

# Partitioning Multimedia Objects for Optimal Allocation in Distributed Computing Systems

Kingsley C. Nwosu  
IBM Corporation,  
POWER Parallel Systems Development,  
Large Scale Computing Division,  
MS/992, Kingston, NY. 12401. USA.

## Abstract

*Object partitioning is an essential mechanism to improving the performance of complex object-based systems. An essential step in developing an efficient information system in such a distributed environment is to optimally distribute data among the applicable sites. As a result, network communication overhead becomes a concern during object partitioning. For multimedia (complex) objects, the communication overhead is greatly affected by the way related or unrelated objects are aggregated for distribution. In this paper, we present two models of multimedia data models and discuss the techniques for efficient distribution of the objects. One model exploits the Mincut Algorithm and introduces the concept of the binding strength to interpret the relationships between objects in order to develop a linear time partitioning algorithm. The other model utilizes user specified or system determined degree of partitioning and bipartite matching to determine the aggregation and optimal allocation of objects.*

**Keywords:** bipartite matching, communication overhead, complex objects, distributed systems, efficient allocation, multimedia, object partitioning.

## 1 Introduction

The advent of multimedia information processing has pushed the object-oriented programming paradigm [1, 2] into becoming one of the most popular techniques for application and system development. Multimedia information processing involves the integrated handling of myriad of different data types derived from different media. As a result of the heterogeneity of the data types and the necessity to create a transparent view of the data types to the users and applications, multimedia data models are usually developed with the object-oriented approach. In order to capture the diversity of multimedia

data, *complex objects* are used to represent the different objects, sub-objects, and their relationships. When handling complex objects for either manipulation or presentation, a number of different problems arise. One of these problems is the storage allocation of the objects. The allocation problem concerns the way related or unrelated objects are stored in a given computing environment. The way objects are stored affects a system performance due the I/O bottlenecks. Most times, due to the sizes and retrieval requirements of the objects, it may be necessary to decompose some of the objects in order to store them for parallel retrievability. In a distributed environment, it becomes imperative that a complex object may have to be partitioned among the components of the distributed system. One of the goals of the partitioning is to minimize the network communication overhead that may otherwise develop if complex objects are allocated randomly to sites. Therefore, it becomes necessary that intelligent partitioning strategies should be utilized to generate the allocation units.

One of the ambitious goals of multimedia information processing technology is to be able to develop Multimedia Data Base Systems (MDBSs) that capture and satisfy the requirements of the conventional database systems as they pertain to multimedia data. These goals have been a great challenge due to the complexities introduced by continuous media such as audio/video. It is not yet possible to manipulate these types of media for conventional and expected database operations. Since the heterogeneity of systems and computing environments are becoming increasingly transparent to users, it is important that multimedia data models be developed that should be able to embed existing data models. The multimedia data models are expected to reflect the fundamental properties of the conventional models, i.e., non-multimedia data models.

Most of the meaningful work on data partitioning for distributed systems have been done in the context of distributed database systems [3, 4]. The partitioning and decomposition strategies proposed and utilized for database systems fit the monolithic nature of conventional data models. They are not sufficiently targeted towards complex objects with particular emphasis on multimedia data. Consequently, although the techniques are helpful for conventional database systems, they are inadequate and insufficient for complex objects. Of particular interest in complex object partitioning is the utilization probabilities between directly or indirectly related objects. Furthermore, although some object-oriented database systems [5, 6] have been developed, it can be argued that their object-orientedness stops at data representation but does not encompass data utilization. For system performance when complex objects are distributed, it should be an uncompromising requirement that inter-object utilization factor be considered during objects' decomposition or partitioning.

In this paper, we present two complex object data models for multimedia data, and the partitioning techniques for data distribution in a distributed system environment. Each model utilizes a different strategy for generating and optimally allocating the multimedia objects.

## 2 The Object Data Models

The clusterization techniques proposed are based on the multimedia data models described below. Our goal is to capture, as exhaustively as possible, the different data allocation conditions necessitated by different configurations of multimedia objects.

### 2.1 Attribute-Based Object Model (ABOM)

An attribute of an object defines its granularity and uniqueness. One of the primary goals of our model is to maintain the fundamental representational characteristics of conventional database models. Object-oriented complex object representations usually possess two fundamental dimensional components, namely, the *orthogonal* and *vertical* components. The orthogonal components are the *attributes* of an object while the vertical components are the sub-objects. The attributes are the storable data that define certain characteristics of a sub-object. A sub-object is a complex object with or without attributes and may in turn consist of sub-objects.

Therefore, our object data model is shown in Figure 1. It is a tree-structured collection of objects whose root node is called a *composite* object. Each internal node of a composite object is a complex object that consists of one or more attributes. The attributes are the leaf nodes of the complex objects which are shown as orthogonal entities in Figure 1. It is possible and permissible for an attribute to be decomposed into a complex object when necessary.

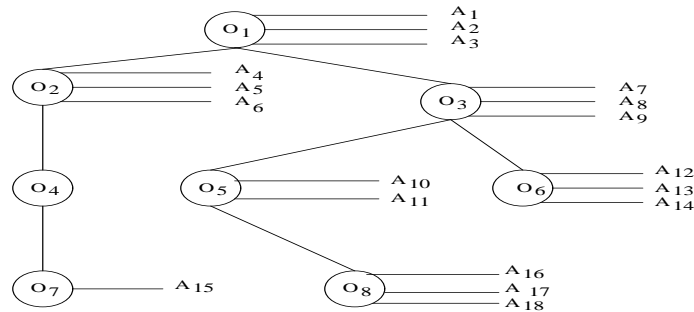


Figure 1: An example of the object data model.

An important feature of complex objects is object sharing. Shared objects logically belong to each of the parent objects, but are physically stored only once.

## 2.2 Media Objects Aggregation Model (MOAM)

The object model shown above is primarily tailored towards complex object database environments. We know that data allocations do not always require the granularity inherent or necessary in ABOM. For certain multimedia application environments, users may need to aggregate objects (complex objects) based on uniform or common perceptual and visionary properties. The Media Objects Aggregation Model (MOAM) allows users and applications to define hierarchical tree-structured object compositions based on some common characteristics. Figure 2 shows an example of MOAM object tree. It comprises a *composite object* ( $o1$ ) whose internal nodes are the *complex objects* ( $o2, o3, o6$ ) and the leaf nodes ( $o4, o5, o7, o8, o9$ ) are the useable and storable objects called *Data Elements* (DEs). Each collection of leaf nodes of a complex object constitutes a particular type of media object. The DEs of a composite object can be dynamically created per need basis. To a user, an object tree represents an instance of a multimedia session.

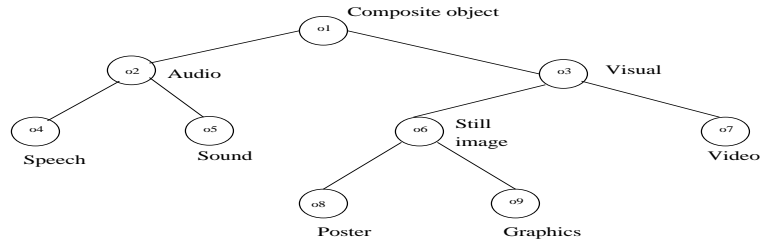


Figure 2: An example of a composite object tree

The distributed system environment consists of a Global Interconnection Network which comprises a number of network clusters (LANs). Each cluster comprises a number of sites. The  $i$ th site in a network cluster is represented as  $H_i$ .

## 3 Problem Formulation

One of the most perturbing problems with complex objects is that, since the objects have varying and differing characteristics and properties, in a computing environment that consists of a number of processing and storage nodes, we must define an efficient allocation strategy for data allocation with respect to some utilization criteria. The objectives being to maximize the utilization of the system resources, balance the loads on the nodes, and increase system efficiency. These issues give rise to the problem of clusterization. It defines conditions under which two or more objects should be allocated in a single node.

Therefore taking cognizance of the issues that have been discussed and the object and system models described, the clusterization problem can be stated as follows: Given a number of composite objects with the associated complex objects and attributes or DEs how do we

- determine the allocation units for each network cluster?
- determine the allocation unit for each site within a network cluster?
- determine the object characteristics and properties that are necessary and sufficient for developing the criteria for generating the allocation units?
- determine the effects of local or remote accessibilities?
- develop the efficient algorithms and processes to achieve our objectives?
- demonstrate the realization of our clusterization techniques?

## 4 Object Creation and Allocation Requirements

The objects of the models are dynamically created and stored in the system. At the creation of a DE, complex object, or attribute, the user or application must specify whether the object is limited to local access (*class-L object*), remote access (*class-R object*), or both accesses. Only local or remote access is legal; therefore, when an object has both accessibilities, remote access takes precedence.

### 4.1 Creation and Allocation Requirements of MOAM Objects

After creating a MOAM class-R composite object, complex object, or DE, the user or application must specify the *degree of allocation* for the applicable entity or entities. The degree of allocation specifies the maximum number of class-R objects of the target object that must be stored in a single node of a distributed system. There are two special cases of a degree of allocation, namely, when (1) the degree of allocation is one, i.e., each DE of a composite object must be stored in a different node. We call this *inter DE degree of allocation* (IDDA) and represent it as  $d^{idda}$ . (2) the degree of allocation requires that the class-R DEs of each complex object be stored in a different node. We call this *inter complex object degree of allocation* (ICDA) and represent it as  $d^{icda}$ . Other degrees of allocation are represented as  $d^{num}$  where  $num$  is the degree of allocation. For example, Figure 3 (left figure) shows a sample composite multimedia object and its allocation with IDDA. Figure 3 (right figure) shows the same composite object with ICDA. As an

example, using IDDA, if  $DE_1, \dots, DE_z$  are the DEs of a composite object, then if  $DE_k$  ( $1 \leq k \leq z$ ) is stored at a site  $H_j$  ( $1 \leq j \leq N$ ), then there should not exist  $DE_{k'}$  ( $1 \leq k' \leq z$  and  $k' \neq k$ ) such that  $DE_{k'}$  is also stored at site  $H_j$ . By replacing the DEs,  $DE_1, \dots, DE_z$ , with complex objects in the above condition, we obtain the allocation policy for the ICDA.

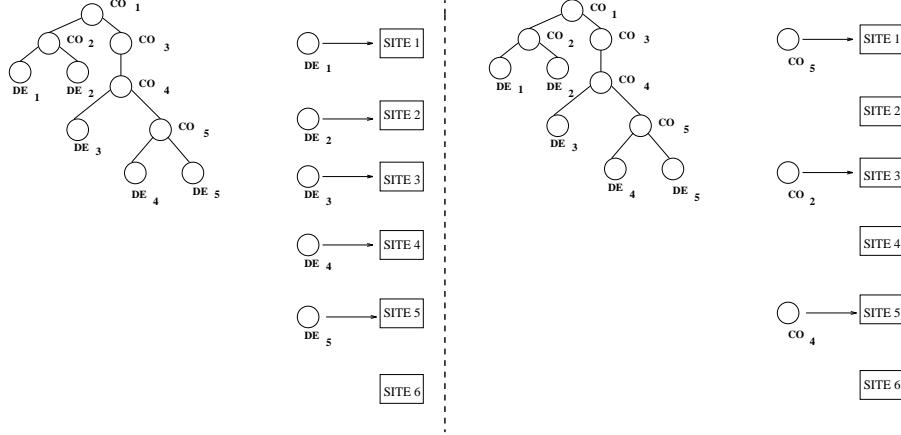


Figure 3: An example of an IDDA and ICDA.

For a composite object with considerable number of DEs relative to the number of target sites, the IDDA may not be possible.

## 4.2 Creation and Allocation Requirements of ABOM Objects

The creation of an ABOM composite object requires the specification of the constituent complex objects and corresponding attributes. The object creation command syntax tries to preserve the hierarchical relationships between the objects and their associated attributes. We enclose the immediate children of a complex object within a parenthesis, each child is separated with a comma, while the associated attributes immediately follow each complex object enclosed in brackets. When a complex object or attribute is for local access only, then we associate it with the symbols “%l”. We use the command **CREAT\_OBJ** to create a composite object. For example, let Figure 4 be a composite object that we need to create in our system. We can accomplish that by executing the command:

$$\mathbf{CREAT\_OBJ}(O_1[A_1, A_2](O_2[A_3, A_4, A_5], O_3(O_5[A_9, A_{10}], O_6[A_{11}, A_{12}, A_{13}\%l]), O_4[A_6, A_7, A_8\%l](O_7\%l [A_{14}, A_{15}](O_8[A_{16}, A_{17}]))))))$$

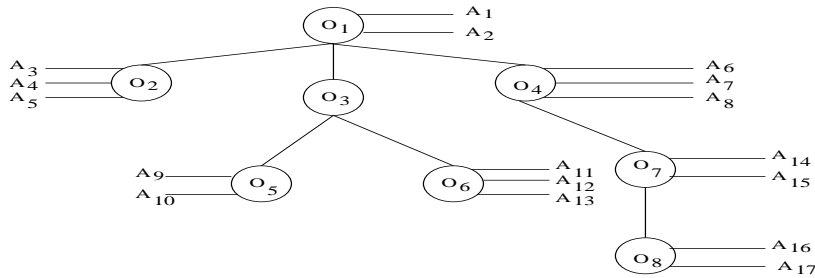


Figure 4: A composite object to be created.

In the case of shared objects, we precede each shared attribute with a string formed from the symbol “#” and the sharing objects.

## 5 Problem Analysis

Having presented the object and system models, and the allocation issues and problems, we now discuss in more detail the analyses of the partitioning objectives and goals. The primary problem in partitioning objects for distributed allocation is the determination of their clusterizations. When a composite object allocation spans multiple LANs, then two levels of allocation arise. At the first level, we partition objects for the network clusters, while at the second level, we partition objects for sites within a cluster.

### 5.1 The MOAM Objects

As we stated previously, we must specify the degree of allocation for class-R objects of a composite object for efficient allocation. Since each DE in class-R is susceptible to concurrent multiple accesses from any number of users, applications, or sites, the overwhelming motivation will be to utilize IDDA for each class-R object of a composite object. The cumulative number of target nodes is the most important limiting factor for the utilization of the IDDA.

#### 5.1.1 Inter DE and Complex Object Degrees of Allocation

There are certain conditions that must be satisfied for an effective utilization of either the IDDA or ICDA. If the number of class-R objects in a composite object is greater than the number of target nodes, then we cannot store the composite object with IDDA. If the number of complex objects in a composite object exceeds the number of nodes, then we cannot store the class-R objects of the composite object with ICDA.

### 5.1.2 Other Degrees of Allocation

The goal of other degrees of allocation is quite different from the goals of ICDA and IDDA. The objective of the other degrees of allocation is to store a number of DEs that belong to different complex objects into a single site. We denote as  $O^{\#cos}$  the number of complex objects in composite object  $O$  that have DEs associated with them. Given  $d^{num}$ , we expect that  $num \leq O^{\#cos}$ . Each DE in the set of DEs to be stored in a given site belongs to a different complex object. We need to generate sets of DEs such that each DE in a set comes from a different complex object. The number of the sets for the allocation of a composite object is limited by the number of target sites. The DE sets selected from the possible sets must cover all the DEs in the composite object. Let  $DS_i$ ,  $1 \leq i \leq m_{ds}$ , be a DE set that must be stored in a single site where  $m_{ds}$  is less than or equal to the number of target sites represented in a given number of network clusters. Therefore, stated formally, if  $DS_1, \dots, DS_{m_{ds}}$  are the DE sets for  $O$ , then

$$\begin{aligned} \text{if } DE_i \in DS_j \text{ (} 1 \leq j \leq m_{ds} \text{) then } \forall k [k = 1 \dots m_{ds}, k \neq j] \nexists DS_k \text{ such} \\ \text{that } DE_i \in DS_k \\ \text{and} \\ \forall DE \in O, DE \in \bigcup_{k=1}^{m_{ds}} DS_k \end{aligned}$$

The union of each  $DS_1, \dots, DS_{m_{ds}}$  yields a set that contains every DE of the composite object. Furthermore, each DE is a member of at most one DE set and none of the complex objects should have a number of DEs greater than the number of target sites. Under normal circumstance, in order to achieve the stated objectives, we expect that the number of target sites is greater than or equal to  $O^{\#cos}$ . There arises a situation when the number of applicable sites is less than  $O^{\#cos}$ . Under this condition, it may become impossible to obtain disjoint sets. However, our allocation strategy allows some flexibility with the satisfiability of the mutual exclusivities. The user is allowed to specify two or more complex objects that can be coalesced to form one complex object. We can still maintain the mutual exclusivity of the DE sets if and only if the number of elements in the coalesced sets does not exceed the number of target sites.

## 5.2 The ABOM Objects

Given some complex objects and associated attributes of an ABOM composite object, we know that related attributes, with respect to their complex objects, have higher degree of concurrent utilization or access. If we use the edges between two complex objects or a complex object and an attribute to denote logical distances, then the more the distance between two attributes, the less the probability that they may be concurrently requested. For example, in Figure 4, the distance between  $A_{10}$  and  $A_7$  is 5, i.e.,  $[(A_{10}, O_5), (O_5, O_3), (O_3, O_1), (O_1, O_4), (O_4, A_7)]$  and the distance

between  $A_{10}$  and  $A_{17}$  is 7, i.e.,  $[(A_{10}, O_5), (O_5, O_3), (O_3, O_1), (O_1, O_4), (O_4, O_7), (O_7, O_8), (O_8, A_{17})]$ . Therefore, with respect to  $A_{10}$ , there is a higher probability that  $A_7$  may be requested concurrently with it than  $A_{17}$ . The partitioning algorithm must then account for these relationships between attributes when generating the allocation units.

At each level of object distribution, we strive to generate allocation units that comprise related objects. By so doing, we do not disperse co-requisite objects thereby introducing unnecessary remote accesses which invariably affects system performance through network communication overheads. Due to the inherent nature of some of the multimedia objects, we assume that each site provides parallel retrieval environment for object decomposition to maximize I/O throughput. Examples of those allocation strategies can be found in [7, 8].

### 5.2.1 Utilizing Sub-graphs and Multigraphs

From the ABOM model, the elements of a composite object are represented as hierarchical entities which may share data among themselves. If we view the objects as nodes and object relations as edges, then our partitioning problem can be summarily viewed as finding sub-graphs of a multigraph. In order to obtain the sub-graphs, we need to flatten each composite object tree to show all the possible edge-relations between pairs of attributes. We label the edge  $(A_i, A_j)$  where  $1 \leq i < n$  and  $i < j < n$  with a weight value  $b_{i,j}$  which corresponds to the distance between the nodes. For example, Figure 5 shows a composite object and its multigraph. The weight on each edge

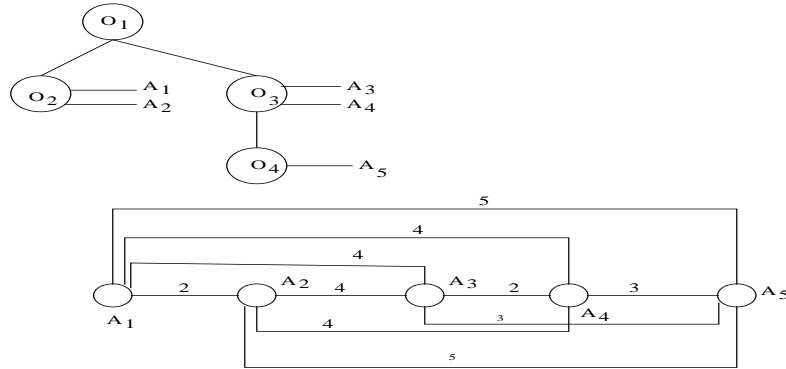


Figure 5: An example of a composite object and its multigraph.

between each pair of nodes indicates the distance between them. Therefore, we need to develop a cost function based on the weights and build a number of sets of nodes,  $G_1, G_2, \dots, G_m$ , such that the cumulative cost,  $\mathcal{C}$ , of the sets is minimized. That is, the sets are selected such that

$$\mathcal{C} = \sum_{i=1}^m \sum_{j=i+1}^m (w_{i,j}) \text{ is minimized, where } w_{i,j} = \sum_{k|A_k \in G_i} \sum_{l|A_l \in G_j} (b_{k,l}).$$

The exact solution of this problem is currently intractable in the sense that no polynomial-time algorithm for it is known to exist [9]. However, we use the Mincut Algorithm [10] to develop a strategy to achieve a nearly optimal solution.

## 6 Multimedia Object Partitioning Strategies

We now discuss in detail the different partitioning strategies necessary for each object model presented in order to efficiently distribute the applicable objects to the target sites.

### 6.1 Partitioning MOAM Objects

Determining the allocatability of a group of DEs necessitates the building of storage units. The storage units are the groupings, according to the degree of allocation, of the DEs that must be allocated per site. In the case of IDDA, the partitioning is as simple as extracting the leaf nodes (DEs) of a composite object as the example in Figure 6 (left) shows. The goal of ICDA is to partition a composite object according to the complex objects such that each storage unit contains only the immediate DEs of the complex object. For example, in Figure 6 (right) we want to be able to partition the complex objects to obtain the storage units shown.

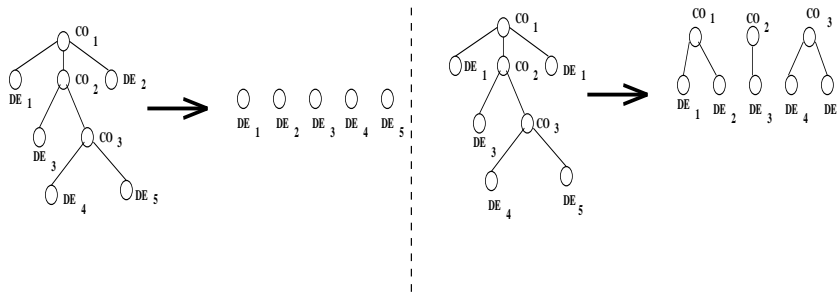


Figure 6: Composite object partitioning for IDDA and ICDA.

In order to systematically obtain our storage units for ICDA, we have to use a traversal algorithm. The traversal algorithm that best fits our purpose is the *Depth First Traversal* (DFT) algorithm [11]. The partition process is described recursively as in the algorithm below.

Algorithm I: A Depth First Traversal algorithm for object partitioning.

```

Algorithm DFT( $v$ )
/* Given a composite object tree with  $n$  nodes and an array VISITED[1 . .  $n$ ] initially set to zero
this algorithm visits all the nodes reachable from  $v$  and extracts the DEs associated with each
complex object. BUILD( $v, w$ ) associates the current DE,  $w$ , with its complex object,  $v$ . */
    VISITED[ $v$ ] = 1
    For every immediate child  $w$  of  $v$  do
        If  $w$  is a leaf node and  $w \in \text{class-R}$ 
            then
                If VISITED[ $w$ ] = 0
                    then
                        VISITED[ $w$ ] = 1
                        BUILD( $v, w$ )
                    endif
                endif
            else DFT( $w$ )
        endif
    endfor
end DFT

```

An analysis of Algorithm I shows that if  $n$  is the number of internal nodes in a composite object tree,  $G$ , and  $T(n)$  is the execution time with respect to  $n$ , then  $T(n) = O(n)$ .

### 6.1.1 Partitioning for Other Degrees of Allocation

From the discussions in Section 5.1.2, we know that in order to build the storage units for these degrees of allocation, we must generate and combine a number of sets of DEs. Precisely, we need to the following:

1. build the DEs of each of the applicable complex object,
2. generate combinations of DEs of different complex objects based on the degree of allocation specified,
3. select a number of sets from step 2 such that the resultant sets form a valid cover for all the applicable DEs,
4. if step 3 failed, then request for coalescing and go back to step 2 until coalescing becomes impossible.

The analysis of the process shows that the time of execution is bounded by the maximum number of sets generated. In other words, if  $d^{num}$  is the degree of allocation,  $O^{\#DEs}$  the number of DEs in a given composite object, and  $T(O)$  the time of partitioning  $O$ , therefore,

$$T(O) = \begin{cases} O(\lceil \frac{O^{\#DEs}}{num} \rceil^{num}) & \text{if } num \leq \frac{O^{\#DEs}}{2} \\ O(2^{O^{\#DEs} - num}) & \text{otherwise} \end{cases}$$

Therefore, generally speaking,  $T(O) = O(\lceil \frac{O^{\#DEs}}{num} \rceil^{\frac{O^{\#DEs}}{2}})$ .

## 6.2 Determining Optimal Allocation Sites

Having generated the storage units, our next important objective is to determine the optimal site to store each storage unit. Within each site, there is a cost induced on the storage devices by some amount of data. We denote as  $ec_{i,j}$ , the *expected cost* of allocating the DEs of the  $i$ th storage unit to  $H_j$ . The expected cost of a DE to a site is a function of the total expected costs of all the DEs already allocated to that site. There are several ways to determine this cost, e.g., [8]. Each site, therefore, computes the expected cost of allocating each storage unit to its storage devices. Obviously, a given storage unit may incur different costs at different sites. Our goal is to find the optimal site to store a storage unit with respect to the current utilizations of the applicable storage devices. Therefore, if we need to balance the utilizations of the storage devices, then we must choose optimal sites based on minimal expected cumulative costs of allocations.

Table II shows some examples of expected costs for the storage units shown in Figure 6 (left). A box with '-' implies that the corresponding DE is unallocatable to the corresponding site.

Table II: Examples of expected costs for the storage units in Figure 6 (left).

|        | $H_1$ | $H_2$  | $H_3$ | $H_4$  | $H_5$ | $H_6$ |
|--------|-------|--------|-------|--------|-------|-------|
| $DE_1$ | 16.6  | 2810.0 | 205.9 | -      | 23.5  | -     |
| $DE_2$ | 17.5  | 2811.0 | -     | 5123.9 | -     | 55.0  |
| $DE_3$ | 10.7  | -      | 0.2   | 23.5   | -     | 16.9  |
| $DE_4$ | 110.0 | -      | 10.3  | 0.1    | 128.1 | -     |
| $DE_5$ | -     | 256.0  | 0.3   | -      | 200.4 | 17.2  |

Applying the allocation and minimization rules, we have  $DE_1 \perp H_5$ ,  $DE_2 \perp H_1$ ,  $DE_3 \perp H_6$ ,  $DE_4 \perp H_4$ ,  $DE_5 \perp H_3$ , with a total allocation cost of 58.3.

Since our goal is to fairly distribute the loads across the sites, we have to select the mappings that yield minimal cumulative expected costs. This selection process is similar to the classical problem of the bipartite matching problem which has been applied to numerous problems such as the *max-flow problem* [12], *bipartite weighted matching problem* [13], also known as the *assignment problem*. Several algorithms have been developed to solve these problems; however, the one that best fits our problem is the *Hungarian Method* [13]. If we build a bipartite graph where the vertex partitions comprise the storage units ( $V_a$ ) and sites ( $V_s$ ), then there is an edge from a node in  $V_a$  to all the sites in  $V_s$  to which it is allocatable. We label each edge with the corresponding expected cost. Consequently, we apply the Hungarian Method to determine the efficient allocation of each storage unit. It tries to map the nodes of one side of the bipartite graph to the nodes of the other side with the aim of minimizing the cumulative weights of the mapped edges. During matching, the nodes are mapped on one-to-one basis and the Hungarian Method guarantees a solution if one exists. Figure 7 shows the bipartite graph built from Table II for an IDDA where the bold-face edges show the results of the optimal bipartite matching from the Hungarian Method.

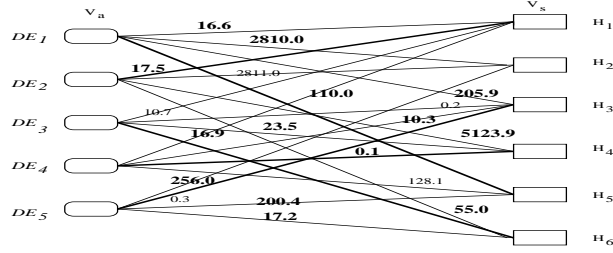


Figure 7: A sample bipartite graph and optimal allocations.

### 6.3 Partitioning ABOM Objects

In order to exploit the capabilities of the Mincut Algorithm, we use the concept of *Binding Strength* as a quantitative scale to interpret the utilization relationships between pairs of objects. The sets generated by the algorithm are used as the allocation units for either the network clusters or individual sites.

The Mincut algorithm method is a heuristic approach that has been applied to numerous graph or network based problems to partition a graph or network. In order to exploit the capability of the mincut algorithm, we define a binding strength between two objects to indicate the degree of closeness or relative co-usability between them. This value is captured by the  $b_{i,j}$ 's and so we assign the binding strength between  $A_i$  and  $A_j$  as  $b_{i,j}$ . The Mincut Algorithm accepts as input a multigraph with the associated binding strengths of the edges and the number of sets to generate. It produces the number of specified sets comprising disjoint nodes whose cumulative binding strength is the minimal possible.

Let  $m$  be the current number of network clusters in a given global network;  $n_{m,ds}^i$  the number of sites in the  $i$ th network cluster that have disk arrays;  $n_{s,ds}^i$  the number of sites in the  $i$ th network cluster without disk arrays; therefore, the partitioning algorithm is described as follows:

1. **begin**
2.     Apply the Mincut Algorithm to obtain  $m$  sets
3.     **for**  $i = 1$  to  $m$
4.         **begin**
5.             Group attributes according to disk configuration requirement
6.             Apply Mincut Algorithm for  $n_{m,ds}^i$  and  $n_{s,ds}^i$ .
7.             **for**  $k = 1$  to  $n_{m,ds}^i; n_{s,ds}^i$
8.                 **begin**
9.                      $G_k \Rightarrow (LAN\ k \vee H_k)$
10.                 **end**
11.             **end**
12.     **end**

An analysis of the algorithm shows that it has a complexity of  $O(N \log m)$  which is the complexity of the Mincut Algorithm. It is important to know

that the partitioning on the site level is done in parallel by applicable sites within each cluster.

## 7 Simulation Models

Due to the differences in the object models and their partitioning strategies, we use two different simulations to understand the degrees of objects distribution. In each case, we want to get a better vivid understanding of the objects distribution with respect to the status of the storage devices in the target sites.

### 7.1 The ABOM Objects

We need to analyze the average binding strengths within each cluster or site. The experiments consist of a number of composite objects whose constituent number of complex objects and attributes are randomly generated. In the first case, we want to observe the distribution of objects with respect to each network cluster and then, with respect to each site. We use the average cumulative binding strength of all the objects allocated in a cluster to determine the degree of distribution. In the second case, we want to compare our partitioning strategy with commonly used methods, i.e., either sequentially allocating objects or size-balancing allocations.

#### 7.1.1 Simulation Results

Figure 8 (left) shows the final average binding strength in each network cluster. The results show that, with respect to the average distance between objects allocated to a cluster, the binding strengths are relatively uniform. They point to the fact that within a cluster, objects that are closer to each other are co-allocated. In the second case, the same objects in the first case were applied to the sequential and size-balancing approaches. Figure 8 (right) shows the results of the average binding strengths among the three methods. The proposed strategy yielded the smallest average bind-

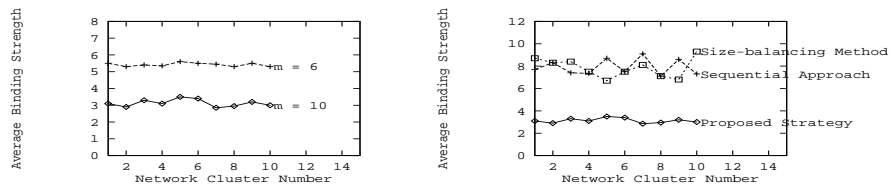


Figure 8: Object distribution (left) and Comparative object distribution (right).

ing strength to indicate that related objects are more closely allocated than

the other two methods since our strategy tries to generate optimal sets for allocation.

## 7.2 The MOAM Objects

We use the final cumulative costs of the storage devices in each site to determine the extent of distribution. The simulation comprises the generation of a number of composite objects with varying number of DEs. The size of a DE (between 1 byte and 10MB) is randomly determined. The expected cost of a DE is determined as a function of its size and the current loads on the target site. We defined a coefficient  $e$  such that if  $DE_i^{size}$  is the size of the  $i$ th DE, and there are  $n$  DEs, then  $\forall k [1 \dots n] e > DE_k^{size}$ . If  $H_i^{cum}$  is the current cumulative expected costs of  $H_i$ , then  $ec_{k,j} = e^{-1} \times DE_k^{size} \times H_i^{cum}$ .

### 7.2.1 Simulation Results

Figure 9 shows the graph of the final cumulative cost per site in the cluster. The figure shows that, within the cluster, the loads among the storage devices are relatively the same. Taking cognizance of the importance and meaning of the allocation parameters and criteria, it becomes obvious that the allocation process has distributed the objects such that all the sites have relatively equal loads.

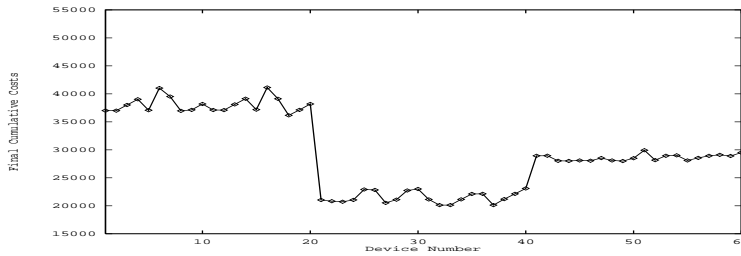


Figure 9: Load distribution in a distributed environment.

## 8 Conclusions

This work points to the necessary need of partitioning complex objects with cognizance of their relative utilizations or inherent allocation requirements. By using the concept of the binding strength derived from the hierarchical distance between pairs of objects, we capture the relative probability of co-requisiteness between pairs of objects. The almost uniform average binding strengths observed from the experiments indicate that the technique efficiently partitions objects of complex objects for balanced distribution among the targets.

## References

- [1] **Oscar Nierstrasz**, A Survey of Object-Oriented Concepts, *Object-Oriented Concepts, Databases, and Applications*, pp. 3-22, Addison-Wesley, Reading, MA, 1989.
- [2] **Bjarne Stroustrup**, *The C++ Programming Language*, Addison-Wesley, Reading, MA, 1986.
- [3] **Douglas W. Cornell, Philip S. Yu**, On Optimal Site Assignment for Relations in the Distributed Database Environment, *IEEE Trans. on Software Engineering*, Vol. 15, No. 8, pp. 1004-1009, 8/89.
- [4] **D. Cornell, P. S. Yu**, A vertical partitioning algorithm for relational databases, *Proc. 3rd Int. Conf. Data Engineering*, Los Angeles, CA. pp. 30-35, 2/87.
- [5] **D. H. Fishman, et al**, Overview of the Iris DBMS, *Object-Oriented Concepts, Databases, and Applications*, pp. 219-250, Addison-Wesley, Reading, MA, 1989.
- [6] **Won Kim, et al**, Features of the ORION Object-Oriented Database, *Object-Oriented Concepts, Databases, and Applications*, pp. 251-282, Addison-Wesley, Reading, MA, 1989.
- [7] **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.
- [8] **C. Y. R. Chen, Kingsley C. Nwosu, P. Bruce Berra**, Multimedia Object Modeling and Storage Allocation Strategies, *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, Boston, MA., Vol. 3, No. 3/4, June 1994.
- [9] **Michael R. Gray, David S. Johnson**, *Computers and Intractability*, W. H. Freeman and Company, 1979.
- [10] **B. W. Kernighan, S. Lin**, An efficient heuristic procedure for partitioning graphs, *Bell System Technology Journal*, Vol. 49, pp. 291-307, 2/70.
- [11] **Ellis Horowitz, Sartaj Sahni**, *Fundamentals of Computer Algorithms*,
- [12] **T.H Cormen, C.E Leiserson, R.L Rivest**, *Introduction to Algorithms*, The MIT Press, 1990.
- [13] **C.H. Papadimitriou, K. Steiglitz**, *Combinatorial Optimization*, Prentice-Hall, 1982.