# Modelling and Storage Allocation Strategies for Homogeneous Parallel Access Storage Devices in Real Time Multimedia Information Processing

C. Y. Roger Chen,
Dept. of Elect. and Comput. Eng.,
Syracuse University,
Syracuse, NY 13244.

Kingsley C. Nwosu,
IBM POWER Parallel Systems,
MS/992, Neighborhood Rd.,
Kingston, NY 12401.

P. Bruce Berra,
CASE Center,
Syracuse University,
Syracuse, NY, 13244.

**Abstract**[1]

*The improvements in disk speeds have not kept up with improvements in processor and memory speeds. Conventional storage techniques, in the face of multimedia data, are inefficient and/or inadequate. Here, an efficient multimedia object allocation strategy is presented. We describe a multimedia object model, the object and storage device characteristics, and the fragmentation strategy. A bipartite graph approach is used for mapping fragments to storage devices and a cost value is used to determine an efficient allocation of an object and to balance the loads on the devices.*

**Keywords**: *multimedia, fragmentation, real-time, storage allocation, bipartite graph, efficient allocation.*

## 1 Introduction

Due to the rapid advances in the technology of display devices, computers, networks, storage devices, and software engineering, the emerging multimedia application has become one of the most important and promising research areas. Multimedia is a combination of motion and still video, special effects, synthetic video, graphics, text, voice, audio, and images. By using the object-oriented approach, multimedia data can be processed and manipulated by users in a universal way, regardless of the media types and sizes of objects. However, from a system's point of view, many problems arise in supporting such an object-oriented multimedia system. Among the problems, a most serious one is related to the storage. This is due to the fact that processor speed, memory speed, and memory size have grown exponentially over the past few years [5], while disk speeds have improved at a far slower rate. Consequently, the speed of the disk rather than the speed of the CPU is the limiting factor in many applications. For real-time information retrieval and presentation, it is imperative that data, for a given medium, be retrievable at some given rate. The rates for some media are very high for current storage devices. The most conspicuous of these is in the area of digital video. For example, the video data object based on the *NTSC* standard requires that video data be retrievable at a rate of 45 Mbits/sec. However, the peak speed of a magnetic disk drive is about 10 Mbits/sec. and CD-ROMs operate at 1.2 Mbits/sec. To meet the bandwidth requirement of a full-motion video file, it is clear that the file has to be spilt into multiple sub-files, stored in different disks; when needed, an interleaving technique will be performed to combine the multiple data streams into a single data stream and then present it to the user.

Conventional allocation techniques (such as data stripping/de-clustering
[3] and data contiguity/clustering [2]) are developed mainly for text and numeric files, which although can be different in sizes, are more or less on the same order. Unfortunately, when applied to multimedia applications, the conventional techniques are inadequate and inefficient. Several file system level approaches [1] have been proposed and utilized; however, they do not encompass the gamut of multimedia types and are mostly for continuous media types (digital audio and video) without addressing the storage allocation with emphasis on a multimedia object's real-time retrievability requirements.

[1] Published in Proc. IEEE International Conference on Computing and Information (ICCI), Sudbury, Ontario, Canada, May 1993

Figure 1: An example of a multimedia object tree.

to achieve some degree of I/O parallelism: (1) when the bandwidth requirement of an object is more than the I/O transfer rate of a disk, e.g., the bandwidth requirement for displaying a full motion digital video file and (2) when parallel access of an object is helpful in computations for either multiprocessors or vector processors. On the other hand, there are objects that cannot benefit from a splitting and will be stored in a single disk. In case 1, it is required that a minimum number of storage devices be available to achieve the expected parallelism.

Figure 2 shows a composite object where $DE1$ is of case 1, and $DE4$ is of case 2. $DE2$ and $DE3$ are not split because they cannot benefit from it.

In the rest of this paper, we will refer to DEs of type $DE1$ as *class-one* DEs; objects of type $DE2$ or $DE3$ as *class-two* DEs and DEs of type $DE4$ as *class-three* DEs. Any of the allocation units, $f1, \ldots, f9, DE2$, and $DE3$, will henceforth also be referred to as an **Atomic Unit** (AU).

## 3 Multimedia object/storage characteristics and problem formulation

Each composite multimedia object will be denoted as $O_i$, the $j$th DE in $O_i$ as $o_j$, and the number of DEs in $O_i$ as $\Psi_i$. Associated with every $o_j$ of a given composite object is the *frequency distribution*, $\pi_j$. This distribution frequency represents the relative probability that a given DE will be requested for retrieval. Clearly, $0 < \pi_j \leq 1$. The size of $o_j$ is denoted as $s_j$. Each class-one DE, $o_j$, has an *expected retrieval rate*, $r_j$. This rate represents the minimum number of bytes of $o_j$ that should be retrieved per unit time in order to achieve its real-time requirements. Each class-three DE, $o_k$, is associated with a *degree of parallelism* which indicates the degree of concurrent access to the DE that may be helpful for computations for either multi-processors or vector processors.

Like the multimedia objects, the $k$th storage device is denoted as $S_k$, its bandwidth as $S_k^r$, total amount of space already allocated as $S_k^a$, and the free space as $S_k^f$.

In this paper, we assume homogeneous storage devices, i.e., $\forall_{k=1,m}\ S_k^r = \sigma$. We denote the $k$th AU of $o_j$ as $a_{j,k}$ and the total number of AUs in $o_j$ as $\alpha_j$. The total number of AUs in $O_i$ is represented as $\Pi_i$. Consequently, the storage allocation problem is formulated as follows: Given a composite object, $O_i$, determine an efficient allocation for each of its AUs such that the (1) real time requirements (expected retrieval rates) for class-one DEs are satisfied, (2) class-two DEs are reasonably distributed, (3) degree of parallelism for class-three DEs are efficiently achieved, (4) amount of data retrievable per unit time is maximized, and (5) the loads are sufficiently balanced on the storage devices.

## 4 Problem and allocation analysis

We define $\models$ as a mapping function where $a \models b$ implies that $a$ is mapped to $b$. The *intra DE allocation* specifies the allocation policy that must exist when allocating the AUs of a DE. In the case of class-one or class-three DE, it is imperative that in order to achieve its expected retrieval rate or degree of parallelism, each of its AU must be stored in a different storage device. Therefore, if $o_i$ is a class-one or class-three DE, then the intra DE allocation policy states that

$$\forall_{k=1,\alpha_i} \text{ if } a_{i,k} \models S_h \text{ then } \forall_{g=1,\alpha_i} \not\exists a_{i,g} \text{ such that } a_{i,g} \models S_h \quad (k \neq g) \text{ and } (1 \leq h \leq m).$$

As is prevalent in most complex object oriented systems, users or applications may need to access all the data associated with a complex object concurrently; therefore, it becomes necessary that all the AUs of a complex object be stored such that all of its data can be retrieved concurrently. We, therefore, allocate each

AU of a complex object to a different storage device. Therefore, if $o_1, \ldots, o_j$ are the DEs of a complex object, then the *intra complex object allocation* policy states that

$$\forall_{x_1=1,j}\ \forall_{x_2=1,\alpha_{x_1}} \text{ if } a_{x_1,x_2} \models S_h \text{ then}$$
$$\forall_{x_3=1,j}\ \forall_{x_4=1,\alpha_{x_3}}\ \nexists a_{x_3,x_4} \text{ such that}$$
$$a_{x_3,x_4} \models S_h, \quad (x_3 \neq x_1, 1 \leq h \leq m).$$

Similarly, we may need to access all the data associated with two or more complex objects concurrently. It becomes necessary that the corresponding AUs of the DEs of those complex objects be stored in different storage devices. The *inter complex* or *intra composite object allocation* policy states that if $o_{j_1}, \ldots, o_{j_n}$ and $o_{k_1}, \ldots, o_{k_n}$ belong to two complex objects, then

$$\forall_{x_1=j_1,j_n}\ \forall_{x_2=1,\alpha_{x_1}} \text{ if } a_{x_1,x_2} \models S_h \text{ then}$$
$$\forall_{x_3=k_1,k_n}\ \forall_{x_4=1,\alpha_{x_3}}\ \nexists a_{x_3,x_4} \text{ such that}$$
$$a_{x_3,x_4} \models S_h, \quad (1 \leq h \leq m).$$

Sometimes, an access crosses multiple composite multimedia objects. When allocating a composite multimedia object, we have to account for the situation depicted in Figure 3 where $O1$ and $O2$ share $O1$'s $o_2$. The *inter composite object allocation* states, from the example in Figure 3, that if $O1$ was allocated first, then none of $O2$'s AUs should be allocated in the same storage device with $O1$'s $o2$.

## 5 Fragmentation strategies

For class-one or class-three DEs, we have to decompose their data into fragments that foster parallel reads to achieve their expected retrieval rates. In order to obtain these fragments, we have to determine the degree of decomposition of a given class-one DE. We need to compute the number of storage devices that can be accessed in parallel to satisfy the retrievability requirement; this value is called a DE's *storage length* ($\gamma$). Let $\gamma_j$ represent the storage length of $o_j$. Therefore, $\gamma_j = \lceil \frac{r_j}{\sigma} \rceil$ After computing $\gamma_j$, one of two situations arises: (1) $\gamma_j > m$ or (2) $\gamma_j \leq m$ where $m$ is the number of storage devices in the system. Each DE consists of chunks of data called *Storage Element* (SE). We have to determine the number of SEs in a DE, generate the necessary SEs, and then group them to meet the bandwidth requirement. We represent the number of SEs in $o_j$ as $\delta_j$ and call it a DE's $\delta$-value. Therefore, $\delta_j = \lceil \frac{s_j}{\sigma} \rceil$. A DE's storage length is the grouping factor for the SEs to produce the AUs. For example, a class-one DE of size $120KB$ and expected retrieval rate of $40KB$ per unit time in an environment with storage devices of $20KB$ bandwidth, can be split into 6 SEs, $\{f_1, f_2, \ldots, f_6\}$, and then grouped into 2 AUs, $\{a_1, a_2\}$, as shown in Figure 4. The numbers above the boxes represent the physical addresses of the SEs.

In the case of class-two DEs, each DE is made up of one AU which consists of one SE.

## 6 Proposed mapping techniques

Given an AU, we have a list of storage devices to which it is allocatable. If the AUs and storage devices represent nodes in a graph, then we can construct an edge from an AU to a storage device to which that AU is allocatable. We must then select one of these storage devices as the most efficient storage for the AU. In order to accomplish this, one must consider the effects of allocating a given AU to all the possible storage devices. If we assign a weight to each of these nodes, then one can, using some criteria, determine the best allocation for a given AU. In order to fairly balance the loads, we need to specify some factors that will help to determine the efficient allocation of an AU. Prominent among these factors are the current status of a storage device with respect to the AUs already allocated, the effect of the free space in the storage device, and the bandwidth of the storage device. The current status function must be defined in terms of an AU's size and frequency and we call it the *expected disk traffic requirement* and represent it by a function $\mathcal{F}$. This function must always be defined such that a DE's frequency is emphasized and certain characteristics of the multimedia environment should also be taken into consideration. Through many experiments, we have found that $\mathcal{F}(f, z) = z\frac{1}{1-f}$, where $f, z$ are frequency and size of an AU, respectively, seems to be a good choice. Of course, $\mathcal{F}$ can be defined in many other ways to emphasize special characteristics of an environment. Let $h$ be the number of AUs already allocated to a storage device $S_k$, $a_i$, $1 \leq i \leq h$, the AUs, $SIZE(a_i)$ the size of AU $a_i$, and $FREQ(a_i)$ the frequency of AU $a_i$. The current cumulative traffic requirement of $S_k$, $S^w_{k,h}$, is computed as:

$$S^w_{k,h} = \sum_{i=1}^{h} \mathcal{F}(FREQ(a_i), SIZE(a_i)).$$

The cumulative traffic requirement of a storage device is an indication of the expected access to the storage device with respect to the AUs allocated to it. That factor alone does not determine an efficient storage device to allocate an AU. We need to determine the *expected disk traffic per unit of allocated space* in a storage device. We extend the expected disk traffic

per unit of allocated space and determine the *induced expected disk traffic per unit of allocated space*. We denote as $\Delta S^w_{k,i}$ the induced expected disk traffic per unit of allocated space by $a_i$ on $S_k$. Consequently, for some $a_i$,

$$\Delta S^w_{k,i} = \frac{S^w_k + \mathcal{F}(FREQ(a_i), SIZE(a_i))}{S^a_k + SIZE(a_i)}.$$

It is undoubtably obvious that the amount of free space in a storage device plays a role in determining the current and future utilization of a storage device. The fact that a storage device has a low cumulative traffic requirement relative to another storage device does not convincingly indicate that it is under-utilized relatively. If the storage device with higher cumulative traffic requirement has considerably larger free space, then it is imperative that relative to their available spaces, it is under-utilized.

The effect of the bandwidths is not considered since the storage devices are homogeneous.

Therefore, our mapping goal is to select the storage device that minimizes these factors. We represent the sum of these factors as a cost function $\varsigma$. If

$c_1$ = cost induced by induced expected disk traffic per unit of allocated space, and
$c_2$ = cost induced by free space with respect to the cumulative traffic requirement,
then,

$$\varsigma = e_1 c_1 + e_2 c_2$$

where $c_1 = \Delta S^w_k$, and $c_2 = \frac{S^w_k}{S^f_k}$.
The coefficients $e_1$ and $e_2$ are the accentuating values. We use them to emphasize or de-emphasize the relative importance of the corresponding induced cost. We recommend that these coefficients be in the range of zero and 1. Given an AU and the $\varsigma$ costs of allocating it to different storage devices, we select the storage device with minimum cost. Our minimization goal is such that given $g$ AUs of a composite multimedia object and $\varsigma_{i,k}$ as the cost of allocating AU $a_i$ to the storage device $S_k$, therefore,

$$\text{minimize } \sum_{i=1}^{g} \varsigma_{i,k} \text{ where } \exists S_k \ (1 \le k \le m) \text{ such that } a_i \perp S_k.$$

## 7 Bipartite graph model

One of the widely studied graphs is the *bipartite graph* [4]. The bipartite graph used here consists of two sets of vertex partitions, $V_a$ and $V_s$. The $V_a$ vertices are made up of AUs while the $V_s$ vertices are made up of the storage devices. Therefore,

$$\forall v \in \Re^{one} \vee \Re^{two} \vee \Re^{three}, v \in V_a$$
and
$$\forall_{k=1,m} S_k \in V_s$$

The *allocatability* of an AU to a storage device depends on space availability. An AU has $m'$ link fields where $m'$ is the number of storage devices to which it is allocatable. We label each edge with the $\varsigma$ of the corresponding storage device.

Figure 5 below shows a typical bipartite graph with 10 AUs. The high-lighted edges show a possible solution for the allocation. This selection process is similar to the bipartite matching problem which has been applied to numerous problems such as the *max-flow problem* [4], *bipartite weighted matching problem* [7], also known as the *assignment problem*. Several algorithms have been developed to solve these problems, however, the most widely utilized and best fits our problem is the *Hungarian Method*[7]. The Hungarian Method guarantees a solution for a complete bipartite graph if one exists. The failure of the Hungarian Method does not necessarily imply that the composite object is unallocatable. After a failure, we reduce the number of AUs in the composite object and try again until the object becomes unreducible.

## 8 Simulation model and results

In the results shown below, each composite object consists of a random number of DEs of size between 1 byte and 500MB. The expected retrieval rates range from .125KB/s to 30MB/s. Each of the storage device has a capacity of 1GB and a bandwidth of 500KB/s.

Figure 6 shows the distribution, according to frequencies, of the total number of DEs generated. The expected disk traffic function used, given size, $z$, and frequency, $f$, is $\mathcal{F}(f,z) = z[\frac{1}{1-f}]$.

Figure 7 shows an example final cumulative traffic requirements of the storage devices when a fair mixture of DEs of different frequencies were used. Comparative results were also obtained when more low or high frequency objects were used.

## 9 Conclusions

We have presented a multimedia object model and described the allocation strategy necessary to achieve the real time retrieval requirements of the modeled multimedia objects. A bipartite graph model is used to represent object allocatability and the Hungarian Method is used to determine efficient allocation using a cost value.

## References


[1] D. P. Anderson and Y. Osawa, "A File for Continuous Media," *ACM Trans. on Computers*, Vol. 10, No. 4, pp. 311-337, Nov., 1992.

[2] A.D. Bell, F.J McErlean, and P.M. Stewart, " Clustering Related Rules in Databases," *The Computer Journal*, Vol. 31, pp. 253-257, June 1988.

[3] P. Chen, D. Patterson, "Maximizing Performance in a Striped Disk Array," *Proc. 1990 ACM SIGARCH 17th Intern. Symp. on Comp. Arch.*, Seattle, WA, pp. 322-331, May 1990.

[4] T.H Cormen, C.E Leiserson, and R.L Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[5] W. Myers, "The Competitiveness of U.S.A. Disk Industry," *IEEE Computer*, Vol. 19, No. 11, pp. 85-90, 1986.

[6] D. Patterson, G. Gibson, and R. Katz, "A case for Redundant Arrays of Inexpensive Disks(RAID)," *In Proc., 1988 ACM SIGMOD Int'l Conf.*, pp. 109-116, May 1988.

[7] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Prentice-Hall, 1982.

[8] P. V. Rangan, T. Kaeppner, and H. M. Vin, "Techniques for Efficient Storage of Digital Video and Audio," *International Workshop on Multimedia*, pp. 68-85, Feb. 1992.

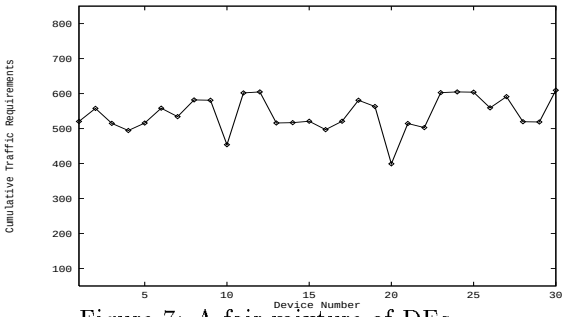


Figure 7: A fair mixture of DEs.

Figure 5: A bipartite graph showing AUs allocatabilities.

| size | < 2KB | < 10KB | < 100KB | < 500KB | < 1MB | < 10MB | < 50MB | < 200MB | < 400MB | < 500MB |
|---|---|---|---|---|---|---|---|---|---|---|
| frequency | .160 | .147 | .130 | .120 | .110 | .093 | .080 | .067 | .053 | .040 |

Figure 6: Distribution of simulation data.