Multimedia Object Modelling and Storage Allocation Strategies for Heterogeneous Parallel Access Storage Devices in Real Time Multimedia Computing Systems

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

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 function is used to determine an efficient allocation of an object and to balance the loads on the devices.

Keywords: *bipartite graphs, bipartite matching, efficient allocation, fragmentation, multimedia, storage allocation.*

1 Introduction

The rapid advances in the technology of display devices, computers, networks, storage devices, and software engineering have pushed the emerging multimedia applications into becoming one of the most important and promising research areas. Multimedia information processing encompasses the integrated generation, representation, processing, storage, and dissemination of independent machine processable information expressed in multifarious time dependent and independent media. A unique feature of multimedia is the highly diversified media types and file sizes. In order to avoid dealing with the heterogeneity of multimedia data, multimedia applications are usually developed using an object-oriented approach, where each object represents a file of video, audio, image, graphics, text, etc; or a combination of them. Moreover, it is usually required to integrate or combine multiple objects of various media types into multiple-level complex objects. 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 [1][2], while disk speeds have improved at a far slower rate. Consequently, the speed of the disk rather than the speed of the CPU's 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][4][5] and data contiguity/clustering [6][7][8]) 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 [9][10] 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 empha-

Figure 2: An example of object splitting.

3 Multimedia object/storage characteristics and problem formulation

The jth DE in a composite object O is denoted as o_j. Each DE is associated with a *frequency distribution*. This frequency distribution represents the relative probability that a given DE will be requested for retrieval. Each class-one DE, o_j, has an *expected retrieval rate*. 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 kth storage device is denoted as S_k, its bandwidth as $BW(S_k)$, total amount of space already allocated as S_k^a, and the free space as S_k^f. The total number of storage devices is represented by m. Since we are dealing with a heterogeneous environment where the computing system comprises different types of storage devices with different characteristics, the differing characteristic that is of paramount importance to us is the bandwidth of a storage device. We group related devices together based on their bandwidths into $\wp^1, \wp^2, \ldots, \wp^\mu$, where μ is the number of different bandwidths in the system and the bandwidth of each storage device in $\wp^i$ is $BW(\wp^i)$. We denote the kth AU of o_j as $a_{j,k}$ and the total number of AUs in o_j as α_j. Consequently, the storage allocation problem is formulated as follows: (1) Given a composite multimedia object, how can one decompose the DEs to build the AUs such that the allocation of the AUs achieve the real time requirements? (2) Given a list of AUs produced from the fragmentation strategy, how does one define the allocation process and AU allocatability? (3) Having determined the storage devices to which an AU is al-

locatable, what criteria are necessary and sufficient in determining the most efficient storage device to store an AU? (4) Given an allocation strategy, how can one demonstrate that it fairly and sufficiently balances the loads among the storage devices?

4 Problem and allocation analysis

Having defined the problem, it is important that we discuss the vital decisions that must be made to address the facets of the allocation process. Each allocation process comprises a composite object with its associated DEs. The storable elements are the DEs. A composite object is dynamically created and stored in the system with respect to the current status of the storage devices. For an allocation process, all the DEs of a composite object are considered simultaneously. Since the most common operation during the allocation process is the determination of the mappings of AUs to storage devices, we denote the fact that AU a is mapped to S_k as $a \models S_k$.

4.1 Intra DE allocation

The *intra DE allocation* stipulates the allocation policy that must exist when allocating the AUs of a DE. In the case of a class-one or class-three DE, it is imperative that each of its AUs be stored in a different storage device in order to achieve the expected retrieval rate or degree of parallelism. Therefore, if o_i is a class-one or class-three DE, then the intra DE allocation states that

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

4.2 Intra complex object allocation

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. In that case, therefore, it becomes necessary that all the DEs of a complex object be stored in such a way that all of its data can be retrieved concurrently. We must, 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* states that

$$\forall_{x_1=1,j}\ \forall_{x_2=1,\alpha_{x_1}} \text{ if } a_{x_1,x_2} \models S_h \text{ then } \forall_{x_3=1,j}\ \forall_{x_4=1,\alpha_{x_3}} \not\exists a_{x_3,x_4} \text{ such that } a_{x_3,x_4} \models S_h, \quad (x_3 \neq x_1, 1 \leq h \leq m).$$

4.3 Inter composite object allocation

Another access situation that we must consider is when the access crosses multiple composite objects. In an environment with a limitless number of storage devices, we can afford to store every AU in a different storage device. However, that situation is unrealistic. We, sometimes, expect the situation depicted in Figure 3 to occur where composite objects $O1$ and $O2$ share $O1$'s $o2$. For allocation purposes, we logically think of shared DEs as, physically, belonging to each of the composite object. For example, using Figure 3, if $O1$ were allocated first, then none of $O2$'s AUs should be allocated in the same device with $O1$'s $o2$.

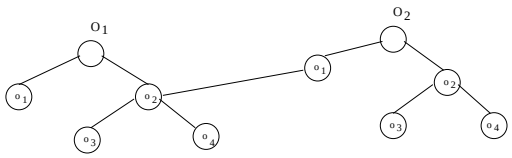

Figure 3: An example of object sharing.

5 Fragmentation strategies

For class-one 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 DE. We need to compute the number of storage devices that can be accessed in parallel to satisfy the retrievability requirement. A DE's *Storage Set* (χ) is the set of the number of storage devices needed to achieve its expected retrieval rate based on the amount of data that can be retrieved from each storage device per unit time (i.e., bandwidth) in parallel. If we have a homogeneous configuration of storage devices, then the computation of the number of storage devices needed is straightforward since all of the storage devices have the same bandwidth. In the case of heterogeneous storage devices, we have to consider different storage devices with different bandwidths. So we are forced to consider combinations of different storage devices with different bandwidths. Consequently, a DE can have multiple storage sets. An element of a storage set indicates the possible number of storage devices for one or more device clusters that is necessary to achieve the real time requirement of a DE. It is obvious that the number of sets in a storage set could become very large. As a result, some con-

straints, as described below, are utilized to minimize the size of a storage set. To that end, therefore, the number of sets in a DE's storage set is reduced to at most $2^{\mu} - 1$. Let $\chi_k = \{y^1, y^2, \ldots\}$ be the kth storage set of o_j where y^i is the number of storage devices from $\wp^i$ needed to achieve the expected retrieval rate. For each combination of device clusters that form a storage set, each cluster must be represented by at least one storage device. The amount of data retrievable from the storage devices of a storage set must not be less than the expected retrieval rate or degree of parallelism, but should exceed that value with minimum value. The storage device clusters are arranged in order of decreasing bandwidths. The number of storage devices per device cluster in a storage set decreases with increasing bandwidth, when applicable. In the rest of the paper, when necessary, the expected retrieval rate of a class-one DE o_j is represented as β_j. For example, given β_j and $\chi_k = \{y^1, y^2, y^3\}$ then the following conditions must hold:

1. $BW(\wp^1) > BW(\wp^2) > BW(\wp^3)$,
2. $y^1 \leq y^2 \leq y^3$,
3. $[y^1 BW(\wp^1) + y^2 BW(\wp^2) + y^3 BW(\wp^3)] \geq \beta_j$, and
   (a) $[(y^1 - 1)BW(\wp^1) + y^2 BW(\wp^2) + y^3 BW(\wp^3)] < \beta_j$,
   (b) $[y^1 BW(\wp^1) + (y^2 - 1)BW(\wp^2) + y^3 BW(\wp^3)] < \beta_j$,
   (c) $[y^1 BW(\wp^1) + y^2 BW(\wp^2) + (y^3 - 1)BW(\wp^3)] < \beta_j$.

If any of the conditions above is violated, then the corresponding storage set is invalid. The above conditions are equivalent to solving the integer linear programming problem:

$$y^1 BW(\wp^1) + y^2 BW(\wp^2) + y^3 BW(\wp^3) \geq \beta_j,$$
$$y^1 \leq y^2, \;\; y^2 \leq y^3, \;\; y^3 > 0.$$

An χ_k with $\mid \chi_k \mid^{1} = g$ is *acceptable* if
(1) $\sum_{i=1}^{g} y^i \leq m$, and
(2) $\forall_{j=1,g} \; y^j \leq \mid \wp^j \mid$.
For example, consider a class-one DE of size $120KB$ and bandwidth requirement of $60KB/s$, given that $\wp^1 = \{S_1, S_2, S_3\}$, $\wp^2 = \{S_4, S_5\}$, $\wp^3 = \{S_6, S_7, S_8\}$, $BW(\wp^1) = 30$, $BW(\wp^2) = 20$, and $BW(\wp^3) = 10$. The sets of the combinations of clusters of storage devices are $\{\wp_1\}$, $\{\wp_2\}$, $\{\wp_3\}$, $\{\wp_1, \wp_2\}$, $\{\wp_1, \wp_3\}$, $\{\wp_2, \wp_3\}$, and $\{\wp_1, \wp_2, \wp_3\}$. The valid storage sets are $\chi_1 = \{2\}$, $\chi_2 = \{3\}$, $\chi_3 = \{6\}$, $\chi_4 = \{1, 2\}$, $\chi_5 = \{1, 3\}$, $\chi_6 = \{2, 2\}$, and $\chi_7 = \{1, 1, 1\}$.

Obviously, χ_2, and χ_3 are not acceptable. Furthermore, without the constraints discussed above, for example, it is evident that given $\{\wp_2, \wp_3\}$, the storage sets $\{3, 0\}, \{1, 4\}, \{0, 6\}, \{2, 3\}$, and $\{1, 5\}$ can achieve the real time requirements. However, applying the constraints limits the option to $\{2, 2\}$. If none of the storage sets of a DE is acceptable, then we can not allocate the DE. When that happens, a message may be sent to the user suggesting a higher degree of data compression on the class-one DEs or a lower degree of parallelism for class-three DEs. Since the size of each data retrieved per unit time from each storage device is its bandwidth, each AU stored in a storage device comprises a number of chunks of data whose size is equal to the bandwidth of the storage device. We call each of this chunk of data a *Storage Element* (SE). An AU then consists of one or more SEs arranged in such a way that guarantees parallel retrieval of contiguous data. For a given DE, we denote as γ_k the number of storage devices in χ_k. We call γ_k the *storage length* of a storage set. Consequently, each χ_k comprises γ_k AUs where each AU is denoted as $a_{k,l}, 1 \leq l \leq \gamma_k$. We represent the number of SEs in $a_{k,l}$ as $\delta_{k,l}$. The sum of all the sizes of all the SEs of all the AUs of a DE must be at least as large as the size of the DE. Furthermore, reducing any AU of any storage set of a DE by one SE must violate the preceding condition. We denote as $\bowtie_{k,l}^{q}$ the fact that SE f_q belongs to AU $a_{k,l}$. Therefore, for χ_k

$$\text{if } \bowtie_{k,l}^{q} \text{ then } \forall_{(h \bmod \gamma_k \; = \; q \bmod \gamma_k)} \;\; \bowtie_{k,l}^{h} \quad (h \neq q).$$

As is evident from building the AUs, the data represented by each AU do not constitute a contiguous data in a DE. The physical addresses of the SEs in an AU differ by some factors of the bandwidths of the storage devices. This is a consequence of data interleaving which is essential for achieving parallel I/O for a stream of data. Figure 4 shows the SEs and AUs of all the storage sets. The numbers beside the boxes represent the physical addresses of the SEs in a DE.

The above discussion on fragmentation strategy has been done in the context of class-one DEs. In the case of class-three DEs, the degree of parallelism also represents the expected number of AUs. Therefore, for a class-three DE, a storage set is valid if its storage length is equal to the DE's

[1] $\mid A \mid \rightarrow$ the number of elements in set A

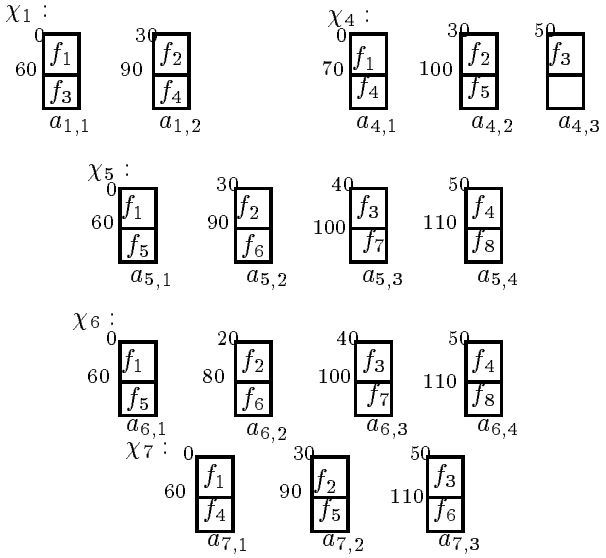

Figure 4: A sample generation of storage sets.

degree of parallelism. Furthermore, conditions (1) and (2) and the acceptibility requirement discussed above must hold. If the application of the above rules yields no storage set, then a storage set whose storage length minimally exceeds the degree of parallelism is selected. In the case of class-two DEs, each DE is made up of one storage set consisting of one AU.

6 The proposed mapping techniques

An AU is allocatable to a storage device if the storage device belongs to the device cluster from which the AU was built. In other words, the bandwidth of the storage device must be equal to the size of the AU's SE. 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 an 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 that 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 a_i be an AU, $SIZE(a_i)$ the size of AU a_i, and $FREQ(a_i)$ the frequency of AU a_i. The current *cumulative traffic requirement* of S_k, assuming that there are a total of h AUs already stored in it, is computed as:

$$S_k^w = \sum_{i=1}^{h} \mathcal{F}(FREQ(a_i), SIZE(a_i)).$$

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. Consequently, a reasonable motivation is to allocate the next AU to the storage device with lowest cumulative traffic requirement. However, that factor alone does not determine an efficient storage device to allocate an AU. In order to get a more vivid understanding of the effect of the cumulative traffic requirement, we need to determine the *expected disk traffic per unit of allocated space* in a storage device. That value indicates the disk traffic exerted per unit of allocated space in a given 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*. That is the expected disk traffic per unit of allocated space if the AU under consideration is allocated to a given storage device. We denote as $\mathcal{G}_k^w$ the induced expected disk traffic per unit of allocated space by an AU on S_k. After a successful allocation of an AU to S_k, S_k^w becomes $\mathcal{G}_k^w$. Consequently, for a given AU,

$$\mathcal{G}_{k,i}^w = \frac{S_k^w + \mathcal{F}(FREQ(a_i), SIZE(a_i))}{S_k^a + SIZE(a_i)}.$$

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. Again, the fact that a storage device has a high cumulative traffic requirement relative to another storage device should not imply an

automatic rejection of that storage device. If a storage device has a high cumulative traffic requirement but a high bandwidth, then the resultant effect of the cumulative traffic requirement is reduced by the fact that a large chunk of data is retrievable per unit time. Therefore, our mapping goal is to select the storage device that minimizes these factors. It is obvious that, in terms of magnitude, the bandwidth of a storage device is comparatively smaller than its total allocated space and free space (in most cases). Therefore, expressing the impacts of allocated space, free space, and bandwidth with respect to the cumulative traffic requirement requires that the impact from the bandwidth be expressed in such a way that it does not obscure its counterparts. The impact from the bandwidth should be related to the disparity between the bandwidths, i.e., if there is a considerable gap between the smallest and largest bandwidths of the storage devices under consideration, then the bandwidth factor should also reflect that. We represent the sum of these factors as a cost function ς. If

c_1 = cost induced by induced expected disk traffic per unit of allocated space,
c_2 = cost induced by free space with respect to the cumulative traffic requirement,
c_3 = bandwidth factor,
then,

$$\varsigma = (e_1 c_1 + e_2 c_2) \times c_3$$

where $c_1 = \mathcal{G}_k^w$, $c_2 = \frac{S_k^w}{S_k^f}$, and
$c_3 = 1 + \frac{(e_3 BW_{max} - BW_{S_k})}{e_3 BW_{max}}$.
BW_{max} is the maximum bandwidth of the storage devices allocatable to an AU. BW_{S_k} is the bandwidth of the storage device currently under consideration from the set of storage devices allocatable to an AU. 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. The coefficient e_3 is used to control the effects of the bandwidth factor and it is determined from the maximum and minimum bandwidths in the system. We recommend that e_3 be selected such that the bandwidth factor is in the range 1.5 to 1.9. In other words, $1.5 \leq 1 + \frac{e_3 BW_{max} - BW_{S_k}}{e_3 BW_{max}} \leq 1.9$. For example, given that $BW_{max} = 10MB$ and $BW_{min} = 50MB$, if $e_3 = 2$, then the bandwidth factor is in the range of 1.5 to 1.9. On the other hand, if $e_3 = 3$, then the bandwidth factor is in the range of 1.67 to 1.93. Given an AU and the ς costs of allocating it to different storage devices, we select the storage device with minimum cost. We denote the fact that an AU, a_i, is allocatable to the storage device, S_k, as $a_i \perp S_k$ and the size of each SE in a_i as $SESIZE(a_i)$. Therefore, $a_i \perp S_k$ if

1. $a_i \in \Re^{one}$ and
   (a) $SESIZE(a_i) = BW(S_k)$,
   (b) $S_k^f \geq SIZE(a_i)$.
2. $a_i \in \Re^{two} \vee a_i \in \Re^{three}$ and
   (a) $S_k^f \geq SIZE(a_i)$.

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,

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

For example, Table I shows the current values for 6 storage devices, and Table II shows the sizes and frequencies of 5 AUs of a composite multimedia object. After determining allocatabilities of the AUs and computing the ς costs, Table III shows the ς cost of each AU to the storage device to which it is allocatable. In this simple example, $e_1 = e_2 = 1$.

Table I: A sample of storage devices' current values used for allocation decision.

	S_1	S_2	S_3	S_4	S_5	S_6
Free space	10MB	100MB	500MB	100MB	1MB	15MB
Allocated space	5MB	120MB	10KB	500KB	50MB	70MB
Cur. cum. traffic reqs.	15728640	262144000	102400	1048576	125829120	104857600
Bandwidths	1MB	1MB	10MB	15MB	15MB	10MB

Table II: A sample of sizes and frequencies of some AUs.

	a_1	a_2	a_3	a_4	a_5
Size	50KB	75KB	1.5MB	100KB	3MB
Frequency	0.4	0.19	0.10	0.3	0.01

Table III: ς costs from Tables I and II.

	S_1	S_2	S_3	S_4	S_5	S_6
a_1	16.6	2810.0	-	-	-	-
a_2	17.5	2811.0	-	-	-	-
a_3	-	-	0.2	-	-	16.9
a_4	-	-	-	0.1	128.1	-
a_5	-	-	0.3	-	-	17.2

Consequently, applying the allocation and minimization rules, we have $a_1 \perp S_2$, $a_2 \perp S_1$, $a_3 \perp S_6$, $a_4 \perp S_4$, $a_5 \perp S_3$, with a total cost of 2844.8.

Figure 5: A sample mapping via Hungarian Method and an unallocatable bipartite graph.

8 Simulation model and results

We generated 3 groups of devices where each group has common characteristics such as the bandwidth and size. Each group comprises 10 storage devices. 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. Devices in group 1 have size of 100MB and bandwidth of 1MB, devices in group 2 have size of 75MB and bandwidth of 750KB, and devices in group 3 have size of 50MB and bandwidth of 500KB. 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. Table IV shows the data distribution by percentage in the storage devices after the allocation of the objects.

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.

Table IV: A sample percentage data distribution.

	.040	.053	.067	.080	.093	.110	.120	.130	.147	.160
1	6.3	4.6	4.6	4.1	0.0	2.4	2.6	3.7	3.5	2.9
2	6.9	4.2	5.7	0.0	6.6	3.5	2.2	1.6	2.6	2.4
3	6.8	4.8	5.6	4.1	3.4	2.8	2.5	3.2	3.2	5.9
4	6.1	4.9	5.5	3.6	5.8	4.3	4.4	2.1	6.9	8.4
5	6.0	4.0	4.1	7.5	3.6	1.2	4.7	4.9	4.1	3.5
6	6.8	4.8	3.0	2.2	2.3	5.4	6.5	1.3	1.1	8.3
7	6.8	5.2	0.0	2.5	0.0	3.4	3.9	1.2	5.3	1.0
8	6.6	4.7	7.1	1.9	1.5	1.5	4.9	2.4	1.5	0.9
9	6.6	5.5	0.0	3.8	3.6	1.3	1.3	2.2	0.9	0.9
10	6.4	5.1	9.3	2.5	2.2	4.5	0.0	0.5	1.5	0.0
11	1.4	3.7	2.3	3.5	4.2	1.6	6.7	4.3	4.8	0.8
12	2.5	3.8	0.0	6.1	5.0	4.1	2.3	1.1	1.7	7.5
13	1.4	3.0	1.4	4.3	7.6	8.4	2.5	11.1	8.1	10.2
14	1.4	3.7	3.1	2.3	5.8	4.0	6.0	4.2	1.6	1.6
15	2.0	2.4	0.0	9.8	0.0	5.8	0.0	1.7	3.4	3.3
16	2.8	2.8	7.9	0.0	0.0	0.0	2.5	1.9	3.0	1.5
17	1.4	3.3	2.9	5.0	10.2	13.3	9.2	11.2	10.2	6.7
18	2.1	2.4	0.0	1.0	0.0	2.5	2.6	0.3	2.2	0.0
19	2.3	2.8	5.5	2.3	2.2	2.8	1.5	5.9	2.7	4.4
20	2.3	2.2	2.3	0.7	2.9	4.6	6.0	4.7	2.2	2.2
21	1.8	1.4	3.0	0.0	5.3	2.6	3.2	2.5	0.0	2.7
22	1.5	1.6	5.2	0.0	1.5	1.2	2.9	8.7	3.3	1.4
23	0.7	2.0	2.6	4.1	0.0	1.6	4.7	0.0	4.0	0.9
24	1.3	2.5	2.1	4.4	2.4	2.2	3.5	1.2	0.0	1.9
25	1.1	2.2	3.2	2.9	2.0	0.0	5.2	6.9	2.8	7.5
26	1.6	1.0	5.4	2.9	5.8	2.4	0.8	1.5	6.3	4.4
27	1.8	2.0	2.8	4.5	3.4	1.4	0.0	0.7	0.7	3.2
28	1.6	2.7	3.3	3.4	4.1	1.4	0.9	1.1	1.0	1.5
29	1.9	2.5	1.0	5.5	1.7	6.8	3.1	4.5	10.4	2.6
30	1.8	2.0	0.9	5.1	7.1	3.1	3.4	3.2	1.1	1.2

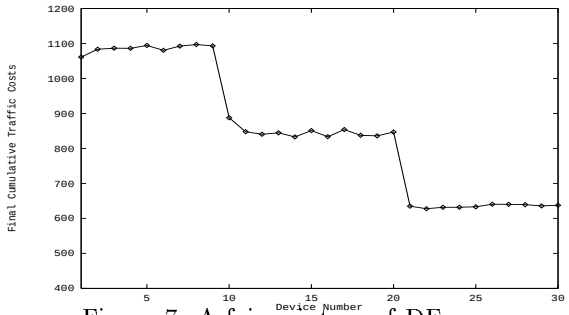

Figure 7: A fair mixture of DEs.

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. We classified the DEs of a composite object into three classes based on their I/O requirements. The allocatability requirements of an AU to storage devices are defined and the necessary and sufficient criteria for obtaining efficient allocations are described. A cost value based on a disk utilization per allocated space, the amount of free space, and the bandwidth of a storage device are used to determine an efficient allocation and to balance the loads on the storage devices. A bipartite graph model is presented and its characteristics discussed. The bipartite graph forms the basis for multimedia object allocation to storage devices. The Hungarian Method for bipartite matching is used to determine efficient allocation for the AUs of a composite object using the cost values.

References

[1] **C. G. Bell**, The mini and macro industries, *IEEE Computer*, Vol. 17, No. 10, 1984, pp. 14-30.

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

[3] **M. Y. Kim**, Synchronized Disk Interleaving, *IEEE Trans. on Computers*, Vol. C-35, Vol. 11, November 1986, pp. 978-988.

[4] **M. Livny, S. Khoshafian, H. Boral**, Multi-Disk Management Algorithms, *Proc. 1987 ACM SIGMETRICS Conf. on Measurement and Modeling of Comp. Syst.*, pp. 69-77.

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

[6] **J. Zupan**, Clustering of Large Data Sets, *Technometrics*, Vol. 29, Nov., 1987, pp. 497.

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

[8] **J.S. Deogun, V.V. Raghava, T.K.W. Tsou**, Organization of Clustered Files for Consecutive Retrieval, *ACM Trans. on Database Systems*, Vol. 9, December 1984, pp. 646-671.

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

[10] **P. V. Rangan, H. M. Vin**, Designing File Systems for Digital Video and Audio, *Proc. 13th ACM Symp. on Operating Systems Principles*, Vol. 25, No. 5, Oct., 1991, pp. 81-94.

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

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