An Efficient Partitioning Strategy for Complex Objects in Distributed Computing Systems

Cheng-Sheng Martin Chang,
Dept. of Computer & Info. Sciences,
Syracuse University,
Syracuse, NY 13244. USA.
email: cschang@cat.syr.edu

Kingsley C. Nwosu,
IBM Corporation,
Large Scale Computing Division,
Neighborhood Rd., MS/992,
Kingston, NY 12401. USA.
email:nwosuck@donald.aix.kingston.ibm.com

Abstract

Object partitioning is an essential mechanism in 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. 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 a concern. In this paper, we present a complex object data model for multimedia data and the partitioning techniques for object distribution. We introduce a concept to capture the possible co-utilization degree between objects and use that concept to partition objects for optimal allocation to network nodes in order to minimize network overhead, with respect to requests traffic, when requesting for the objects.

Keywords: communication overhead, complex objects, databases, distributed environment, 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 for 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, subobjects, 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. Our allocation problem addresses the issue of how related and unrelated objects are stored in a given computing environment. The way objects are stored affects a system 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. 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 traffic that may otherwise develop if complex objects are allocated at random to sites.

One of the ultimate goals of multimedia information processing technology is to be able to develop a multimedia database system that captures and satisfies the requirements of the conventional database systems as they pertain to multimedia data. These expectations have been 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 is becoming increasingly transparent to users, it is important that multimedia data models be developed that can sup-

port existing data models. The data models are expected to reflect the fundamental properties of the conventional models for non-multimedia data.

Most of the meaningful work on data partitioning for distributed systems has been done in the context of distributed database systems [3, 4, 5]. 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 [6, 7] 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 factors be considered during objects' decomposition or partitioning.

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 Min-cut 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 we propose is based. As we mentioned earlier, one of the primary goals of our model is to maintain and incorporate 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 that 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

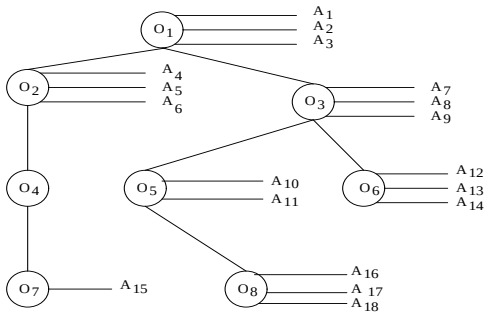

Figure 1: An example of the object data model.

in Figure 1, O_1 is the composite object, $O_2, \ldots, O_8$ are the complex objects of O_1, $A_1, \ldots, A_3$ are the attributes of O_1, $A_{16}, \ldots, A_{18}$ 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}.

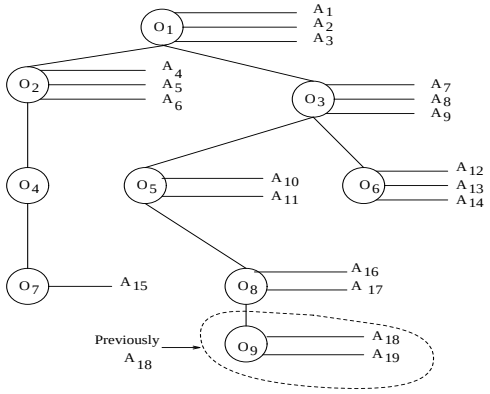

Figure 2: An example decomposition of an attribute.

An important feature of complex objects is object sharing. Sharing can occur at complex object and/or attribute levels. 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 4: An example of the distributed system environment.

ficient way of distributing complex objects within a distributed system to minimize access traffic with respect to the utilization of related objects. The ith 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 previ-

ously 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 access traffic, it becomes imperative that we strive to distinguish between the objects that are destined for local and/or remote access. Consequently, in addition to specifying the elements of a composite object, we must also specify those complex objects or attributes that are restricted to local accesses. Objects that are destined for local consumption do not participate in the partitioning process as far as generating the allocation units are concerned. In the case that an object or attribute has both types of access, then 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:

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

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:

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

When a composite object grows as a result of adding more complex objects or attributes, or decomposing an attribute, in order to determine the allocation of the new components, we must respecify the elements of the composite object in their entirety as if we are re-creating the composite object.

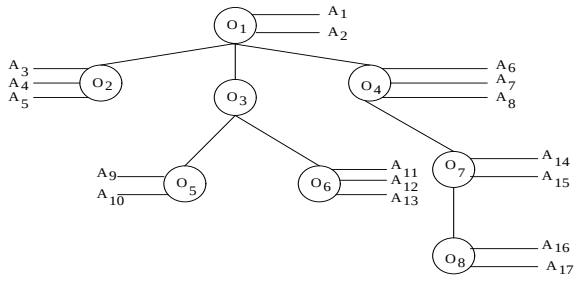

Figure 5: A composite object to be created.

This helps to maintain the current and correct morphology of a composite object.

6 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 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 a 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 greater the distance between two attributes, the smaller the probability that they may be concurrently requested. Obviously, this assumption may be application dependent; however, we expect that kind of relationship to exist more often than not since it is accounted for at the time of the hierarchical definition of the object. For example, in Figure 5, the distance between A_{10} and A_7 is 5, i.e., $[(O_5, A_7), (O_5, O_3), (O_3, O_1), (O_1, O_4), (O_4, A_7)]$ and the distance between A_{10} and A_{17} is 7, i.e., $[(O_5, A_7), (O_5, O_3), (O_3, O_1), (O_1, O_4), (O_4, O_7),$ $(O_7, O_8), (O_8, A_{17}]$. 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. When necessary, objects can be re-organized to reflect their realistic concurrent accesses.

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. However, whether at the cluster or site level,

the objective is that objects that may be concurrently utilized should be allocated together. That is, 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 affect 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 [8, 9, 10].

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 becomes obvious that we need to 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 and 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 $1 \leq i < n$ and $i < j < n$ with a weight value $b_{i,j}$ which 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 logical distance between them. Therefore, we need to develop a cost function based on the weights and build a number of sets of nodes, $G_1, G_2, \ldots, G_m$, such that the cumulative cost, $\mathcal{C}$, of the sets is minimized. That is, the sets are selected such that

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

is minimized, where

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

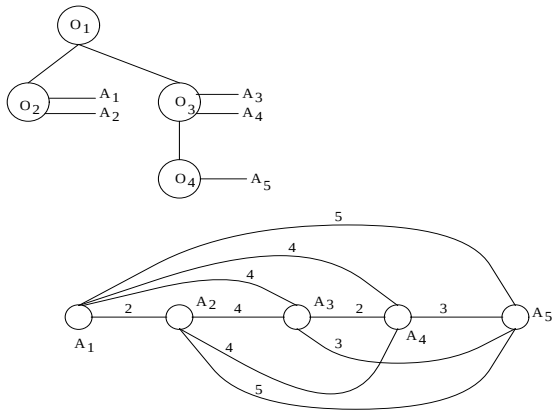

Figure 6: A composite object and its multigraph.

The exact solution of this problem is currently intractable in the sense that no polynomial-time algorithm for it is known to exist [11]. However, we use the Mincut Algorithm [12] to develop a strategy to achieve a nearly optimal 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. It primarily operates by moving one *cell* at a time between two *blocks* to partition a network. A network is a set of cells connected by a set of *nets*. A net is defined as a set consisting of at least two cells and each cell is contained in at least one net. A *Cutset* is a set of nets that have at least one cell in each block. The size of a block is the sum of the sizes of its constituent cells. The size of a cell is the number of nets to which it belongs. The mincut algorithm partitions a network into two blocks keeping the cardinality of the cutset as small as possible while maintaining a desired balance based on the size of the blocks rather than the number of cells per block. For example, Figure 7 illustrates how the mincut algorithm partitions five nodes into two blocks to minimize the cardinality of the cutset. In order to exploit the capability

Figure 7: Partitioning by the Mincut Algorithm.

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 exist 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 ith cluster; S_i is the size of A_i; $G_1, \ldots, G_m$ 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 $G_i \Rightarrow LAN\ j$ the fact that G_i is mapped to H_j or $LAN\ j$, respectively. Therefore, the following constraint also applies:

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

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. Therefore, when determining the target sites for some of the objects, we have to be cognizant of their data availability rates and the storage device configurations in the applicable sites. For those objects that require parallel access to achieve their data availability rates, we have to map them to sites that provide the capability. As with the global partitioning and disk space availability, the same condition must prevail 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_{mds}^i the number of sites in the ith network cluster that have disk arrays; n_{sds}^i the number of sites in the ith 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_{mds}^i and n_{sds}^i.
7. **for** $k = 1$ to $n_{mds}^i; n_{sds}^i$
8. **begin**
9. $G_k \Rightarrow (LAN\ k \vee H_k)$
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 $O(Nlogm)$ 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 constituent number of complex objects and attributes are randomly generated. The accessibility of an object, i.e., remote or local, is also randomly determined with more emphasis given to remote accessibility. Each composite object is independently partitioned for allocation. In order to get a better vivid 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.

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 $\frac{g}{h}$ objetcs 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 8 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 8 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.

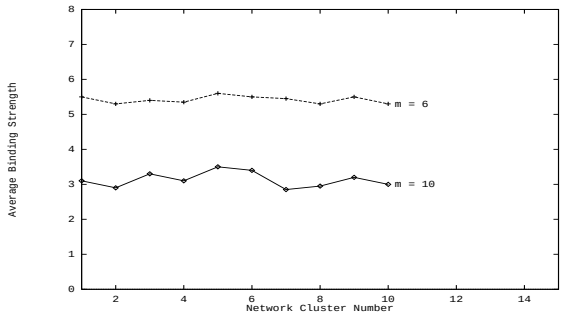

Figure 8: Objects distribution within clusters.

In the second case, the same objects in the first case were applied to the sequential and size-balancing approaches. Figure 9 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 contrast with the proposed technique. The preference of the proposed strategy is then evident and obvious.

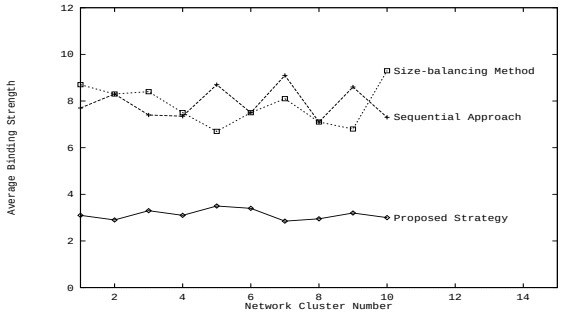

Figure 9: Comparison with other techniques.

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. In order to address the problem of balanced and optimal distribution of objects in the distributed environment, 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. We also presented some experimental results to validate the operations of the proposed partitioning strategies.

This work points to the need of partitioning complex object with cognizance of their relative utilizations. By using the concept of the binding strength derived from the hierarchical distance between pairs of objects, we capture the relative probability of corequisiteness 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," in *Object-Oriented Concepts, Databases, and Applications*, Addison-Wesley, Reading, MA, 1989, pp. 3-22.

[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] **Joel L. Wolf**, *et al*, "Multisystem Coupling by a Combination of Data Sharing and Data Partitioning," *IEEE Trans. on Software Engineering*, Vol. 15, No. 7, pp. 854-860, 7/89.

[5] **D. Cornell, P. S. Yu**, "Site assignment for relations and join operations in the distributed transaction processing environment," in *Proc. 4th Int. Conf. Data Engineering*, Los Angeles, CA., 2/88, pp. 100-108.

[6] **D. H. Fishman,** *et al*, "Overview of the Iris DBMS," in *Object-Oriented Concepts, Databases, and Application*, Addison-Wesley, Reading, MA, 1989, pp. 219-250.

[7] **Robert Bretl,** *et al*, "The GemStone Data Management System," in *Object-Oriented Concepts, Databases, and Applications*, Addison-Wesley, Reading, MA, 1989, pp. 282-308.

[8] **M. Y. Kim**, "Synchronized Disk Interleaving," *IEEE Trans. on Computers*, Vol. C-35, No. 11, pp. 978-988, Nov. 1986.

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

[10] **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," in *Proc. IEEE 17th Annual International Computer Software and Applications Conference (COMPSAC)*, Phoenix, Arizona, 11/93, pp. 216-223.

[11] **Michael R. Gray, David S. Johnson**, *Computers and Intractability*, W. H. Freeman and Company, 1979.

[12] **B. W. Kernighan, S. Lin**, "An efficient heuristic procedure for partitioning graphs," *Bell System Technology Journal*, Vol. 49, pp. 291-307, 2/70.