tape, assuming that tiles are placed on tape in a row-major fashion. While the choice of using a row-major layout may affect performance, we will demonstrate that tile size is the dominating factor, and not whether tiles are laid out on tape in a row-major or column-major order.

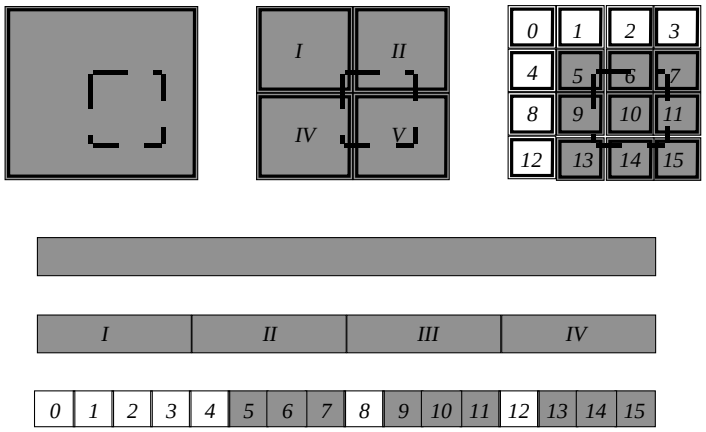

Figure 1: Single Image as 1, 4, 16 Tiles and their Linear Layout

The dashed rectangle in Figure 1 corresponds to the portion of the image that the analysis wishes to examine – what we term the ***clip region***. The shaded area indicates which tiles must be retrieved from tertiary storage in order to satisfy the clip request. The impact of tiling is best illustrated in the comparison between the un-tiled image (top-left rectangle in Figure 1) and the 16-tile image (top-right rectangle in Figure 1). For the un-tiled image, the entire image must be read from tape in order to process the clip request. On the other hand for the image that has been partitioned into 16 tiles, only 9/16ths of the image (9 of the 16 tiles) must be read. However, the number of seek operations increases from 0 to 3 assuming that the tape head is initially positioned at the start of the image.

In general, the use of tiling can reduce the amount of data that must be transferred when clipping partial images. On the other hand, it can introduce additional seek operations between consecutive tile accesses. The total time spent in migrating the necessary parts of the image to memory or disk depends on the *tape seek speed*, *the tape transfer speed*, and the *seek startup cost*. The *seek startup cost* is a fixed overhead associated with each tape head movement while the tape seek speed indicates how fast the tape head can be advanced when not actually read/writing data. Together, these two parameters determine the random access latency on a single tape. Besides these three parameters, there are a number of other factors that affect query performance. For example, consider the image in Figure 1 that was partitioned into 4 tiles. For the clip request shown in Figure 1, this partition strategy has no advantage with respect to the number of seeks performed or the amount of data transferred compared to the un-tiled image. Other clip requests would have different results if the clip region was entirely contained inside tile #2 of the 4 tile image. In this case, the un-tiled image would incur no seeks but would have to transfer the entire image. The image tiled into 4 pieces would incur one seek (to the start of tile 2) and would transfer ¼ of the image. The image tiled into 16 pieces would incur two seeks (one to transfer tiles 2 & 3 and a second to transfer tiles 6 &7) and would also

transfer ¼ of the image. Thus, both the size and location of the clip region can affect the performance of the various tiling alternatives. In order to better understand the problem, we developed an analytical formula to model the average-case behavior.

2.2 Model Assumptions

In order to reduce the complexity of the problem, the analytical model makes following assumptions:

1. Each tile is stored as a single *tape block*. A *tape block* is the unit of migration from tape to memory.
2. The tape head is initially positioned at the beginning of the image.
3. Images are square (e.g. 5 by 5 or 9 by 9 tiles but not 4 by 6 tiles).
4. The shape of the clipping region is proportional to the image shape, and the clipping region is always contained inside the image boundary.
5. The order in which clipped tiles are returned from tape is the same as the order they are stored on tape.

The first assumption eliminates the indirect effect of tape block size since multiple tiles could potentially be packed into a single tape block. Later in Section 3, we will examine the effect of this assumption. The second assumption allows us to concentrate on a single image without considering the residual impact from the previous tape head position. The third and fourth assumptions reduce the number of parameters that must be considered since variation in tile and clipping region shape may change the tape access behavior. This will be discussed in Section 5. The final assumption is to minimize the randomness between seeks within one clipping operation.

2.3 Analytical Formula Derivation

We model the response time to migrate the tiles containing the clipped region from tape to memory as the sum of the time spent in the following four operations: *Initial Seek*, *Intermediate Seeks*, *Tile Transfer*, and *Total Seek Startup*. Table 1 describes all the symbols used in the analytical model. Figure 2 exhibits some of the symbols in a graphical view. Note from the figure that each tile has unit length 1, and a is an integer while b may contain a fraction, which we modeled as $b - \lfloor b \rfloor$. Also it is obvious from the figure that the number of tiles covered or partially covered by the clip region varies depending on the particular position of each clip region. In the next sections, we will start by figuring out the probabilities for various clip region locations, then analyzing the times spent in all operations for different clip cases and finally give the final average case formula.

Description	**Symbol**	**Comments**
Image Size	M	Kbytes
Single Tile Size	t	Kbytes
Image Dimension	$a \times a$	$M = a^2 \cdot t$
Clip Dimension	$b \times b$	Clip Size $= b^2 \cdot t$
Clip Area/ Image Area	$1/c^2$	$a = c \cdot b$
Tape Seek Rate	S	Kbytes/Sec
Startup Seek Cost	I	Sec
Tape Transfer rate	R	Kbytes/Sec

Table 1: Parameters

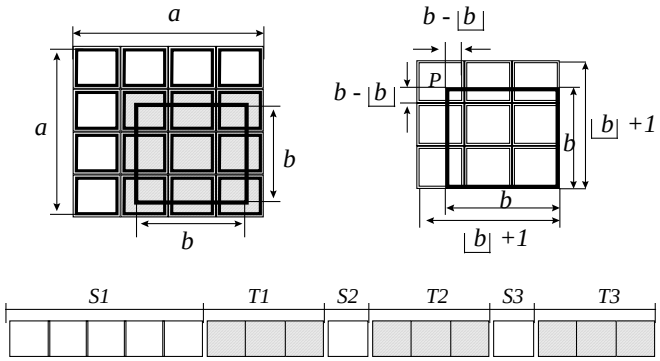

Figure 2: Target Clipping Region

Clipping Cases

Figure 3 illustrates the four cases that a clip region can be put on an image: Case 1 -- overlapping $(\lfloor b \rfloor + 2)^2$ tiles; Case 2 -- overlapping $(\lfloor b \rfloor + 2) \times (\lfloor b \rfloor + 1)$ tiles (elongated horizontally); Case 3 -- overlapping $(\lfloor b \rfloor + 1) \times (\lfloor b \rfloor + 2)$ tiles (elongated vertically); Case 4 -- overlapping $(\lfloor b \rfloor + 1)^2$ tiles. To find out the probability of each case under a uniform distribution, consider the upper left corner point (P) of the clip region. P has to reside in one of the tiles. For that particular tile, the areas that P can reside for each cases are illustrated in Figure 4. With the total area of regions (A) that P can possibly stay in the image being $(a-b)^2$, the probability for each case to happen can be derived by considering the number of tiles (F) that P can move to and the area (Af) that P can stay within one tile under each case. The probability comes as $Pb = F \cdot Af / A$.

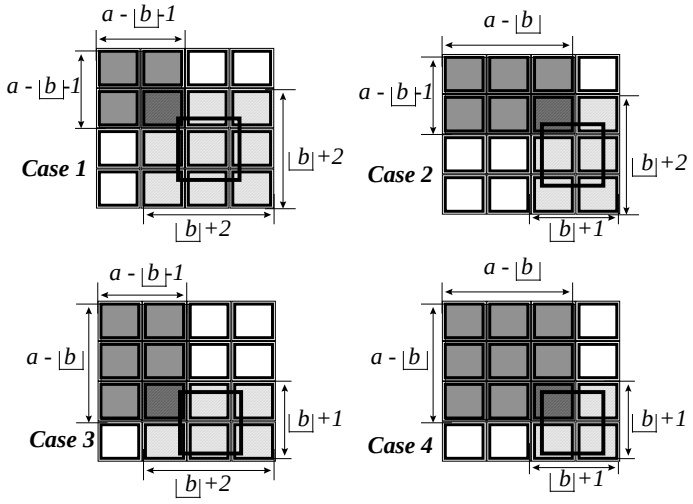

Figure 3: Four Cases for Clipping

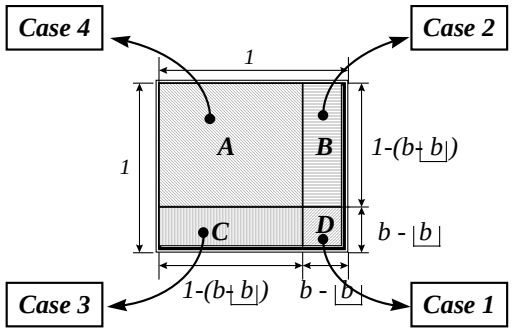

Figure 4: Areas for P under Different Cases

Below are the probabilities for each case.

Case 1: $Pf = (a-\lfloor b \rfloor -1)^2$, $Af = (b-\lfloor b \rfloor)^2$ (area *D*), then

$$Pb1 = (a-\lfloor b \rfloor -1)^2 \cdot (b-\lfloor b \rfloor)^2 / (a-b)^2 .$$

Case 2: $Pf = (a-\lfloor b \rfloor)\cdot(a-\lfloor b \rfloor -1)$, $Af = (b-\lfloor b \rfloor)\cdot(1-(b-\lfloor b \rfloor))$ (area *B*), then

$$Pb2 = (a-\lfloor b \rfloor)\cdot(a-\lfloor b \rfloor -1)\cdot(b-\lfloor b \rfloor)\cdot(1-(b-\lfloor b \rfloor))/ (a-b)^2 .$$

Case 3: $Pf = (a-\lfloor b \rfloor -1)\cdot(a-\lfloor b \rfloor)$, $Af = (1-(b-\lfloor b \rfloor))\cdot(b-\lfloor b \rfloor)$ (area *C*), then

$$Pb3 = (a-\lfloor b \rfloor -1)\cdot(a-\lfloor b \rfloor)\cdot(1-(b-\lfloor b \rfloor))\cdot(b-\lfloor b \rfloor)/ (a-b)^2 .$$

Case 4: $Pf = (a-\lfloor b \rfloor)^2$, $Af = (1-(b-\lfloor b \rfloor))^2$ (area *A*), then

$$Pb4 = (a-\lfloor b \rfloor)^2 \cdot (1-(b-\lfloor b \rfloor))^2 / (a-b)^2 .$$

Initial Seek Time (*Tf*)

The *Initial Seek Time*, ***Tf***, is the time to move the tape head from the beginning of the image to the first tile touched by the clip region (indicated as S1 in Figure 2). From the analysis above, we know that the upper left corner (*P*) of the clip region can only reside in a restricted area depending on the various cases. Suppose this area is *M* by *N* at the upper left corner of the image, then P has an equal probability of being in any of the *X* by *Y* tiles in this region. Assume that P is in the tile defined by row i and column j ($0 \le i < M$, $0 \le j < N$). Then, the number of tiles that must be skipped to reach P from the beginning of the image is $j + i \cdot a$. On average, $\left(\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(j+i\cdot a)\right)\Big/(M\cdot N)$ tiles are skipped to reach the first tile covered by the clip. Hence, $Tf = \left(\left(\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(j+i\cdot a)\right)\Big/(M\cdot N)\right)\cdot t \Big/ S$. Applying this to each of the 4 cases in Figure 3, we get:

Case 1: $M = N = (a-\lfloor b \rfloor -1)$, then

$$Tf1 = \left(\left(\sum_{i=0}^{a-\lfloor b \rfloor -2}\sum_{j=0}^{a-\lfloor b \rfloor -2}(j+i\cdot a)\right)\Big/(a-\lfloor b \rfloor -1)^2\right)\cdot t \Big/ S$$

Case 2: $M = (a - \lfloor b \rfloor)$, $N = (a - \lfloor b \rfloor - 1)$, then

$$Tf2 = \left(\left(\sum_{i=0}^{a-\lfloor b \rfloor - 1} \sum_{j=0}^{a-\lfloor b \rfloor - 2} (j + i \cdot a) \right) \Big/ \left((a - \lfloor b \rfloor) \cdot (a - \lfloor b \rfloor - 1) \right) \right) \cdot t \Big/ S$$

Case 3: $M = (a - \lfloor b \rfloor - 1)$, $N = (a - \lfloor b \rfloor)$, then

$$Tf3 = \left(\left(\sum_{i=0}^{a-\lfloor b \rfloor - 2} \sum_{j=0}^{a-\lfloor b \rfloor - 2} (j + i \cdot a) \right) \Big/ (a - \lfloor b \rfloor - 1)^2 \right) \cdot t \Big/ S$$

Case 4: $M = N = (a - \lfloor b \rfloor)$, then

$$Tf4 = \left(\left(\sum_{i=0}^{a-\lfloor b \rfloor - 1} \sum_{j=0}^{a-\lfloor b \rfloor - 1} (j + i \cdot a) \right) \Big/ (a - \lfloor b \rfloor)^2 \right) \cdot t \Big/ S$$

Finally, the *Initial Tile Seek Time* is given by :

$$Tf = Pb1 \cdot Tf1 + Pb2 \cdot Tf2 + Pb3 \cdot Tf3 + Pb4 \cdot Tf4 .$$

Intermediate Seek Time (*Ti*)

The *Intermediate Seek Time*, ***Ti***, is the total time spent seeking between transfers of contiguous sets of tiles contained in the clip region. For example, in Figure 2 above, after transferring the set of tiles in region T1, we must perform seek S2 before we can transfer the tiles in T2, and, after transferring the tiles in T2, we must perform seek S3 before transferring the tiles in T3. After the transfer of a set of contiguous tiles, the tape head must be moved to the next group of tiles affected by the clipping region. Assuming the tiles touched by the clip region form a X by Y region, then the number of tiles to be skipped over after the initial seek is $(a - X)$, and there are $(Y - 1)$ such movements. Thus, $Ti = (a - X) \cdot (Y - 1) \cdot t/S$. Applying this to all the cases in Figure 3, we can reach:

Case 1: $X = Y = (\lfloor b \rfloor + 2)$, then $Ti1 = (a - (\lfloor b \rfloor + 2)) \cdot (\lfloor b \rfloor + 1) \cdot t/S$;

Case 2: $X = (\lfloor b \rfloor + 2)$, $Y = (\lfloor b \rfloor + 1)$, then $Ti2 = (a - (\lfloor b \rfloor + 2)) \cdot \lfloor b \rfloor \cdot t/S$;

Case 3: $X = (\lfloor b \rfloor + 1)$, $Y = (\lfloor b \rfloor + 2)$, then $Ti3 = (a - (\lfloor b \rfloor + 1)) \cdot (\lfloor b \rfloor + 1) \cdot t/S$;

Case 4: $X = Y = (\lfloor b \rfloor + 1)$, then $Ti4 = (a - (\lfloor b \rfloor + 1)) \cdot \lfloor b \rfloor \cdot t/S$.

The final version of the *Intermediate Seek Time* is :

$$Ti = Pb1 \cdot Ti1 + Pb2 \cdot Ti2 + Pb3 \cdot Ti3 + Pb4 \cdot Ti4 .$$

Transfer Time (Tr)

Tile Transfer Time, *Tr*, is the total time spent transferring tiles in the clipped region to memory (*T1, T2* and *T3* in Figure 2). Based on the same assumption made in previous calculation for *Intermediate Seek Time*, $X \cdot Y$ tiles must be transferred . This leads to: $Tr = X \cdot Y \cdot t/R$. Again, analyzing the different cases in Figure 3 using this formula, we get:

Case 1: $X = Y = (\lfloor b \rfloor + 2)$, then $Tr1 = (\lfloor b \rfloor + 2)^2 \cdot t/R$;

Case 2: $X = (\lfloor b \rfloor + 2)$, $Y = (\lfloor b \rfloor + 1)$, then $Tr2 = (\lfloor b \rfloor + 2) \cdot (\lfloor b \rfloor + 1) \cdot t/R$;

Case 3: $X = (\lfloor b \rfloor + 1)$, $Y = (\lfloor b \rfloor + 2)$, then $Tr3 = (\lfloor b \rfloor + 1) \cdot (\lfloor b \rfloor + 2) \cdot t/R$;

Case 4: $X = Y = (\lfloor b \rfloor + 1)$, then $Tr4 = (\lfloor b \rfloor + 1)^2 \cdot t/R$.

The overall version of the *Transfer Time* is :

$$Tr = Pb1 \cdot Tr1 + Pb2 \cdot Tr2 + Pb3 \cdot Tr3 + Pb4 \cdot Tr4 .$$

Seek Startup Time (Ts)

Each tape seek is associated with a fixed startup overhead which we model via *Seek Startup Time*. This overhead only depends on the number of seeks performed but not the size of each seek operation. If we use the same $X \cdot Y$ region used above, then totally Y seeks are needed for each clip. Then, $Ts = Y \cdot I$. Breaking this into different cases in Figure 3, we get:

Case 1: $Y = (\lfloor b \rfloor + 2)$, then $Ts1 = (\lfloor b \rfloor + 2) \cdot I$;

Case 2: $Y = (\lfloor b \rfloor + 1)$, then $Ts2 = (\lfloor b \rfloor + 1) \cdot I$;

Case 3: $Y = (\lfloor b \rfloor + 2)$, then $Ts3 = (\lfloor b \rfloor + 2) \cdot I$;

Case 4: $Y = (\lfloor b \rfloor + 1)$, then $Ts4 = (\lfloor b \rfloor + 1) \cdot I$.

The overall version of the Seek Startup *Time* is :

$$Ts = Pb1 \cdot Ts1 + Pb2 \cdot sr2 + Pb3 \cdot Ts3 + Pb4 \cdot Ts4 .$$

Total Response Time

The *Total Response Time* is the sum of the above four times. Due to the complexity of the formula (the number of items involved even after simplification), we do not write down the entire formula here. Instead, we just list the parameters used in the function and rely on plotting figures to analyze the behavior of the formula. From all the components derived above, we know $T = Tf + Ti + Tr + Ts = f(a.b.S, R, I)$. To make the plot graph easier to understand, we substitute a, b with $a = c \cdot b$ and $M = a^2 \cdot t$, with M being the image size, t the tile size and c the clip selectivity. Now it becomes $T = f(M, t, c.S, R, I)$.

For analysis purposes, we fix M for the image size and c for the clipping area ($1/c^2$ of an image). Given this, as we will show through plotting graphs below, the formula reveals the following interesting property: as the tile size increases, the seek time (including *Tf*, *Ti*, and *Ts*) decreases while the transfer time (*Tr*) increases. The combination of these two opposite effects makes the response time a complex function of the tile size.

Analytical Model Analysis

To help understand the implications of the response time formula derived above, we next evaluate it for a variety of parameters and plot the response time curve as a function of the tile size. The values for the parameters are listed in Table 2. The image size is varied from 8 MB to 128 MB and the clip selectivity from 1/4 to 1/256 (relative to the image size). The tape-related parameters (*S*, *R*, *I*) are selected based on the Quantum DLT-4000 tape drive [QUA95] with compression turned off.

Parameter	Values Evaluated
M	8192, 32768, 131072 (Kbytes)
c	2, 4, 8, 16
S [iii]	2048 (Kbytes/Sec)
R	1356 (Kbytes/Sec)
I	0.1 (Sec)

Table 2: Selected values for parameters

Figure 5 to 7 show the response times for a variety image size and clip size combination and their relative gain over whole image fetching. The response time curves exhibit some kind of *diminishing on return* for the left region of the curves before they head off upwards steadily to the right. The relative gain curves demonstrates that smaller tile sizes tend to perform better than bigger tile sizes for all 3 image sizes. To take a closer look at why it is happening this way, we pick one of the curves and break the response time curve into 3 parts: *Seek, Transfer and Seek Overhead*.

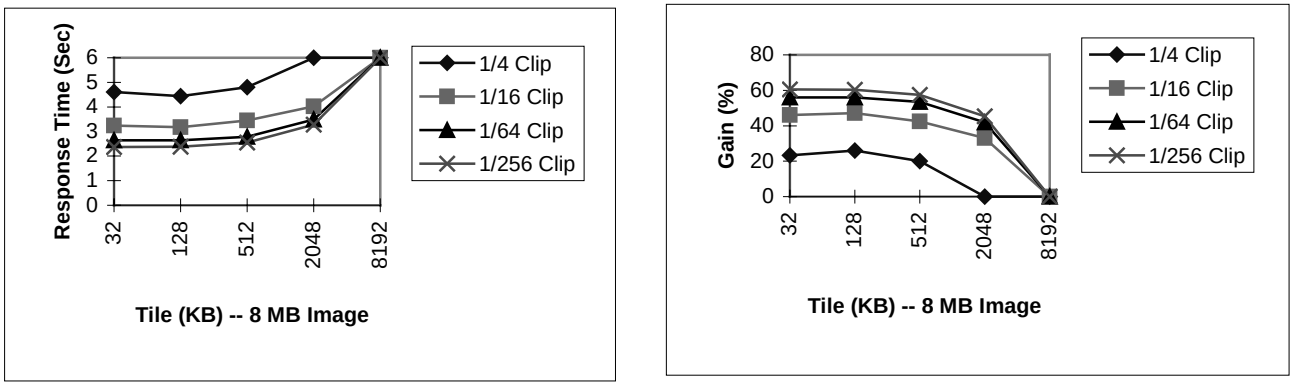

Figure 5: Response Time and Performance Gain over Whole Image Fetching for 8 MB Images

[iii] DLT uses serpentine tapes, seek time is not a linear function of seek distance for seeking between different tracks. It has an observed maximum of 180 seconds seek time (90 seconds in the technical specification) between two random blocks. But on a single track, seek time is close to a linear function. Accurate modeling of seek time for DLT drive can be found in [HS95].

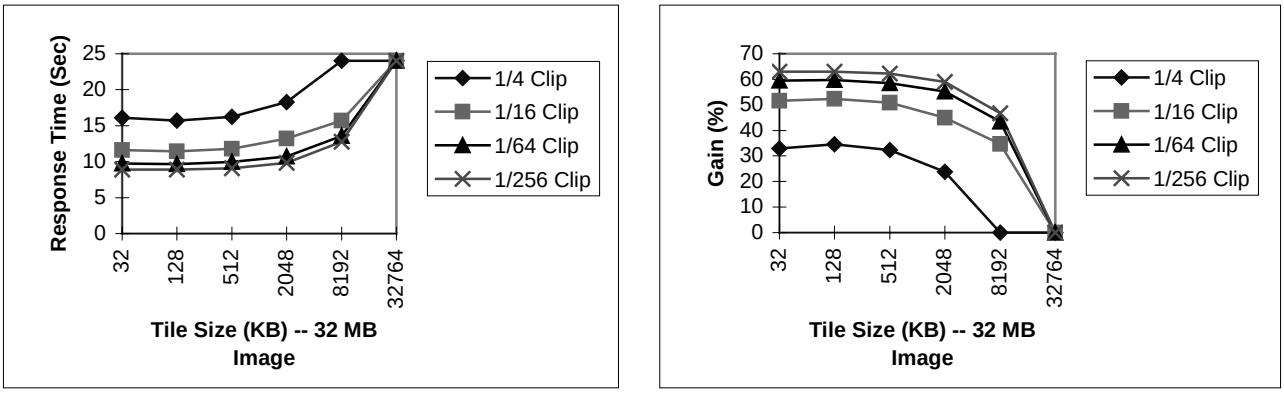

Figure 6: Response Time and Performance Gain over Whole Image Fetching for 32 MB Images

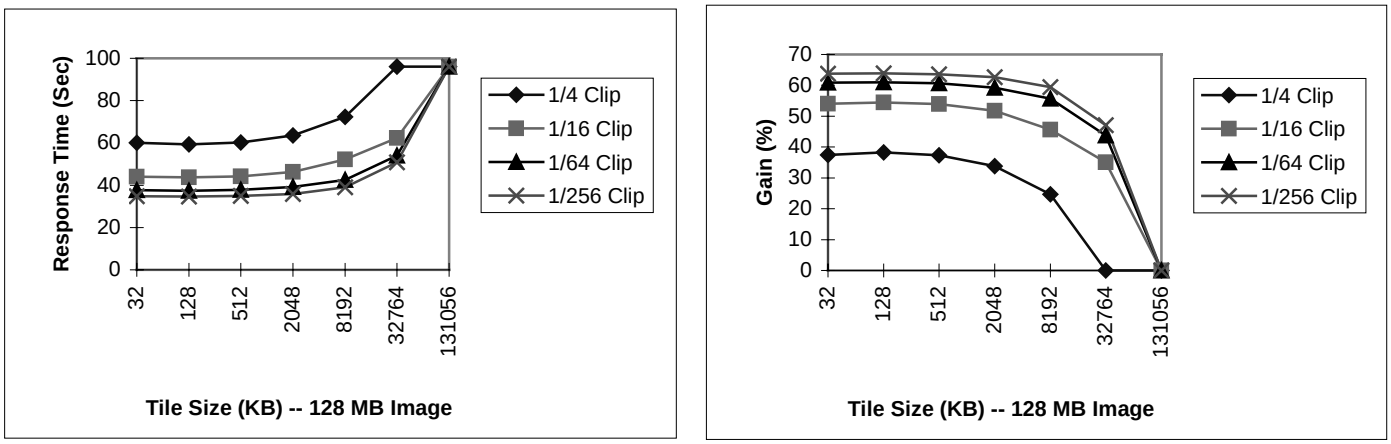

Figure 7: Response Time and Performance Gain over Whole Image Fetching for 128 MB Images

Figure 8 shows the breakdown of response time into its *seek, transfer and seek overhead* components when clipping $1/16^{th}$ of a 32 MB image. While both the seek time (*Tf+Ti*) and seek overhead (*Ts*) decrease as the tile size increases, the transfer time (*Tr*) increases at a faster rate and thus dominates the response time. The effect of tape seeks is best illustrated in the region to the left of a 512 KB tile size in this figure, where the reduction in seek time, resulting from both fewer and shorter seeks, offsets the increase in transfer time as the tile size increases. Based on these figures, for the DLT drive, tile size in the region between 64 KB and 512 KB are better choices for any sized images, where they yield between 60% and 70% reduction in total clip time over whole image fetching.

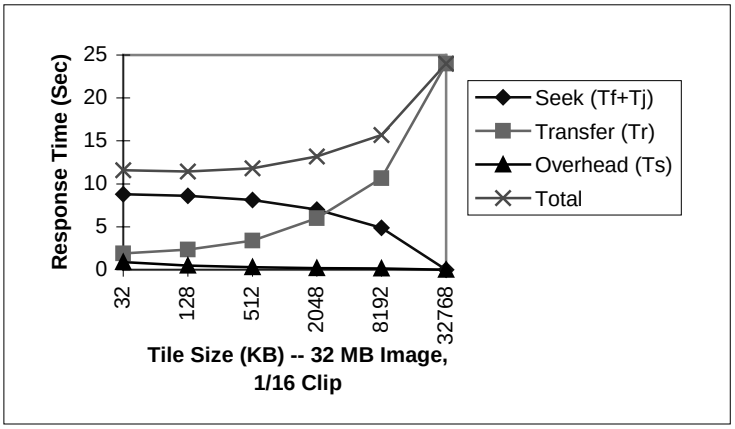

Figure 8: Breakdown of Response Time

3. Simulation Experiments

From the analytical model described in Section 2 there is a trade-off between decreasing the seek time and increasing the transfer time as the tile size is increased. The analytical model, however, was based on a number of simplifying assumptions so that it would be easy to derive and analyze. There may be conditions for which the results obtained using it are not valid. To verify its accuracy, we developed a simulation model which is used to explore a broader range of test conditions than possible with the analytical model.

3.1 Simulation Configuration

As with the analytical model, the simulation model focuses on the response time for transferring data from tape to memory. The simulator consists of three components: a *work load generator* for generating clip requests of various sizes, shapes, and locations, an *image clipper* for generating the sequence of tile accesses required to satisfy a particular clip operation, and a *simulated block-based tertiary storage manager*. Given a request for a tile, the tertiary storage manager simulator first converts the tile number to a tape-block address. Next, it simulates moving the tape head from its current position to the desired tape block and transferring the block from tape to memory. The tape parameters from Table 2 are used to model a Quantum DLT-4000 tape drive. Each result data point represents the average response time of 1000 clip requests at different locations. There are several major differences between the analytical and the simulation models. First, assumptions 3 and 4 from Section 2.1 are relaxed: images no longer must be square, the clip shape need not be proportional to image shape. In addition, the tape block size can be different than the tile size. That is, multiple tiles can be packed into a single tape block. We believe that this simulation model is general enough to capture any cases that might not be covered by the analytical model.

3.2 Analytical Model Verification

To verify the analytical model, we first repeated the experiments by using the same tile/tape block size and clipping shape and area as those used in analytical model analysis. For better comparison with the analytical result, we plot the relative difference data between response times from simulation and analytical results in Figure 9, 10 and

11. The difference margin is no more than 4% in any sized image clips. The response time generated by the simulation model is slightly lower than the analytical model in most cases (negative number in difference), which is due to the randomness of the clip region selection. This error margin is acceptable given the amount of runs we use in the simulation study. Based on these numbers, it is clear that the analytical model captured the exact behavior of clipping under those assumptions. This in turn confirmed all the finding that we made from the analytical model. .

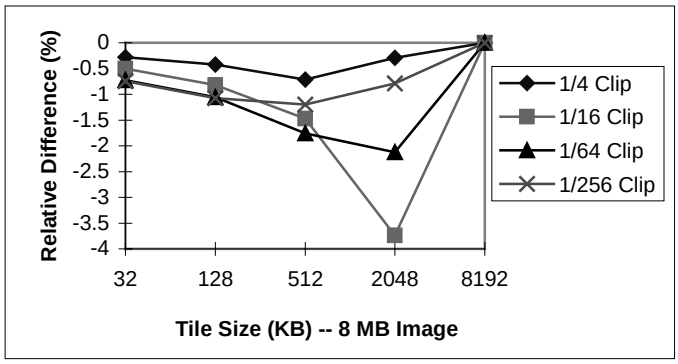

Figure 9: Relative Difference between Analytical and Simulation Result -- 8 MB Image

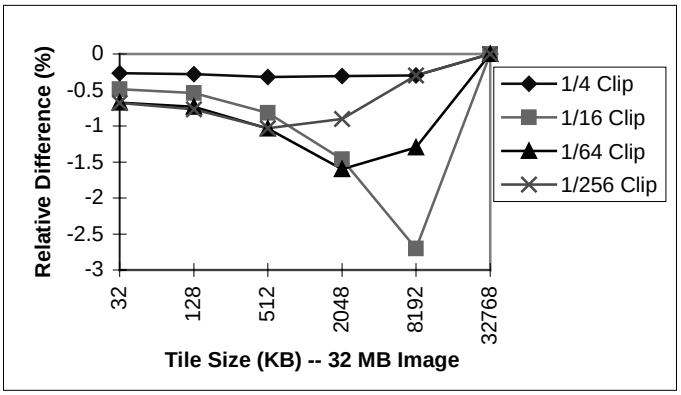

Figure 10: Relative Difference between Analytical and Simulation Result -- 32 MB Image

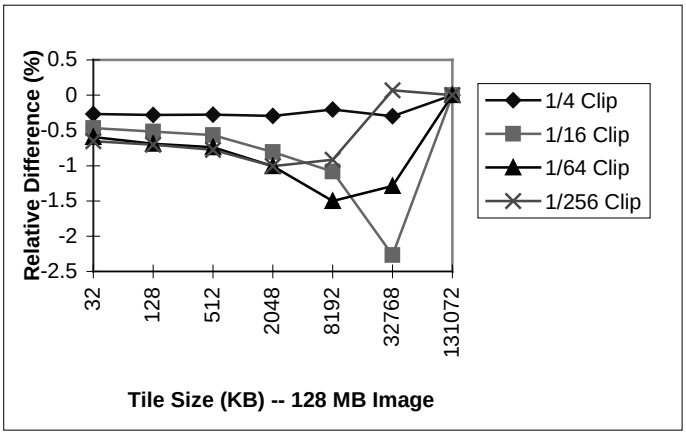

Figure 11: Relative Difference between Analytical and Simulation Result – 128 MB Image

3.3 Tape Block Size vs. Tile Size

As mentioned in Section 3.1, the simulation model allows us to examine the impact of varying the tile size and the tape block size independently. Using a tape block smaller than a tile will not affect performance much since each tile is then stored as a set of contiguous tape blocks. On the other hand, when the size of a tape block is larger than the size of a tile, multiple tiles can be packed into a single tape block. This can affect response time as fetching one tile is likely to result in reading tiles that are not covered by the clip operation. Figure 12 contains three curves corresponding to three different *Tape Block Size/Tile Size* ratios. Clearly, using a tape block size larger than the tile size causes performance to degrade. In general, packing multiple tiles into a single tape block has a similar effect to increasing the tile size. However, since packing tiles into a bigger tape block is usually in a row-major order, when too many tiles are packed into a tape block, the over-all shape of all the tiles in a tape block is not rectangular any more. This irregular shape can further degrade clipping performance. These results indicate that it is the best to use the same tile and tape block size. Consequently, all results presented in this paper use the same size for both tile and tape block.

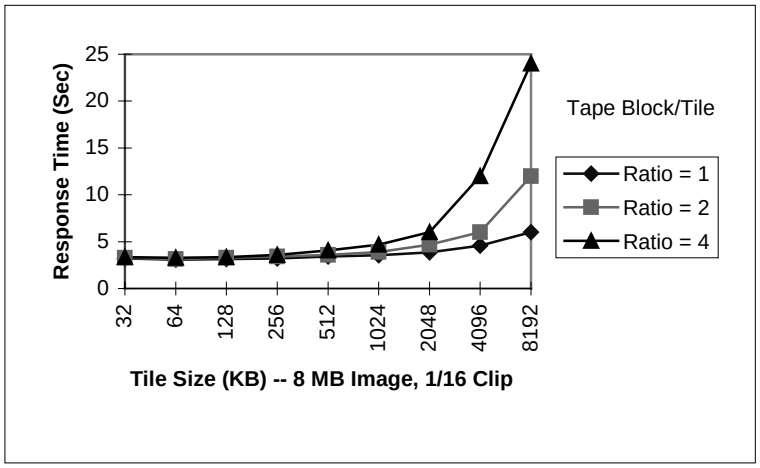

Figure 12: Effect of Tape Block Size vs. Tile Size

3.4 Varying Clipping Shapes

One of the assumptions made for the analytical model is that the shape of the clipping region must be proportional to the image shape. Clipping regions with the same area but different shapes can also affect performance. To investigate this effect, we experimented with 3 different clipping region shapes: *Long*, *Wide* and *Square*. *Long* is a thin rectangular shape whose height is 4 times its width; *Wide* is the *Long* shape rotated 90 degrees; *Square* is proportional to the image shape (which has been used in all previous experiments). Figure 13 shows the result from this experiment and illustrates that different shapes have different response times. It is interesting to notice that the *"Wide"* curve is consistently below the *"Long"* curve until the tile size increases to 4096 KB. This is caused by the the row-major linear ordering of tiles on tape. Such a layout helps *"Wide"* clips reduce the number of tiles that must be seeked over. The drop of the "*Long*" curve at the 4096 KB is due to the fact that the 8192 KB image is partitioned into two 4096 KB tiles along the x-axis. On average, only half of a 4096 KB tile must be skipped under the tile size of 4096 KB, while one and a half of 2048 KB tiles must be skipped under the 2048 KB tile size. In addi-

tion, this tile layout forces "*Wide*" shape clips read the entire 8192 KB image, and favors the "*Long*" clips via its column-major nature. The "*Square*" clips is relatively close to the average behavior. No matter how different the response times may differ between various clipping shapes, we still observe the same trend along each curve: the best response time is observed when tile/tape block size is in the range from 32 KB to 256 KB.

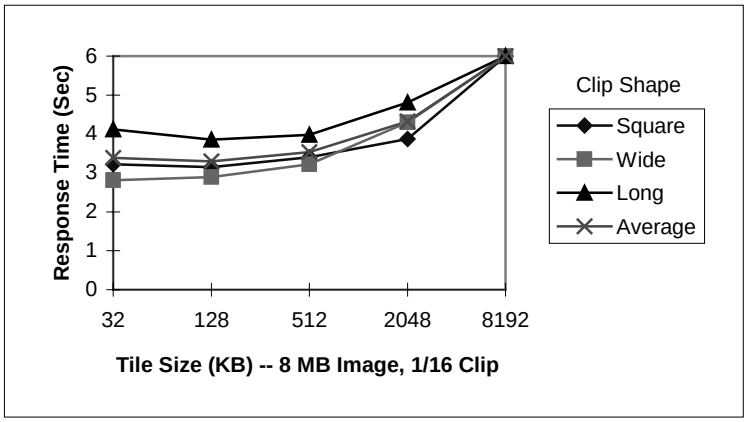

Figure 13: Varying Clip Shape, Simulation

4. Implementation Results

To further verify the previous results in more realistic environments, we chose to repeat some of the experiments in the following two configurations: direct application -- an user application program that accesses raster images directly on tapes, and Paradise -- a DBMS extended to handle data on tapes. The application program is a stand-alone program that is capable of accessing tiled raster image data on tapes. This is corresponding to typical scientific applications that access tape resident data set. Paradise [DKL94] is an object-relational database system developed at University of Wisconsin – Madison, which is capable of handling both spatial data (vector-based) and images (raster-based) efficiently. For raster images, Paradise combines tiling and compression for high performance on a disk based system. In a separate project [YD96], we have extended Paradise to include a tertiary storage manager that accesses data on a Quantum DLT-4000 tape drive. Both implementations share the same block-based tape device driver and the raster image clipping code. So the time involved in tape access and raster image clipping in memory is comparable. However, the application program directly transfers data from tape to user space without going through staging disk and main memory buffer pool, which can give us a realistic measurement for tape access time alone. While the experiments in Paradise represents the end-to-end performance for clipping raster images in a tertiary database system.

Experiments in both configurations were conducted on a DEC CELEBRIS XL590 (Pentium 90MHz) with 64 MB memory. The tertiary device used is the Quantum DLT4000 tape drive. The tape driver used for the experiment breaks any large block into multiple of 32 KB internal physical chunks before operating the actual tape I/O via ioctl. This scheme is used to simplify the physical tape record address mapping by ensuring each single chunk write

call actually generate a single physical record on the tape drive[iv]. A single logical tape block is mapped to multiple contiguous 32 KB physical tape records on tape. For results from Paradise, all queries were executed with cold buffer pools (for both main memory and disk cache). Due to the high cost of experiments in those realistic environments, we had to cut down the number of randomly generated clip shapes from 1000 (used in simulation result) to 20.

Figure 14 shows the result from the application program for the same clip shape experiment in Figure 13. As you can see, the two figures are very close to each other. The general trend of the curves are the same, except that the absolute numbers for the application program under all tile sizes are consistently higher than simulation numbers (about 1 sec difference for smaller tile sizes, and 0.3 sec difference for lager tile sizes). This is caused by several factors that are either not well captured by the simulation model or not included at all, which include post-tape processing time (clipping time in memory), smaller population of random samples and the overhead in breaking each logical tape block transfer into multiple 32 KB physical chunk transfer. This difference demonstrates the importance of these application experiments for providing useful realistic performance numbers.

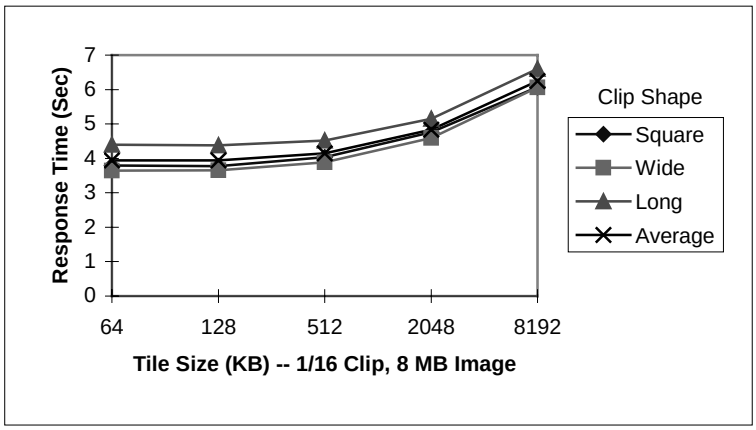

Figure 14: Varying Tile Shape, Application

Figure 15 shows the response times obtained for clipping 1/16th of 50 32 MB Images under all four configurations (Analytical, Simulation, Application and Paradise). Again, all the curves demonstrate the same trend: as tile size increases, response time increases. The amount of growth is also bigger for larger tile sizes in all cases. The best case happens between tile size 64 KB and 256 KB. As explained before, the difference between simulation and application results is mainly caused by the extra overhead in transferring everything in one big chunk versus transferring multiple contiguous smaller records (used in the tape driver). Besides, it seems that there are some extra fixed startup cost that we didn't model in analytical and simulation experiments. The difference between Paradise result and application result is more interesting. The major source of the difference is the extra hop through the staging disk. As a general DBMS engine, Paradise cannot assume the exclusive access of tape blocks for this particular

[iv] The maximum buffer size allowed for a single read/write request without being translated into multiple kernel level I/O calls is 63 KB. For tapes, each individual write call from OS is considered for a separate tape record. While tape seek operation is based on the physical record number instead of logical record number.

query, it has to transfer tape blocks to the staging disk for the purpose of sharing among multiple queries or recurrent accesses. Then, reading the tiles from staging disk into main memory buffer pool for the final processing adds even more overhead to this test. Finally, there is some extra processing overhead in the Paradise storage engine for managing large objects like tiles. Nevertheless, the tile size impact is still obvious for all experiments.

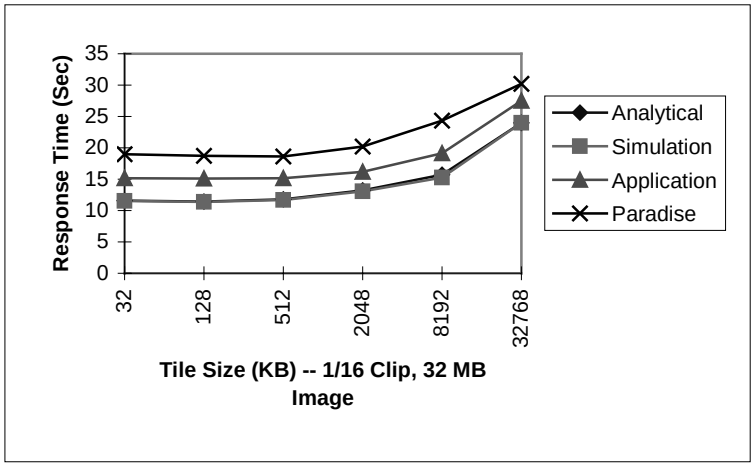

Figure 15: Comparison Among All Configurations

To see what kind of effect of clip queries on a large number of images, we next conducted the experiment of clipping 50 32 MB images with a fixed region with its area being $1/16^{th}$ of the image boundary. Figure 16 and 17 show the results from the application program and the Paradise system. Both figures show the advantage of using a smaller tile size. The 8 MB tile is the worst in both cases. Although the gain is relative small in both cases, whose upper-bound can be determined by the difference of transfer speed and seek speed, at least it shows smaller tiles post no performance penalty on the query processing. Further more, if we take into account the resource being occupied by larger tiles (especially in the case of Paradise, where staging disk space is used to buffer each tape block), a smaller tile size will definitely save resource, and thus leads to more efficient processing.

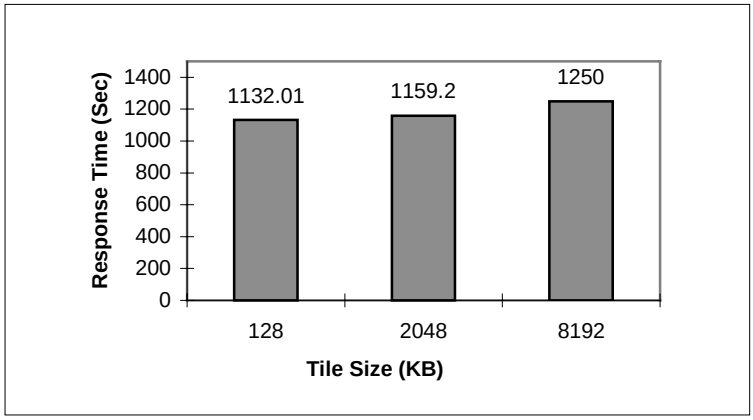

Figure 16: Clipping $1/16^{th}$ of 50 X 32 MB Images – Application

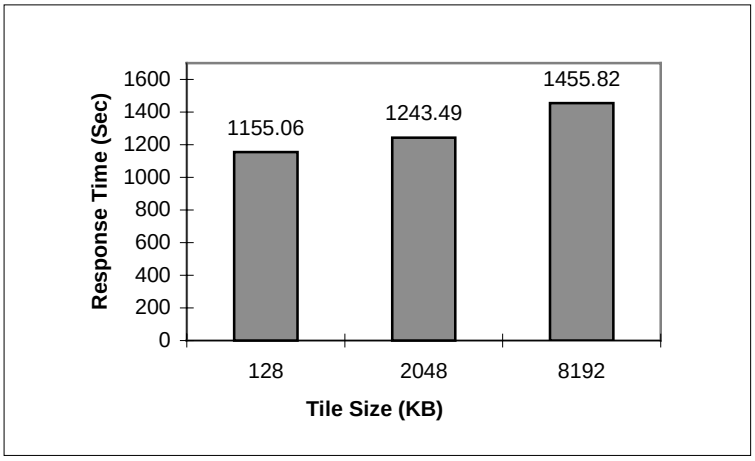

Figure 17: Clipping 1/16th of 50 X 32 MB Images -- Paradise

5. Conclusions and Future Work

We described a useful analytical model to study the impact of tile size on performance for efficient retrieval of partial images from tapes. We demonstrated from both detail simulation and performance evaluation in direct application program and an end-to-end system that the tile size has a complex effect on the response time of image clipping queries. Some interesting observations and their implications are analyzed. Overall, we find the tile size has a direct impact on retrieving partial images in a tertiary image store. Tile size in the range of 64KB and 256 KB is observed to be the optimal for the DLT tape drive. The experiments conducted in both application program and Paradise system gave us convincing evidence to support an optimal tile/tape block size to be in the range of 100 KB instead of 100 MB for efficient processing raster images in such an environment that partial image access is frequent and their access order are in the images' original loading order. This is really encouraging since block size at such a small range can help tremendously provide more flexible resource management of in-memory transfer buffer and cache disk space. The constraint of the access order to be in their tape storage order may seem to be hard to conform in a ad-hoc query environment. However, in a companion study [YD96], we developed a set of new techniques to actually utilize this order constraint to schedule query execution in a tertiary database management system.

Although we have concentrated on 2-D image clipping in this study, applying the same analysis to 3-D images (or higher dimensional arrays) may lead to more interesting findings. In particular, we believe the result can be scaled to n-dimensional arrays. With the expansion of dimension sizes, the number of seeks and seek distance can explode exponentially. An even bigger impact of tile size selection is expected for higher dimensions.

Another interesting direction is to study the impact of the ordering of tiles. We may apply techniques for clustering spatial objects (spatial filling curves like Z curve or Hilbert curve) and techniques for hierarchical sub-chunking of tiles at different levels for optimizing tape accesses.

References

[DFG94] F. Davis, W. Farrel, J. Gray, et al. “EOSDIS Alternative Architecture,” Study Report, September 6, 1994.

[EKD95] W. Emery, T.Kelley, J. Dozier and P. Rotar. “On-line Access to Weather Satellite Imagery and Image Manipulation Software,” Bulletin of the American Meteorological Society, January, 1995.

[GRA94] J. Gray. “MOX, GOX and SCANS: The Way to Measure an Archive,” EOSDIS V0 Experience, June, 1994.

[HS95] B. Hillyer and Avi Silbertschatz. “On the Modeling and Performance Characteristics of a Serpentine Tape Drive,” AT&T Bell Lab Technical Report, August, 1995.

[QUA95] Quantum Corporation, “DLT-4000 Product Manual,” 1995.

[SFG93] M. Stonebraker, J. Frew, K. Gardels and J. Meredith. “The SEQUOIA 2000 Storage Benchmark,” SIGMOD Record, 22(2):2-11, 1993.

[SUN94] S. Sarawagi. “Efficient Processing for Multi-dimensional Arrays,” Proceedings of the 1994 IEEE Data Engineering Conference, February, 1994.

[YD96] J. Yu and D. DeWitt. “Query Pre-Execution and Batching: A Two-Pronged Approach to the Efficient Processing of Tape-Resident Data Sets,” Submitted for publication, February, 1996.