

# On Complex Object Distribution Technique for Distributed Computing Systems

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

## Abstract

Object partitioning is an essential mechanism for improving the performance of object-based systems. However, in most of the work to date, emphasis has been confined to the optimization of disk *rotation* and *seek* time. The trend in the development of advanced information systems has become more and more towards distributed environments. In a distributed environment, multiple loosely-coupled (or even independent) systems are interconnected through communication networks, each, possibly, with its own local disks. An essential step in developing an information system in such a distributed environment is to efficiently distribute data to disks attached to the sites. Therefore, network communication overhead becomes an important concern during object partitioning. In this paper, we exploit the Mincut algorithm and introduce the concept of the binding strength to interpret the relationships between objects in order to develop a linear time partitioning algorithm. The algorithm partitions the objects of a complex object with the aim of minimizing the network communication overhead. The experimental results show a very promising performance advantage in comparison with other methods.

**Keywords:** binding strength, communication overhead, complex objects, distributed systems, multimedia objects, object partitioning.

## 1 Introduction

The advent of multimedia information processing has pushed the object-oriented programming paradigm [1]-[3] into becoming one of the most popular techniques for application and system development. Multimedia information processing involves the integrated handling of myriad of 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. A complex object is a tree-structured collection of objects with differing characteristics and properties whose nodes are associated with private and/or inherited attributes or properties. 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's performance due to 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. One of the goals of the partitioning strategy is to minimize the network communication overhead that may otherwise develop if complex objects are allocated as a unit to sites.

One of the primary objectives for multimedia information processing is to be able to ultimately develop a multimedia database system that captures and satisfies the requirements of the conventional database systems as they pertain to multimedia data. This goal is a great challenge due to the complexities introduced by continuous media such as audio/video. It is not yet possible to manipulate those 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 incorporate the capabilities of the existing data models.

Most of the meaningful work on data partitioning for distributed systems have been done in the context of distributed database systems [4]-[6]. 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. Of particular interest in complex object partitioning is the utilization probabilities between directly or indirectly related objects. Furthermore, although some object-oriented database systems [7, 8] have been developed, it can be argued that their object-orientedness stops at data representation, but does not encompass data utilization implications.

In this paper, therefore, we present a complex object data model for multimedia data, and the partitioning techniques for data distribution in a distributed system environment. We employ the Mincut approach for generating the storage units for the components of a distributed system.

## 2 The object data model

We now present the description of the object data model on which the data partitioning to be proposed is based. 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. Using the example in Figure 1,  $O_1$  is the composite object;  $O_2, \dots, O_8$  are the

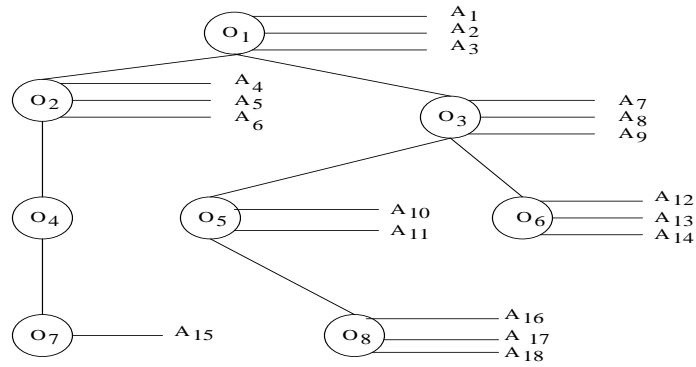


Figure 1: An example of the object data model.

complex objects of  $O_1$ ;  $A_1, \dots, A_3$  are the attributes of  $O_1$ ;  $A_{16}, \dots, A_{18}$  are the attributes of  $O_8$ ; etc. Furthermore, as shown in Figure 2, the attribute  $A_{18}$  of  $O_8$  can be decomposed to form the complex object  $O_9$  with new attributes  $A_{18}$  and  $A_{19}$ .

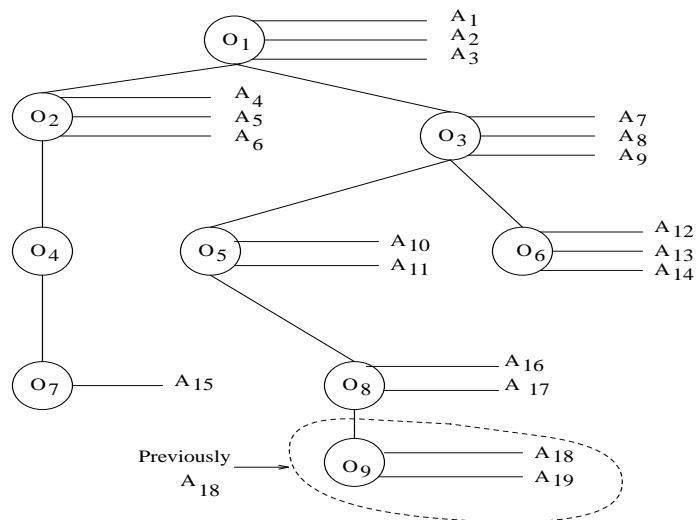


Figure 2: An example decomposition of an attribute.

An important feature of complex objects is object sharing. Sharing can occur at complex object or attribute level. Shared objects logically belong to each of the parent objects, but are physically stored only once. For example, Figure 3 shows that  $O_2$  shares  $A_3$  with  $O_4$  and  $A_7$  with  $O_6$ . Note that more than two objects can share an attribute. In the rest of the discussion, we may also refer to attributes as simply objects.

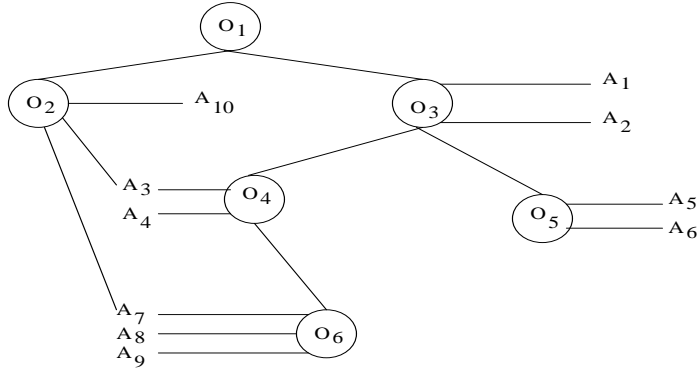


Figure 3: An example of data sharing.

In order to create a composite object, the constituent complex objects and attributes are initially defined as dummy entities. The partitioning strategies to be described are then applied on the elements of the composite object to determine the allocation units. A composite object can grow or shrink at any time. After a composite object has grown by one or more attributes or complex objects, it will be necessary to re-partition the composite object for purposes of determining the optimal allocation of the new components with respect to existing ones. The re-partitioning does not invalidate other stored objects.

### 3 The distributed system model

In order to get a full understanding of the partitioning strategies to be proposed, we briefly describe the envisioned distributed system environment. The distributed system environment, as shown in Figure 4, consists of a Global Interconnection Network and a number of network clusters. Each cluster comprises a number of sites. Each site within a cluster

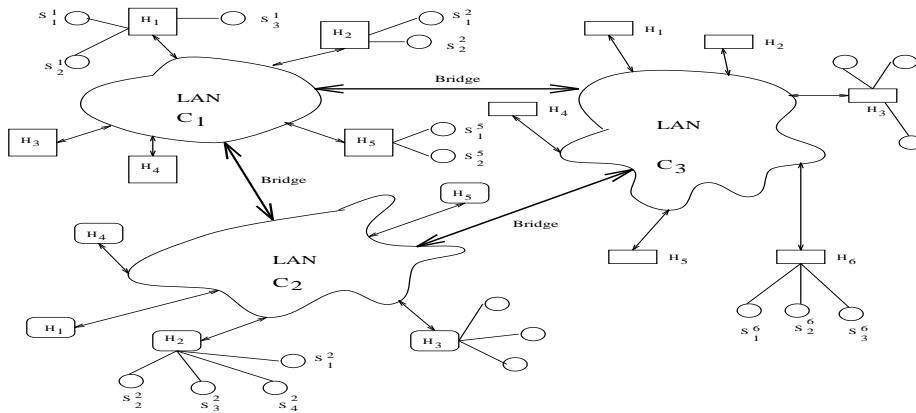


Figure 4: An example of the distributed system environment.

may contain some storage devices for storing the objects. Figure 4 shows one such global network with 3 network clusters  $C_1$ ,  $C_2$ , and  $C_3$ . It is important to state categorically that

in partitioning the complex objects for the distributed systems, we do not account for some of the network issues such as bandwidths, delays, etc. We assume that those issues are better addressed by each specific allocation strategy within a given site. The  $i$ th site in a network cluster is represented as  $H_i$ .

## 4 Problem formulation

Taking cognizance of the issues that have been highlighted and the object and system model described, the partitioning problem can be stated as follows: Given a number of composite objects with the associated complex objects and attributes how do we

- determine the allocation units for each network cluster? An allocation unit is a subset of the attributes of a composite object that must be allocated within a single target.
- 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?
- develop the efficient algorithms and processes to achieve our objectives?
- demonstrate the realization of our partitioning objectives?

## 5 Object creation and accessibilities

Before a composite object can be partitioned or requested for manipulation, it must have been previously created. The creation of a composite object requires the specification of the constituent complex objects and corresponding attributes. The corresponding data for the attribute values need not exist at this time. Since one of the important goals of object partitioning is to distribute data in order to minimize network communication overhead, it becomes imperative that we must strive to distinguish between the objects that are destined for local and/or remote accessibilities. Objects that are destined for local consumption do not participate in the partitioning process for generating the allocation units. In the case that an object or attribute has both accessibilities, remote access takes precedence.

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 5 be a composite object that we need to create in our system. We can accomplish that by executing the command:

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

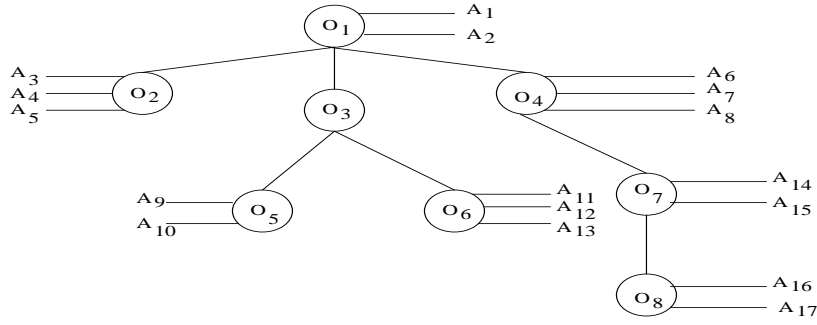


Figure 5: A composite object to be created.

Consequently, complex object  $O_7$  and attributes  $A_{13}$  and  $A_8$  will be allocated in the local site. They will not belong to any of the allocation units to be generated from the partitioning process.

In the case of shared objects, we precede each shared attribute with a string formed from the symbol “#” and the sharing objects. For example, using Figure 3, we create the composite object thus:

$$\mathbf{CREAT\_OBJ}(O_1(O_2[\#O_4A_3, \#O_6A_7, A_{10}], O_3[A_1, A_2](O_4[A_3, A_4] \\ (O_6[A_7, A_8, A_9]), O_5[A_5, A_6])))$$

## 6 Problem analysis

Having presented the object and system models, 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 how to determine their clusterizations. Given some complex objects and their associated attributes, 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 5, 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.

Judging from our distributed system model, two levels of allocation decisions must be made. At the first level, we partition objects for the network clusters, while at the second level, we partition objects for sites. At each level, 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 [9]-[11].

## 6.1 Utilizing subgraphs and multigraphs

As is evident from our object data 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 subgraphs of a multigraph. The subgraphs are formed such that the objects of a subgraph form an allocation unit for either a network cluster or site, and also the allocation of the units achieves the stated objectives. In order to obtain the subgraphs, it is necessary that we flatten each composite object tree to show all the possible edge-relations between pairs of attributes. Obviously, there exists an edge between every pair of nodes. Given that a multigraph has  $n$  nodes, there are at most  $\frac{n(n-1)}{2}$  edges. We label the edge  $(A_i, A_j)$  where  $1 \leq i < j < n$  with a weight value,  $b_{i,j}$ , that corresponds to the distance between the nodes. For example, Figure 6 shows a composite object and its multigraph. The weight on each edge 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.

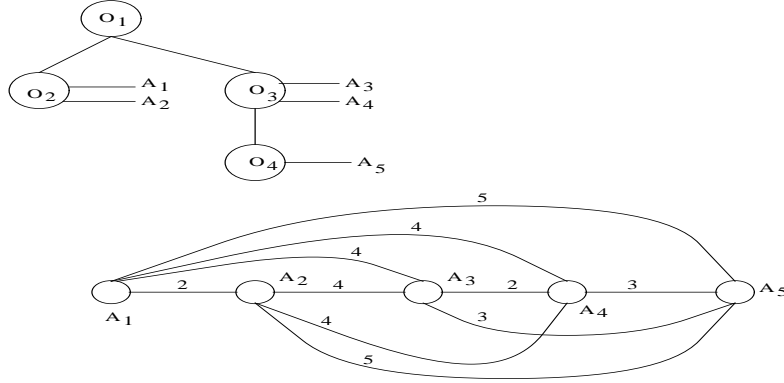


Figure 6: An example of a composite object and its multigraph.

That is, the sets are selected such that

$$\mathcal{C} = \sum_{i=1}^m \sum_{j=i+1}^m (w_{i,j}) \quad (1)$$

is minimized, where

$$w_{i,j} = \sum_{k|A_k \in G_i} \sum_{l|A_l \in G_j} (b_{k,l}). \quad (2)$$

The exact solution of this problem is currently intractable in the sense that no polynomial-time algorithm for it is known to exist [12]. However, we use the Mincut Algorithm [13] to develop a strategy to achieve an efficient solution.

## 7 The proposed approach

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.

### 7.1 The mincut algorithm

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. Obviously, from our previous discussions, this value is captured by the  $b_{i,j}$ 's. Therefore, we assign the binding strength between  $A_i$  and  $A_j$  as  $b_{i,j}$ . The binding strengths are the weights used on a multigraph to exploit the Mincut heuristic graph partitioning technique. 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.

### 7.2 Global and site-specific partitioning

In our previous discussion, we stipulated that two levels of partitioning are necessary to completely distribute the objects of a composite object in our distributed environment. The first involves the partitioning of objects with respect to the network clusters, while the other involves the individual sites within a network cluster. When mapping an allocation unit to a network cluster, it is necessary that there exists enough free space in the sites that comprise the cluster to accommodate the unit. Therefore, given that  $D_i^{free}$  is the cumulative free space in the  $i$ th cluster;  $S_i$  is the size of  $A_i$ ;  $G_1, \dots, G_m$  are the allocation sets for  $m$  clusters/sites generated from a minimization of the cost function  $\mathcal{C}$ ; and that we represent as  $G_i \Rightarrow H_j$  and  $G_i \Rightarrow C_j$  the fact that  $G_i$  is mapped to  $H_j$  or  $C_j$ , respectively. Therefore, the following constraint also applies:

$$\forall i [i = 1 \dots m] \text{ if } G_i \Rightarrow (H_j \vee C_j), 1 \leq j \leq m \text{ then } (\sum_{A_k \in G_i} S_k) \leq D_j^{free}.$$

The site partitioning handles the specific allocation of objects to individual sites within a cluster. In doing so, we have to recognize two conditions due to the sizes of objects and bandwidths of storage devices. It is obvious that some of the multimedia objects are enormously large and require very high data availability. In order to achieve the data availability rates of these objects, it may be necessary to fragment them for disk arrays that foster parallel retrievability. As with the global partitioning and disk space availability, the same condition must be satisfied for site allocations.

### 7.3 The partitioning algorithm

The two-level partitioning algorithm described in the previous sections, is then derived as follows:

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.

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.                      $G_k \Rightarrow (C_k \vee H_k)$
10.                 **end**
11.             **end**
12.     **end**

In step (2) of the algorithm, we utilize the Mincut Algorithm to determine the optimal sets for the network clusters. In step (5), for each set generated, we group the attributes according to their data availability rates, i.e., we group together those objects that require parallel disk configuration to satisfy their expected retrieval rates. In step (6), we apply

the Mincut Algorithm again to the groups produced to generate optimal allocations for the applicable sites. Finally, in step (9), we assign each set to the appropriate site. It is necessary to point out that we may not always generate a number of sets equal to the target sites or clusters; it depends on the number of objects being handled.

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.

## 8 Simulation model

In order to verify our partitioning technique, we need to analyze the average binding strengths within each cluster or site. The experiments consist of a number of composite objects whose number of complex objects and attributes are randomly generated. The accessibility of an object, i.e., remote or local, is also randomly determined with higher probability given to remote accessibility. To control the size of the composite objects, each composite or complex object is limited to a specific maximum number of attributes. Each composite object is independently partitioned for allocation; however, more than one partitioning can be done concurrently since we are operating in a distributed environment. In order to get a better understanding of the actual behavior of our strategy, two modes of experiments are conducted.

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. The binding strength used in analyzing the results are the binding strengths between each pair of objects within a cluster or site. Furthermore, the average binding strengths between different sites in a cluster are used to determine the degree of distribution among the sites. Due to the bias introduced by the co-existence of sites with or without disk arrays, we intentionally analyzed clusters with only disk arrays or without disk arrays. This is only to help understand the degree of distribution and does not in any way affect the actual configuration of sites in real environments.

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. In the sequential allocation, given that there are  $g$  objects and  $h$  targets (sites or clusters), the first  $\frac{g}{h}$  objects are allocated to the first target; the second  $\frac{g}{h}$  objects to the second target, etc. In the size-balancing approach, the total size of the objects to be allocated is equally distributed among the targets. For example, if *size* is the total size of the objects to be allocated among  $h$  targets, then the first number of objects whose cumulative size is less than or equal to  $\frac{size}{h}$  are allocated to the first target and the second number of objects whose cumulative size does not exceed  $\frac{size}{h}$  are allocated to the second target, etc. We use the average binding strength of allocated objects to determine the degree of distribution.

### 8.1 Simulation results

In the first case experiment, we used a global network that comprises 10 network clusters where the size of each cluster ranged between 1 and 40 sites. A total of 1,000 composite

objects where each composite object consisted of between 1 and 50 attributes were used. Figure 7 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 conducting different experiments, we noticed that the average binding strength within a cluster increases as the number of clusters gets smaller. This is explainable by the fact that, as the number of sets to be generated gets smaller, the size of each set gets bigger which invariably means that objects that may have considerable distance between them may end up in the same set. When that happens, the average binding strength increases. Figure 7 shows both behavior, i.e., the graph with higher average binding strengths among the clusters was obtained when the number of clusters decreased from 10 to 6. In the second case, the same objects in the first case were applied to the sequential and

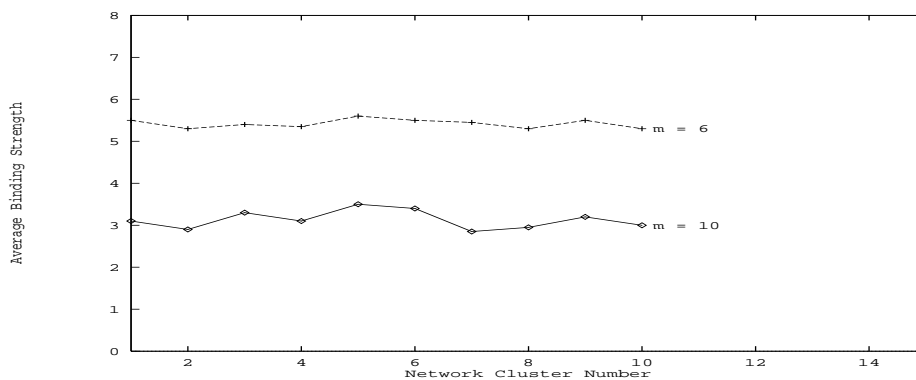


Figure 7: Proposed strategy’s object distribution within clusters.

size-balancing approaches. Figure 8 shows the results of the average binding strengths among the three methods. The proposed strategy yielded the smallest average binding 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. We are not concerned about the relative behaviors between the sequential and size-balancing methods; we are only reporting their observed behavior in relation to the proposed technique. The preference of the proposed strategy then becomes evident and obvious.

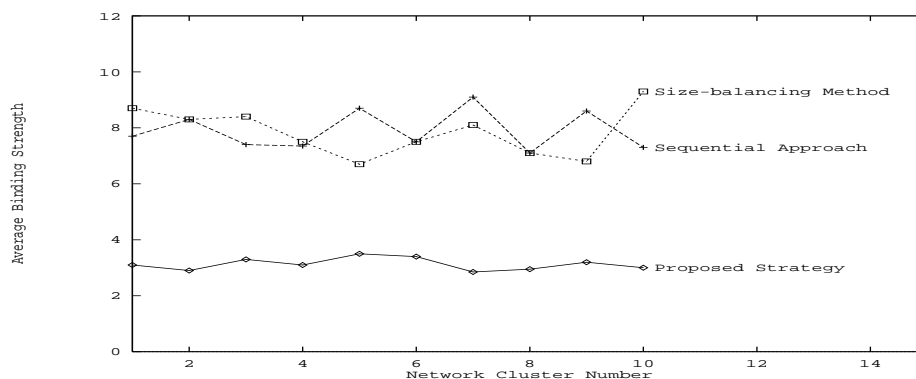


Figure 8: Comparative object distribution with other techniques.

## 9 Conclusions

In this paper, we have proposed an object data model that captures many of the fundamental representational and storage properties of the primary conventional database data models, and also a distributed system model. We described the essential characteristics of the objects in terms of their accessibilities and creation. We applied the Mincut algorithm through the utilization of multigraphs to generate subgraphs that represent optimal collection of objects that should be co-allocated within a network cluster or site.

This work points to the necessary need of partitioning complex object with cognizance of their relative utilizations. 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] **Adele Goldberg, David Robson**, *Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading, MA, 1983.
- [4] **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.
- [5] **D. Cornell, P. S. Yu**, Site assignment for relations and join operations in the distributed transaction processing environment, *Proc. 4th Int. Conf. Data Engineering*, Los Angeles, CA. pp. 100-108, 2/88.
- [6] **D. Sacca, G. Wiederhold**, Database partitioning in a cluster of processors, *IBM Res. Rep.* RJ4076, 10/83.
- [7] **Won Kim, F. H. Lochovsky**, Features of the ORION Object-Oriented Database, *Object-Oriented Concepts, Databases, and Applications*, pp. 251-282, Addison-Wesley, Reading, MA, 1989.
- [8] **Robert Bretl, et al**, The GemStone Data Management System, *Object-Oriented Concepts, Databases, and Applications*, pp. 283-308, Addison-Wesley, Reading, MA, 1989.
- [9] **M. Y. Kim**, Synchronized Disk Interleaving, *IEEE Trans. on Computers*, Vol. C-35, Vol. 11, pp. 978-988, November 1986.
- [10] **P. Chen, D. Patterson**, Maximizing Performance in a Striped Disk Array, *Proc. 1990 ACM SIGARCH 17th Intern. Symp. on Comp. Arch.*, pp. 322-331, Seattle, WA, May 1990.

- [11] **C. Y. R. Chen, Kingsley C. Nwosu, P. Bruce Berra**, Multimedia Object Modeling and Storage Allocation Strategies for Heterogeneous Parallel Storage Devices in Real Time Multimedia Computing Systems, *Proc. IEEE 17th Annual International Computer Software and Applications Conference (COMPSAC)*, Phoenix, Arizona, pp. 216-223, 11/93.
- [12] **Michael R. Gray, David S. Johnson**, Computers and Intractability, *W. H. Freeman and Company*, 1979.
- [13] **B. W. Kernighan, S. Lin**, An efficient heuristic procedure for partitioning graphs, *Bell System Technology Journal*, Vol. 49, pp. 291-307, 2/70.