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

JieBing Yu
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 [1]. Given today's mass storage technology, the data almost certainly will be stored on tape. The latest tape technology offers media that is very dense and reliable, as well as drives with transfer rates in the same range as 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 passes [2]. 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 [3]. While the cost/capacity gap [4] 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), 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.

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 this paper is how best to handle images stored on tape. We make the following assumptions:

---

[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.

1. All the images of interest to a scientist are stored on a single tape.
2. Images are accessed and processed in the order that they are stored on tape.
3. The analysis requires access to only a portion of each image and not the entire 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 [5] 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 the references 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 [6] 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 [7] also places 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 [8]. 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 by 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 [9] and Paradise [10] 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 [11].

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 KB is a typical block size for moving data between memory and disk, it is too small to use as the unit of transfer between tape and either memory or disk, especially when dealing with large raster images. The approach used instead by Postgres [5] and Paradise [10] 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 disk block size. [ii]

[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.

This paper examines the impact of tile size on the time required to retrieve one or more partial (i.e. clipped) images residing on tape. The evaluation employs a simplified analytical model, a simple simulation study to verify the analytical model, and actual implementations using both a stand-alone program and the Paradise database system, extended to include support for tertiary storage [6]. 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 demonstrate that for tape drives such as the Quantum DLT-4000, a tile size in the range of 32 KB to 512 KB provides the best performance 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 untiled 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 been tiled, the tiles are stored sequentially on tape. 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.

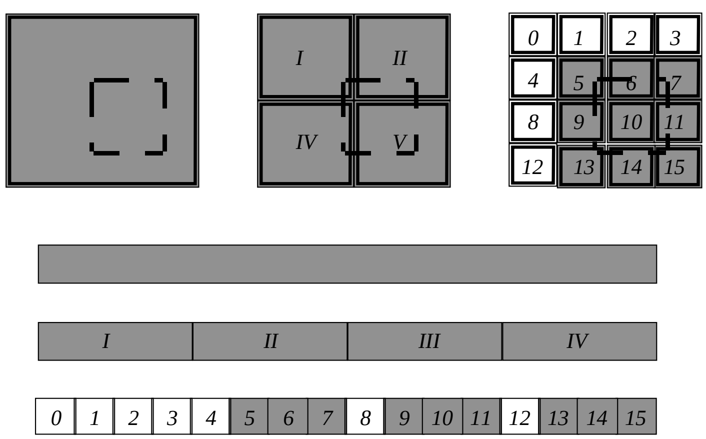

Figure 1: Single Image as 1, 4, 16 Tiles and their Linear Layout

The dashed rectangle in Figure 1 corresponds to the portion of the image that the analyst 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 untiled image (top-left rectangle in Figure 1) and the 16-tile image (top-right rectangle in Figure 1). For the untiled 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. In addition, there are a number of other factors that affect 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 untiled image. Other clip requests would have different results; for example, if the clip region was entirely contained inside tile *II* of the 4 tile image. In this case, the untiled image would incur no seeks but would transfer the entire image. The image tiled into 4 pieces would incur one seek (to the start of tile *II*) 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 the following assumptions:

1. Each tile is stored as a single *tape block*, which 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. Clipped tiles are returned from tape in their original order 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. We examine the effect of this assumption in Section 3. 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 variations in tile size and the shape of the clip region may effect performance. This will be discussed in Section 4. The final assumption minimizes the randomness between seeks within one clip 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 contains all the symbols used in the analytical model. Figure 2 graphically illustrates the roles of a number of the symbols. Note that each tile has unit length 1, a is an integer, and the fraction part of b is modeled as $b - \lfloor b \rfloor$. From Figure 2 it is obvious that the number of tiles covered or partially covered by the clip region varies depends on the particular position of the clip area. The probabilities of the various clip locations are calculated below. Finally, we analyze the time spent on each of the four operations to produce a formula that captures the average case behavior.

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

Table 1: Parameters

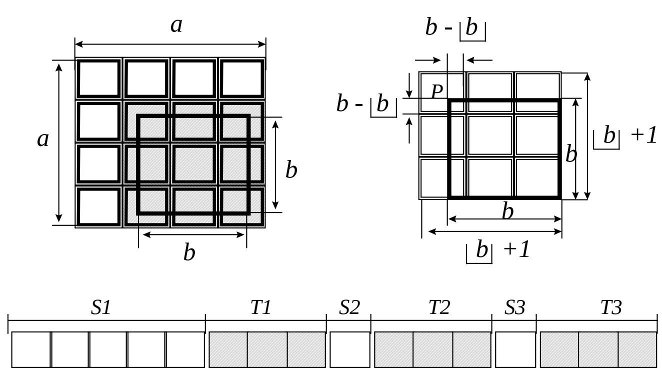

Figure 2: Target Clip Region

Four Clip Cases

As illustrated by Figure 3, there are four different ways that a region of constant area and shape can clip an image:

Case 1 -- touches $(\lfloor b \rfloor + 2)^2$ tiles;

Case 2 -- touches $(\lfloor b \rfloor + 2) \times (\lfloor b \rfloor + 1)$ tiles (elongated horizontally);

Case 3 -- touches $(\lfloor b \rfloor + 1) \times (\lfloor b \rfloor + 2)$ tiles (elongated vertically);

Case 4 -- touches $(\lfloor b \rfloor + 1)^2$ tiles.

In each case, the placement of the upper left corner (P) of the clip region is restricted to certain tiles in the image and certain regions within each of those tiles. The dark gray tiles in Figure 3 show the tiles where P can possibly reside, and the four regions in Figure 4 show where P can be placed within each of those tiles for each case. The probability for each of the four cases[iii] can be determined by examining the placement of P. Since the total area (A) containing P is $(a-b)^2$, the probability, Pb, for each case can be derived by considering the number of tiles (F) where P can be placed and the area (Af) within each of those tiles. Then, $Pb = F \cdot Af / A$.

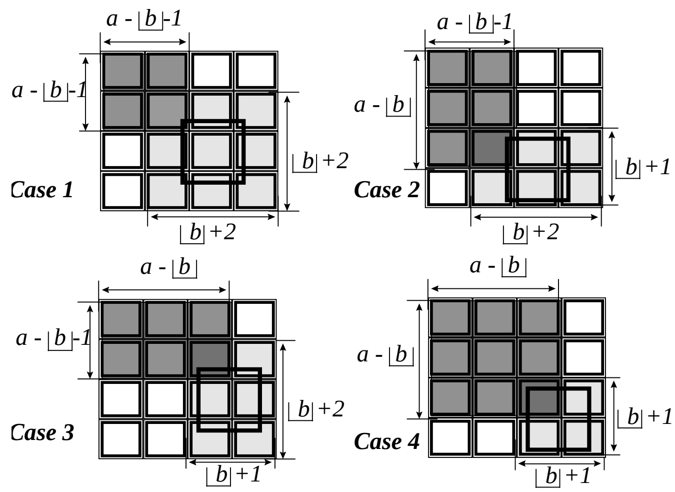

Figure 3: Four Clip Cases

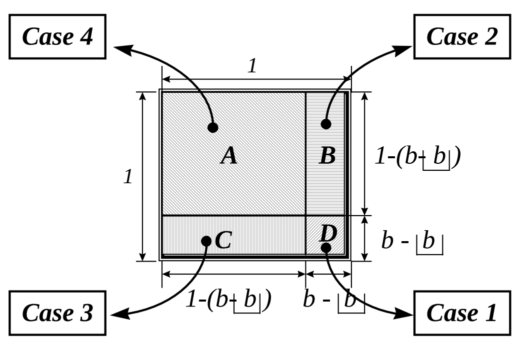

Figure 4: Regions in which P can reside for the different cases (within one tile)

The probabilities of each of the four cases occurring are specified below:

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

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

[iii] Assuming a uniform distribution of clip sizes and locations.

Case 2: $F = (a-\lfloor b \rfloor)\cdot(a-\lfloor b \rfloor-1)$, $Af = (b-\lfloor b \rfloor)\cdot(1-(b-\lfloor b \rfloor))$ (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: $F = (a-\lfloor b \rfloor-1)\cdot(a-\lfloor b \rfloor)$, $Af = (1-(b-\lfloor b \rfloor))\cdot(b-\lfloor b \rfloor)$ (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: $F = (a-\lfloor b \rfloor)^2$, $Af = (1-(b-\lfloor b \rfloor))^2$ (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 required 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 four 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: $M = (a-\lfloor b \rfloor-1)$, $N = (a-\lfloor b \rfloor)$, 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 3: $M = (a-\lfloor b \rfloor)$, $N = (a-\lfloor b \rfloor-1)$, 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 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 that 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 obtain:

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 $Ti4 = (a - (\lfloor b \rfloor + 1)) \cdot \lfloor b \rfloor \cdot t/S$.

Finally, the *Intermediate Seek Time* is :

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

Transfer Time (*Tr*)

The *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 assumptions made when calculating the *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 obtain:

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 *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 with the variable *Seek Startup Time*, ***Ts***. This overhead only depends on the number of seeks performed, and not the size of each seek operation. Using the same $X \cdot Y$ region as above, then Y seeks are needed for each clip and $Ts = Y \cdot I$. Breaking this into the different cases yields:

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 $Ts4 = (\lfloor b \rfloor + 1) \cdot I$.

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

Total Response Time

The *Total Response Time*, ***T***, is the sum of the four terms above: $T = Tf + Ti + Tr + Ts = f(a,b,S,R,I)$[iv]. To make our results easier to interpret, we substitute image size M, tile size t, and clip selectivity c for a and b ($c = a/b$ and $M = a^2 \cdot t$). Now $T = f'(M,t,c,S,R,I)$.

For analysis purposes, we fix the image size M and the clip region size ($1/c^2$ of an image). Under these conditions, the formula reveals the following interesting properties: 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 *T* derived above, we next evaluate it for a variety of parameters, plotting the response time as a function of the tile size. The values for the various 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 of the image. The tape-related parameters (*S, R, I*) are selected based on the Quantum DLT-4000 tape drive [2] with compression turned off.

Parameter	Values Evaluated
M	8 MB, 32 MB, 128 MB
c	2, 4, 8, 16
S [v]	2,048 KB/Sec
R	1,356 KB/Sec
I	0.1 (Sec)

Table 2: Selected values for parameters

The response times for a variety of image and clip size combinations along with the gain relative to always fetching an entire image are shown in Figures 5 to 7. These results indicate that, as the tile size is increased, the response time first decreases slightly before increasing and that tiling provides a significant benefit compared to fetching an entire image.

[iv] The whole formula of T, even after simplification, is too complicated to present. Instead, we will show its parameters and present them graphically.

[v] Because DLT uses serpentine tapes the seek time is not a linear function of the seek distance. For the DLT4000 we observed a maximum seek time of 180 between two random blocks. If the seek is within a single track the seek time is close to a linear function. [12] contains an accurate model of seek time for DLT drives.

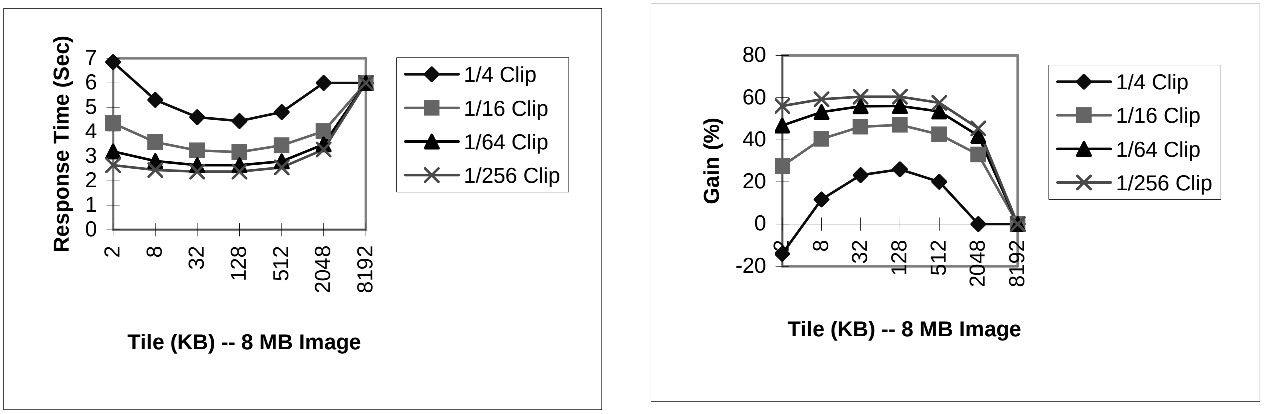

Figure 5: Response Time and Performance Gain over Entire Image Transfer for 8 MB Images

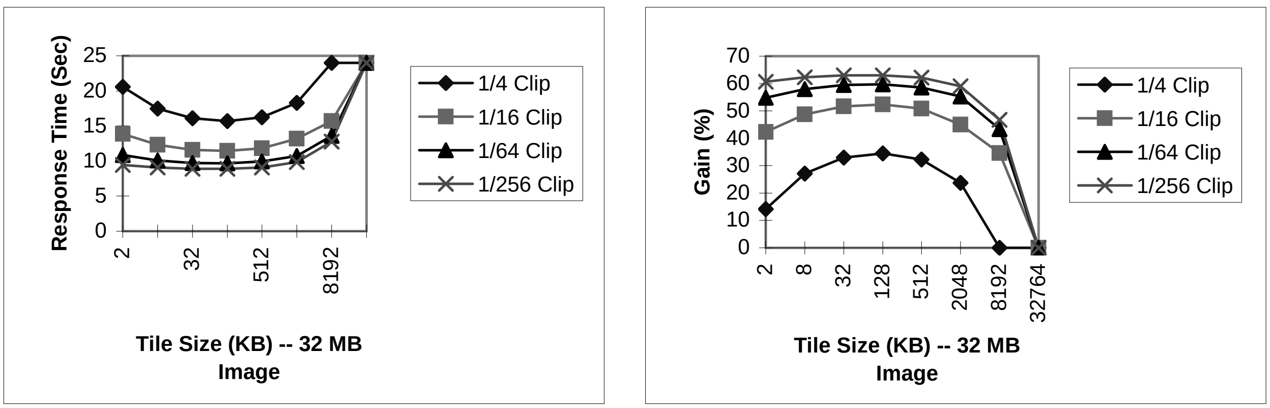

Figure 6: Response Time and Performance Gain over Entire Image Transfer for 32 MB Images

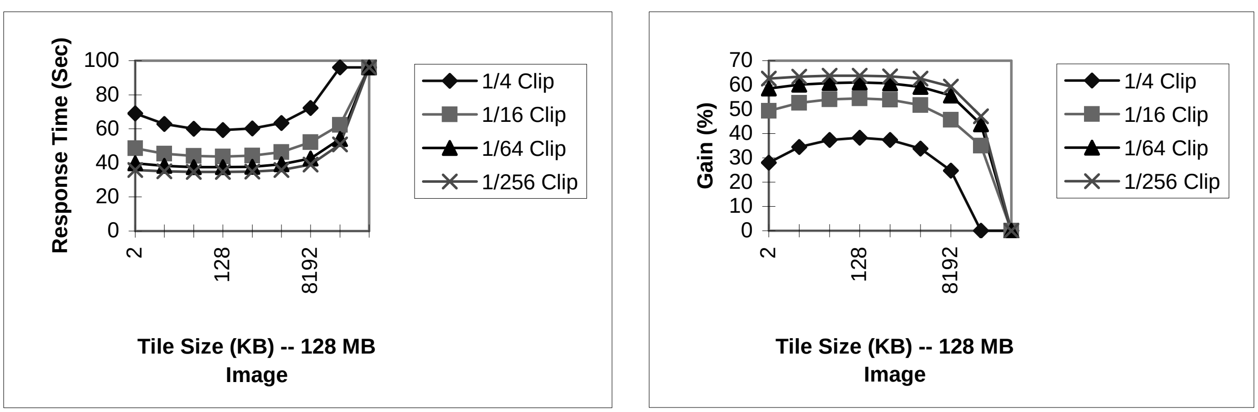

Figure 7: Response Time and Performance Gain over Entire Image Transfer for 128 MB Images

To help understand these results, Figure 8 decomposes the time required to clip 1/16th of a 32 MB image into its three component parts: *Seek*, *Transfer*, *and Seek Overhead*. 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 eventually dominates the response time on the right side of the graph. The effect of tape seeks is best illustrated for tile sizes less than 512 KB where the reduction in seek time, resulting from both fewer and shorter seeks, offsets the increase in transfer time as the tile size increases.

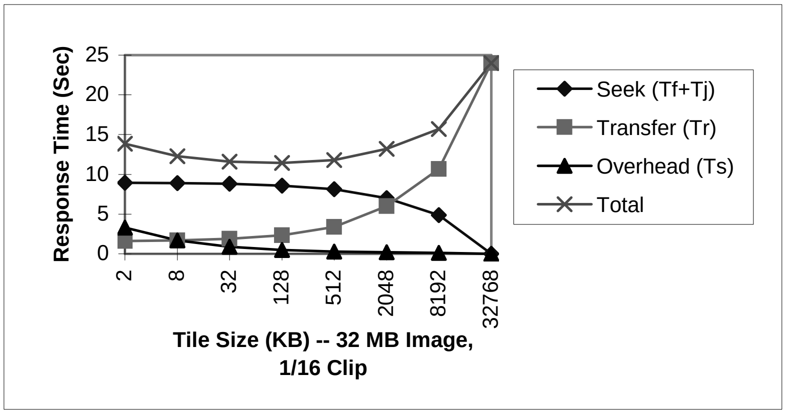

Figure 8: Breakdown of Response Time

Based on these figures, for the DLT drive a tile size in the region between 32 KB and 512 KB provides the best performance for a variety of image and clip sizes. In general, using appropriately sized tiles provides between a 20% and 70% improvement compared to fetching the entire image.

3. Simulation Experiments

From the analytical model described in Section 2 it is clear that 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 that we use in this section to explore a broader range of test conditions.

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 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, and the clip shape need not be proportional to the shape of the image. 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.

3.2 Analytical Model Verification

To verify the analytical model, using the simulation model we repeated the experiments presented in Figures 5, 6, and 7.. To simplify the comparison of the two sets of results, we show the relative differences in response times from the two models in Figures 9, 10, and 11. The difference is never greater than 4%. In general, the response times generated by the simulation model are slightly lower than the analytical model (i.e. a negative difference) for larger tile sizes and slightly higher for extremely small tile sizes. This is due to the randomness in selecting clip regions. This error margin is acceptable given the number of tests used. Based on these results, it is clear that the analytical model is quite accurate given the assumptions it is based on.

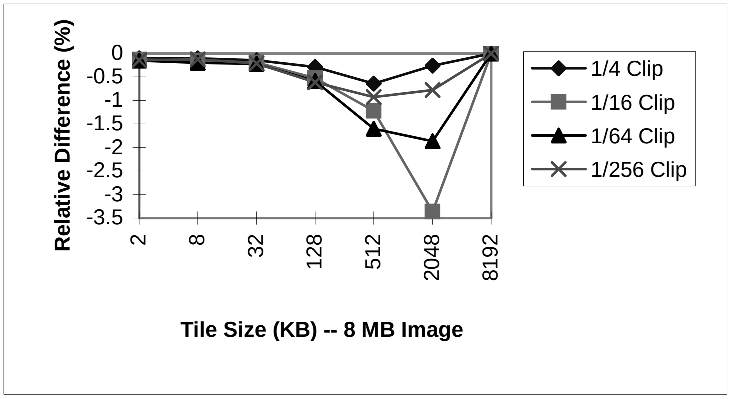

Figure 9: Relative Difference between Analytical and Simulation Result -- 8 MB Image

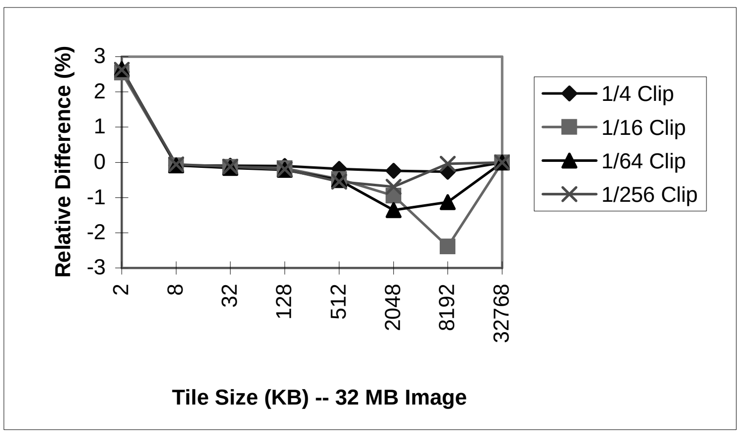

Figure 10: Relative Difference between Analytical and Simulation Result -- 32 MB Image

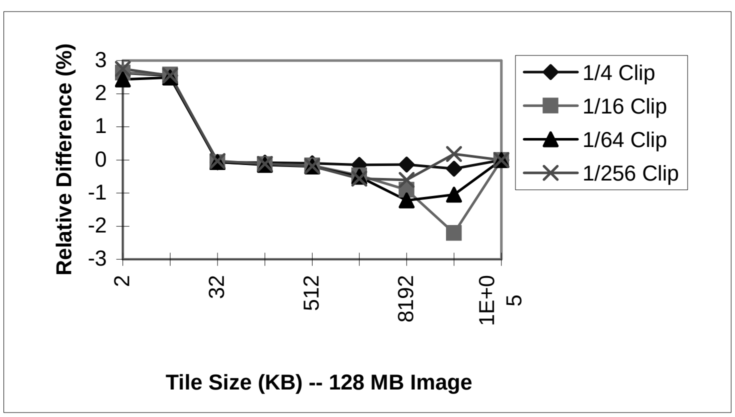

Figure 11: Relative Difference between Analytical and Simulation Result – 128 MB Image

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 significantly affect performance 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 an effect similar to increasing the tile size. However, since packing tiles into a bigger tape block is done in 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 no longer rectangular. This irregular shape can further degrade performance. These results indicate that it is the best to use the same tile and tape block size. Consequently, all subsequent results presented in this paper use the same block size and tile size.

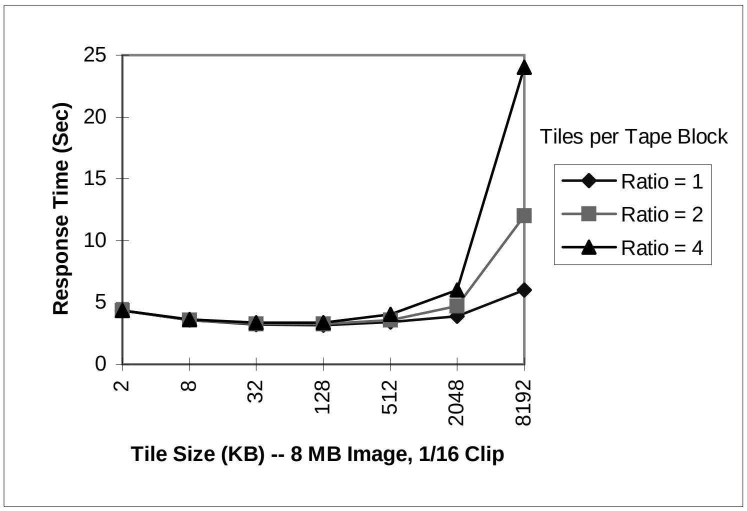

Figure 12: Effect of Tape Block Size vs. Tile Size

3.4 Alternative Clip Shapes

One of the assumptions made for the analytical model was that the shapes of the clip region and image had to be the same. Clip regions with the same area but different shapes can also affect performance. To investigate this effect, we experimented with three different clip shapes: *Long*, *Wide*, and *Square*. *Long* is a thin rectangular shape whose height is four 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 results from this experiment and illustrates that the different shapes indeed have different response times. It is interesting to notice that the *"Wide"* curve is consistently below the *"Long"* curve. This is caused by 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 "*Square*" clips is relatively close to the average behavior, and in most cases, is just between the "*Wide*" and "*Long*" curves. However, when the tile size is much larger than the clip size (e.g. 2,048 KB), the "*Square*" shape had lower responses than the "*Wide*" shape because it is less likely to overlap more than one tile. A key result from this set of experiments is that, regardless of the clip shape, the best response time occurs when the tile size is between 32 KB and 512 KB.

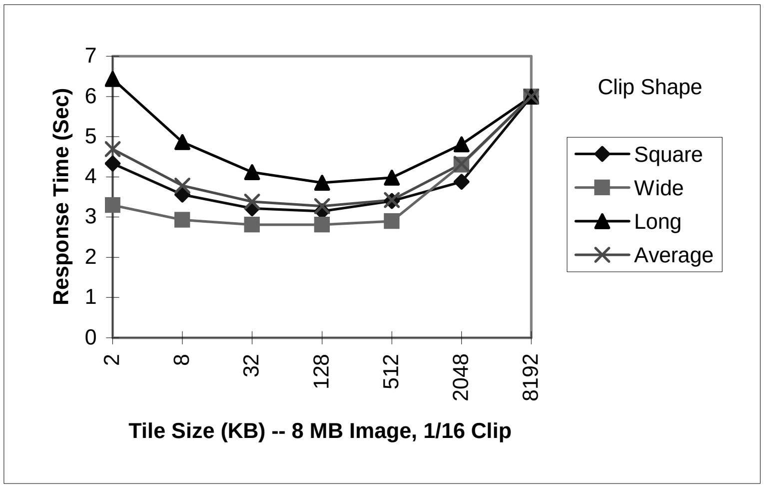

Figure 13: Alternative Clip Shapes, Simulation

4. Implementation Results

To verify the results obtained both analytically and through simulation, we next repeat some of the experiments using the following two configurations: an application-level program that accesses raster images directly on tape, and Paradise -- a DBMS extended to handle data on tape.

4.1 Configuration

The application program is a stand-alone program that is capable of accessing tiled raster image stored on tape. This corresponds to a typical scientific application accessing a tape-resident data set. Paradise [10] 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 with compression to maximize performance. In a separate project [6], Paradise was extended to include support for tertiary storage. Both the application-level program and Paradise share the same block-based DLT device driver and the raster image clip code. Thus, the amount of time each spends doing tape I/Os and clipping raster images in memory is comparable. However, the application program directly transfers data from tape to user space while Paradise first stages tape blocks on a staging disk. The experiments using Paradise represents the end-to-end performance 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 block-based tape driver breaks each tape blocks larger than 32 KB into multiple 32 KB physical chunks before performing the actual tape I/O via an ioctl call. This scheme is used to simplify the mapping between physical and logical block addresses[vi]. A single logical tape block is mapped to multiple contiguous 32 KB physical records on tape. For Paradise, all queries were executed with cold buffer pools (for both main memory and disk cache). Due to the high cost of running actual tests, we had to cut down the number of randomly generated clip shapes from 1000 (used in simulation experiments) to 20.

[vi] The maximum system I/O buffer size for a single read/write request without being translated into multiple kernel level I/O calls is 63 KB. Since each kernel-level tape write call generates a separate record on tape, it is easier to manage smaller physical chunks and provide a large (variable sized) record interface for the higher level.

4.2 Alternative Clip Shapes -- Application Program

Figure 14 displays the results obtained using the application program for the same set of experiments presented in Figure 13. Although there are some discrepancies at the ends of the curves, the general trends are the same. The *Wide* shape benefits the most from the layout of the tiles and, thus, has consistently lower response times. The *Long* shape tends to seek across more tiles than the other cases and hence has the highest response times. Finally, the *Square* shape is close to the average case. Note that the response times for 2 and 8 KB tile sizes are lower than the values predicted by the simulation model. This might be explained by the DLT tape drive's internal read-ahead cache. While no literature was found in the product manual on how it works, we suspect that tape head does not stop at the end of the last read but may actually read ahead to some location further down the track, thus hiding some of the seek latency. With small tile sizes it is more likely that most of the intermediate seeks can be absorbed by this read-ahead mechanism. This can also explain why the different clip shapes are closer together with the application-level program. Another discrepancy between the two sets of results is the difference in absolute times. This is caused by several factors that are not captured by the simulation model, including post-tape processing time (ie. the clip time in memory), a smaller population of random samples, and the overhead of decomposing logical tape blocks larger than 32 KB into multiple 32 KB physical chunk transfers.

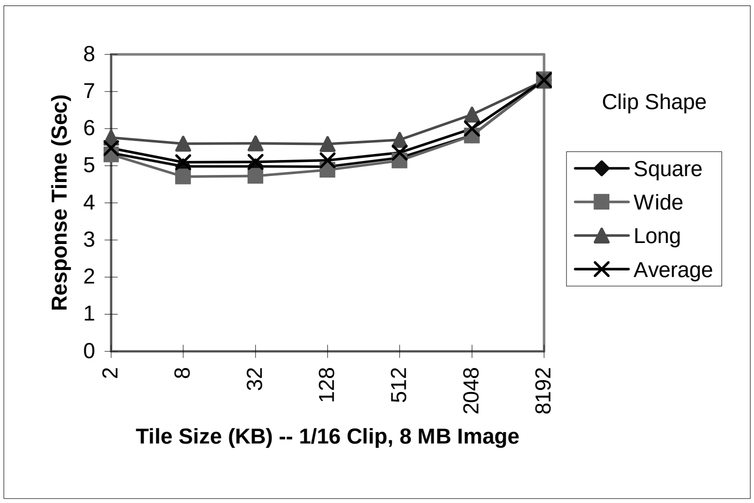

Figure 14: Alternative Clip Shape, Application Program

4.3 Comparison among all Models

Figure 15 shows the response times obtained for clipping 1/16th of a 32 MB Image under all four configurations (Analytical, Simulation, Application and Paradise)[vii]. Again, all the curves illustrate the same trend: as the tile size is increased beyond 32 KB the response time increases. As explained above, the difference between the simulation and application-level results is mainly due to the extra overhead of having to break large tape I/Os into multiple 32 KB chunks. In addition, it appears that there are additional fixed startup costs that are not captured by the analytical or simulation models.

There are a number of causes for the difference between Paradise and the application-level program. First, as a general DBMS engine, Paradise assumes that tape blocks requested by one query might be

[vii] The curves for analytical and simulation models are almost the same as discussed in Section 3.2.

useful for other, concurrently executing queries. Thus, tape blocks are staged first on disk and then copied into Paradise's buffer pool in main memory on demand. Second, Paradise incurs some extra processing overhead while managing large objects like tiles. Nevertheless, the impact of tile size on performance is apparent.

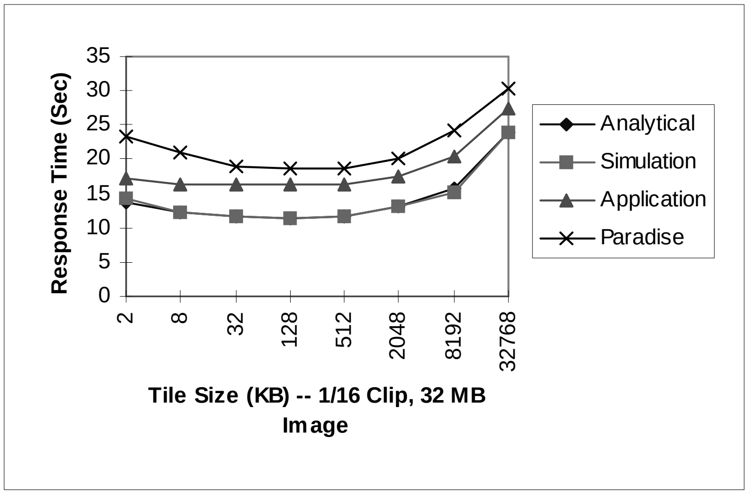

Figure 15: All Configurations

4.5 Processing of Multiple Images

To determine the effect of clipping a large number of images, we next conducted an experiment in which fifty 32 MB images were clipped by a fixed region whose area was 1/16th the size of the image boundary. Figure 16 and 17 show the results from this test on the application-level program and Paradise. While both figures show a small advantage of using a smaller tile size, the most promising result is that Paradise's performance for this test is within 5-10% of the hand-coded application pr ogram.

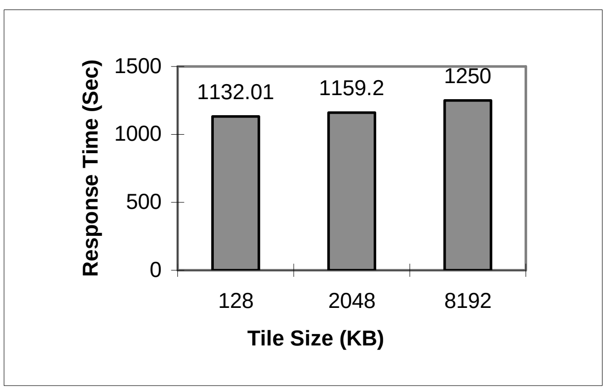

Figure 16: Time to Clip 1/16th of Fifty 32 MB Images – Application

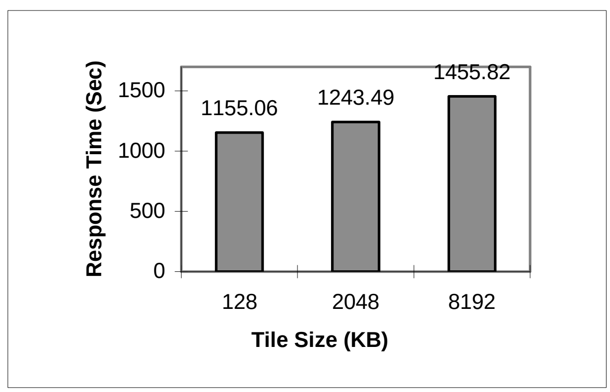

Figure 17: Time to Clip 1/16th of Fifty 32 MB Images -- Paradise

5. Conclusions and Future Work

In this paper we described a simple analytical model to study the impact of tile size on the performance of retrieving partial satellite images from tape. Using this analytical model as well as a simulation model and two actual implementations (Paradise and a hand-coded application program) we demonstrated that the tile size has a complex effect on the cost of executing queries involving clips of partial satellite images stored on tape. Our results indicate that, for the Quantum DLT 4000 tape drive, a tile size between 32 KB and 512 KB provides the best performance for a wide range of image and clip region sizes. These results are very encouraging as smaller tile sizes simplify space management on both the disk cache and buffer pool while providing good performance.

While this study assumed that the images being processed are stored sequentially on tape, in a companion study [6] we describe a set of new techniques that are capable of reordering tape accesses from complex object-relational queries in order to satisfy this constraint. Although this paper dealt only with 2D images, we believe that our results also apply to 3D images as well as arbitrary N-dimensional arrays stored on tape. As dimensionality is increased, the number of seeks and the seek distances can increase exponentially. We expect that in these cases the tile size will have even a larger impact.

References

[1] B. Kobler and J. Berbert, "NASA Earth Observing System Data Information System (EOSDIS)," Digest of Papers: 11th IEEE Symposium on Mass Storage Systems, Los Alamitos, 1991.

[2] Quantum Corporation, "DLT-4000 Product Manual," 1995.

[3] M. Carey, L. Haas and M. Livny. "Tapes Hold Data Too: Challenges of Tuples on Tertiary Store," Proceedings of the 1993 SIGMOD Conference, May, 1993.

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

[5] S. Sarawagi. "Efficient Organization for Multi-dimensional Arrays," Proceedings of the 1994 IEEE Data Engineering Conference, February, 1994.

[6] 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.

[7] 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.

[8] R. Herring and L. Tefend. "Volume Serving and Media Management in a Networked, Distributed Client/Server Environment," Proceedings of the 3rd Goddard Conference on Mass Storage Systems and Technologies, NASA Con. Publication 3263, 1993.

[9] F. Davis, W. Farrel, J. Gray, et al. "The Sequoia Alternative Architecture Study for EOSDIS," NASA Study Report, October, 1994.

[10] D. DeWitt, N. Kabra, J. Luo, J. Patel and J. Yu. "Client Server Paradise," Proceedings of the 20th VLDB Conference, September, 1994.

[11] M. Stonebraker, J. Frew, K. Gardels and J. Meredith. "The SEQUOIA 2000 Storage Benchmark," Proceedings of the 1993 SIGMOD Conference, May, 1993.

[12] B. Hillyer and Avi Silbertschatz. "On the Modeling and Performance Characteristics of a Serpentine Tape Drive," Proceedings of the 1996 SIGMETRICS Conference, May, 1996.