An Object-Based Approach for Effective XML Data Storage

Athena Vakali and Evimaria Terzi

Department of Informatics
Aristotle University of Thessaloniki, Greece
E-mail: {*avakali,evimaria*}*@csd.auth.gr*

Abstract. XML data storage is a critical issue due to the so-called *I/O bottleneck* problem emerged in nowdays computer systems. This paper presents an object-based XML data representation model towards effective XML data placement The proposed representation of XML documents is analysed in a two-level scheme : the external level is based on the structure of a browsing graph whereas the internal level is supported by a tree-like structure. The main contribution of the paper is that it exploits the object data model in order to consider XML data dependencies, access frequencies and constraints. A simulation model has been developed in order to evaluate different XML data placement strategies and the impact of the proposed representation model in the overall storage process. XML data placement is applied on a tertiary storage subsystem by either constructive or iterative placement techniques. Three popular policies : the Organ-pipe, the Camel and the Simulated Annealing algorithms, have been considered and experiments have been carried out on synthetic workloads of XML data sets. The need of applying an XML data storage policy is apparent as indicated by the resulted improvements in seek and service times. The Simulated Annealing approach has been proven to outperform the other XML data placement strategies.

Index terms: *XML data storage, XML data representation model, tertiary storage subsystems, XML data placement algorithms.*

1 Introduction

XML (e*X*tensible *M*arkup *L*anguage) has recently emerged as a new standard for data representation and exchange on the Internet [9, 25]. XML has been widely adopted for semistructured data exchange due to its simplicity and its facilitating on Internet access. Efficient physical layout of XML data is required and has become crucial issue due to large stores of XML data circulated across the Internet.

The graph and tree-like structures have been proposed in earlier research papers in relation to the representation of semi-structured data. Different algorithms have been proposed in the literature for the analysis of graph-based schemes under an approach to classify semi-structured data [2]. The XML data

model is usually implemented by considering the XML document as a linearization of a tree structure. [4, 17]. The XML data tree like structure has a number of different type nodes and a number of connections between arbitrary nodes, with links among children as well. Furthermore, the DOM (*D*ocument *O*bject *M*odel) defined for XML considers everything as nodes and objects [26].

Efficient physical layout of XML data is a major research topic as identified in [25] and (according to authors knowledge) only preliminary research work has been published on this topic. The most common approach is to store XML documents in flat files whereas efficient storage of XML data has been introduced in [17]. In this paper an efficient, native repository (*NATIX*) is introduced for storing, retrieving and managing tree-structured large objects, preferably XML documents.

The present paper proposes an object-based XML data representation model such that the XML data placement is guided effectively on a tertiary storage subsystem. Tertiary storage medium is considered for large store of XML data due to their high capacities and low costs. The role and importance of Tertiary Storage Systems has been enforced since recent technological advances include large-scale storage servers such as near-line, robot-based, tertiary storage libraries. It is interesting to note that tertiary systems have different and diverse performance factors not applicable to all technologies as indicated in [19, 20]. Several policies have been proposed in secondary storage level in order to place the most often requested data around the central of the disk's surface ("organ-pipe" placement) or around the middle of the two intervals derived by the central disk cylinder ("camel" placement). Data placement has been also studied for Tertiary Storage SubSystems. For example, in [8] different data placement policies on Tape Libraries consisting of different technology tapes implemented, while in [22] optimal arrangement of cartridges and file-partitioning schemes are examined in Carousel type Mass Storage Systems. Furthermore, information placement schemes are considered in three different models corresponding to three mid-range magnetic tape systems [24]. Iterative improvement placement algorithms, and Simulated Annealing in particular, have also been implemented in [6]. A detailed description of the simulated annealing algorithm is given in [15] while its implementation on database systems has been discussed in [14].

The main contribution of the paper is that it considers an object-based model to implement the data structures used for XML documents interaction, towards an effective XML data storage policy. The storage policy is applied on a tertiary storage model according to the following considerations :

- the *navigation path* among various XML documents is considered for XML data representation and their storage. The relationships among various XML documents are considered to guide the storage policy and XML data are no longer considered to be independently accessed.
- the *physical entities* are defined for the XML documents and the XML data placement is studied with respect to these entities. Each XML document is analysed into objects to be physically stored. The frequency of access of an

XML document is the most crucial metric for the positioning of its physically stored objects.

The remainder of the paper is organized as follows. The next section introduces the proposed XML data representation model while the XML objects storage model is described in Section 3. The considered tertiary storage models are analysed in Section 4 and the XML data placement algorithms are described in detail in Section 5. Section 6 has the simulation, experimentation details and results whereas conclusions and further research topics are identified in Section 7.

2 The XML Data Representation Model

```
<LAB>
        <MEMBER>
                <NAME>John Smith< \NAME>
                <AGE>25<\AGE>
                <TITLE>Phd Student<  \TITLE>
        <   \MEMBER>
        <PROJECT>
           <TITLE>Tertiary Storage< \TITLE>
           <WORKLOAD src="…wrkld.file"> <   \WORKLOAD>
           <REPORTS src="…rpt.file">  <   \REPORTS
        < \PROJECT>
< /LAB>
```

Fig. 1. An XML Document.

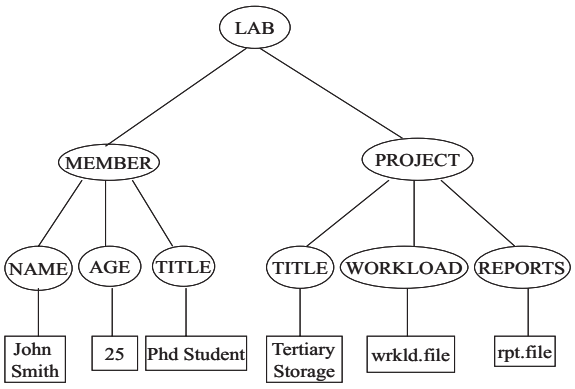

Fig. 2. Tree structure of an XML Document.

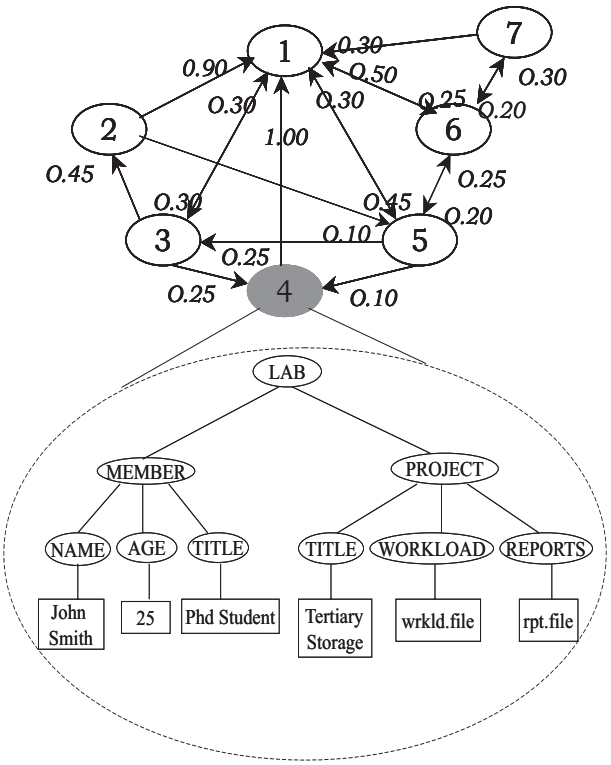

Fig. 3. A browsing graph for XML documents and their objects

As mentioned in the previous section, the data model for XML documents is usually considered as a linearization of a tree structure. At every node in the tree there are several character strings. The tree structure and the character strings together form the information content of an XML document. The tree graph overlaid and the different types of nodes are defined in [4]. XML documents are circulated across the Internet and there are certain access patterns for interaction and interconnection of XML documents. These patterns are the result of users/clients navigation among several XML documents. The graph model is quite appropriate for XML application representation, since we need a conceptual model to represent the overall access pattern among different XML documents. Since a user "moves" from an XML document to another, the navigation path can be represented by arcs in a directed graph. Each distinct XML document will be considered as a node of a directed graph and each node has an internal tree-like representation analysis. Thus, two representation "levels" are introduced :

1. *The External level :* user's interaction can be represented by using the browsing graph as a "map" of XML documents visited by the user. The idea of a browsing graph was used in [6] to capture multimedia documents naviga-

tion. Here, each node in the browsing graph corresponds to a distinct XML document itself, while the directed arcs represent the relationships among the various visited XML documents.

Definition 1 : *The Browsing Graph* is a directed graph $G = (N, A)$ where $N = \{1, 2, \cdots, k\}$ is a set of k nodes corresponding to k XML documents and A is a set of directed edges connecting specific pairs of N. Additionally, every edge in A is weighted by an access or transition probability. These probabilities define the so-called transition matrix:

Definition 2 : The *transition matrix* P associated with the graph G, is a $(k \times k)$ matrix of access or transition probabilities , where by p_{ij} $(i, j \epsilon \{1, 2, \cdots, k\})$ we denote the probability of accessing node j from node i at a single step. The probabilities can be estimated by the frequency of accesses to nodes, computed by actual (traced) data logs of accesses on the XML documents.

2. *The Internal level :* At this level each node of the global view structure (i.e. the browsing graph) is a tree-like structure [4, 17]. Figure 1 presents an example of an XML fragment of a particular XML document. The tree-like representation (an example is given in Figure 2) corresponds to the logical view of the XML document and it is used to manage and guide the storage of XML documents. This model allows us to emerge all the necessary information needed to specify an XML document and all of its components.

   Definition 3 : *XML object* is a physical storage unit identified by an entity or group of entities which correspond to the nodes of the original XML document tree structure.

   Thus, the XML document can be viewed as a set of XML objects in order to define an appropriate XML data storage policy.

3 The XML objects storage model

3.1 The External Level

The browsing graph representation model is actually a homogeneous Markov chain since the transition probabilities are time-independent. The notion of the "system" in the Markov chains terminology stands for the user's actions while the "states" of the system are the XML documents or the nodes of the browsing graph. It is also clear that the transition matrix P is a stochastic matrix, i.e. its elements are either zero or positive and its row sums are all ones, that is $\sum_{j=1}^{k} p_{ij} = 1$, (for all i). The assumptions posed in [6] and a similar ergodic Markov chain is also considered here.

Definition 4 : The row vector $f = (f_1, \cdots, f_k)$, where $\sum_{i=1}^{k} f_i = 1$ and

$$fP = f \quad or \quad f_j = \sum_{i=1}^{k} f_i P_{ij}, \quad \text{for} \quad j = 1, 2, \cdots, k$$

is called *vector of access frequencies* of the XML objects $1, \cdots, k$ and its elements provide metrics to identify the popularity of each XML object involved in the browsing graph.

The evaluation of the frequencies vector f will be based on a well-known result summarized in the following theorem:

Theorem 1 : If P is the transition matrix of a homogeneous ergodic Markov chain, then there is a unique vector $f = (f_1, \cdots, f_k)$, such that

$$\lim_{k \to \infty} P^k = \begin{pmatrix} f \\ f \\ \vdots \\ f \end{pmatrix} \tag{1}$$

Proof : A thorough study and classification of finite Markov chains and the proof of this theorem is given in [10, 16]. The theorem gives us a way of approximately evaluating the access frequencies of the nodes, by simply calculating powers of the transition matrix.

Example 1 : In Figure 3 we depict the browsing graph for the access patterns among seven distinct XML documents. In a real environment the access probabilities are estimated by keeping track of the users interaction over a certain time period. Thus, the access probabilities will be stored in the following transition matrix :

$$P = \begin{bmatrix} 0 & 0 & 0.3 & 0 & 0.45 & 0.25 & 0 \\ 0.9 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0.3 & 0.45 & 0 & 0.25 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0.25 & 0.1 & 0 & 0.35 & 0 \\ 0.5 & 0 & 0 & 0 & 0.2 & 0 & 0.3 \\ 0.3 & 0 & 0.5 & 0 & 0 & 0.2 & 0 \end{bmatrix}$$

When raising this transition matrix to a power we result to the vector f of access frequencies. For example f is any row in the 32-th power of P and thus, the frequency of accesses will be the vector
$f = (0.3205, 0.0741, 0.1647, 0.0594, 0.1823, 0.1531, 0.0459)$.

3.2 The Internal Level

As mentioned earlier, each node of the browsing graph is represented by a tree-like structure. The nodes of this tree are classified into two main categories :

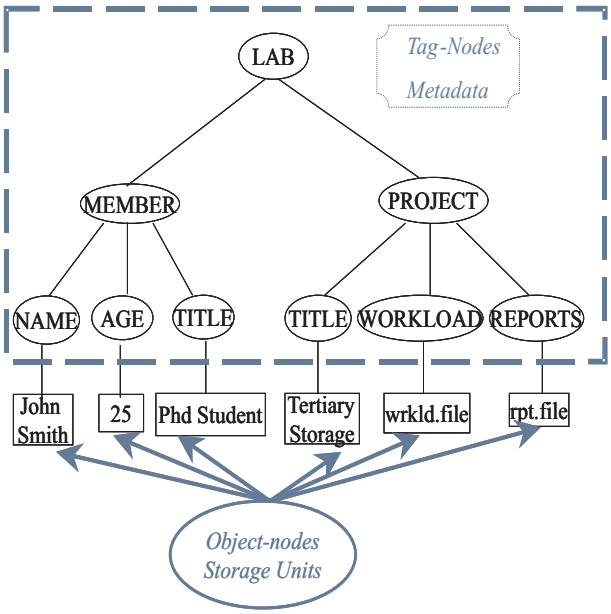

Fig. 4. XML document storage units.

- *Tag-nodes :* All the internal tree nodes are tag-nodes. These nodes refer to the relative tags of the XML document, they mainly contain metadata, and therefore they guide the request servicing procedure.
- *Object-nodes :* These nodes are equivalent to the leaves of the tree. They represent the XML objects which are the primary components of the XML document. Their storage should be carefully studied in order to improve the performance of a system presenting XML documents.

Figure 4 depicts the considered tag-nodes and object nodes for the XML document discussed in Example 1. In order to appropriately guide the XML document's storage we propose a separate and independent storage of Tag-nodes and Object-nodes. More specifically the following storage approach has been followed:

1. The Tag-nodes of each XML document are stored either in the secondary storage or in the cache memory so as to be easily and quickly accessed and therefore guide the system to the respective XML Objects effectively.
2. The Object nodes are stored in tertiary storage devices, named magnetic tapes, and they are elevated whenever a request for an XML document that contains them arrives. Notice, that there might be XML objects that are included in more than one XML documents. However, there is only one replica of them. In other words, there is a "pool" of XML objects, and each one of them is considered to be a separate storage unit that can be stored and retrieved when necessary.

4 Tertiary Storage and XML Data Placement

In our approach, tertiary storage subsystems are considered for XML data placement. XML documents frequently contain multimedia data which are characterized by their large size, and therefore require high capacity storage systems. Moreover, XML documents are becoming increasingly popular, which results in the formulation of large XML repositories which need to be stored on efficient tape-based systems.

Tertiary storage includes magnetic tapes, optical disk devices and some more recent technologies like optical tapes and holographic storage. Recent advances in tertiary storage technology have made it of increasing interest to computer system designers. A variety of inexpensive tape drives have become available and the low cost of magnetic tape makes it economical to built massive tertiary storage systems. The most common and widely adopted tertiary storage subsystems include *Magnetic Tapes* and *Tape Libraries* which are considered in the present paper.

- **Magnetic Tapes**
  Magnetic tapes have been the most dominant type of tertiary storage media. Tapes can be classified according to the format in which data is recorded and they are usually categorized as follows:
  1. *Tapes that Rewind to the PBOT:* These are tapes that always rewind to their beginning before being ejected.
  2. *Tapes that Rewind to the nearest Zone:* These tapes enable tape rewinding to the nearest zone as opposed to those require to be rewound to their physical beginning before being ejected.

  In our approach tapes are thought to be linear areas. Each linear area is divided into a certain number of fixed-size *segments*, which are the smallest accessible parts of the tape. *Sections* consist of a number of consequent segments, while *tracks* consist of a number of consequent sections. The number of segments that will be allocated to a data object mainly depends on the object's and the segment's size.

  Definition 5 : The number of segments s reserved for a single stored XML object is given by :

$$s = \left\lceil \frac{size\ of\ object}{size\ of\ a\ segment} \right\rceil$$

  If the object's size is not divided by the segment's size, then the last segment allocated to an object is not full, and therefore some space remains empty. The amount of this space depends on both the size of XML objects and the capacity of a single segment. In other words, the smaller the segment size is the less the unused space will be. On the other hand for a constant segment size, it is obvious that the percentage of the empty space depends on the size of the XML objects and the smaller they are the higher this percentage will be. Since XML documents may include any type of data (text, image, video, audio etc) with variable sizes, the ideal segment size depends on the

specific application. However, it could be supposed that the amount of this space is relatively small when compared to the capacity of the tape and the total size of objects that can be stored on it.

– **Tape Libraries**
Several tertiary storage manufacturers have built automated library systems in order to provide higher bandwidth and capacity. These libraries hold large number of cartridges that can be loaded by robot arms into a collection of magnetic tape drives. Robot access times vary from a few seconds to about 20 seconds for most libraries.
Tertiary Storage Libraries come in many different sizes and configurations. Despite their significant differences, they all contain : *Drives, Robot Arms* and *Tapes or Disks*. Consequently, they can all be modeled very similarly. The library is thought to have one or more robot arms, which are assumed to be identical. Each arm is assumed to be capable of moving between any tape stored in the library. There are three operations the robot arms make: *Pick and Place* where a pick operation refers to the robot picking up a tape from its storage space in the library and taking it into the drive and *Move* which is an operation refering to the robot moving between different locations on the shelf. Although there is variation in the times required for these operations, they are generally modeled as constant times. This is acceptable because the variation is much smaller than the total time required to exchange tapes or disks on drives. Consequently:

$$Robot\_Arm\_Service\_Time = Pick\_Time + Move\_Time + Put\_Time$$

The drives, which are also assumed to be identical, perform the following operations: seek, rewind, read, write, load and eject. The seek and rewind operations for tapes are modeled as constant startup times followed by a constant transfer rate. Therefore, the total tape access time is evaluated by :

$$Total_{Service-Time} = Robot_{Arm-Service-Time} + Drive_{service-time}$$

whereas the $Drive_{service-time}$ if further estimated by [7, 13] :

$$Drive_{service-time} = Rewind_T + Eject_T + Load_T + Seek_T + Transfer_T$$

5 XML Data Placement Algorithms

5.1 XML Data Constraints

XML documents may include different types of data varying from text to multimedia (e.g. video,audio,images) and therefore their demands may exceed the resource demands of conventional data types and require massive amounts of space and bandwidth for their storage and their display. Therefore, physical entities should be placed and retrieved in an effective way, so as to lead up to a

satisfactory system's performance. XML documents include a number of physical entities which define a "pool" of physical objects. Each one of them should be placed appropriately in order to guarantee high quality of service. Therefore our problem is :

"How to place the XML objects in a certain storage medium in order to improve access to the requested XML documents"

The following questions arise in relation to XML objects classification and characterization :

1. *What is an XML object Access of Frequency ?* The relative frequency of access of each XML document (a node in our browsing graph), must be considered in the estimation of the relative frequency of access of each physical entity in the pool. Since each physical entity may appear a number of times within a node and also participate in a number of distinct nodes there is a chance that an XML object is included in both "popular" and "non-popular" nodes. Therefore, a metric should be defined in order to compare the XML objects frequency of access.
2. *How is the XML document continuous display supported ?* As long as XML documents may contain multimedia data there might be timing conststraints imposed on physical entities should also be taken into consideration. Therefore, we should specify which physical entities are needed simultaneously in the existing nodes, and how many times this co-existence occurs .

In order to answer the above questions we need to define certain criteria that will impose classification of physical entities and perform a prioritization and sorting among them in order to appropriately guide the data placement policy. The relative frequency of access of each XML document (a node in our browsing graph), must be considered in the relative frequency of access evaluation for each XML object (a leaf in our tree structure).

Definition 6 : The *popularity of an XML object* x is defined by

$$pop[x] = \sum_{i=1}^{k} f_i \times (number\ of\ object\ x\ references\ in\ node\ i)$$

where f_i is the frequency of access (Definition 4) of the XML object corresponding to node i and estimated by the formula given in Theorem 1. It is obvious that the higher the frequency of access of an XML physical object, the more popular this object is.

The *pop* value is higher when an object is part of a "popular" XML document (node) and furthermore when it appears several times within this node. Therefore all objects included in popular nodes will have a high popularity value. This parameter is indicative and convenient in guiding the storage policy since both "popularity" of physical objects and their timing constrains are considered.

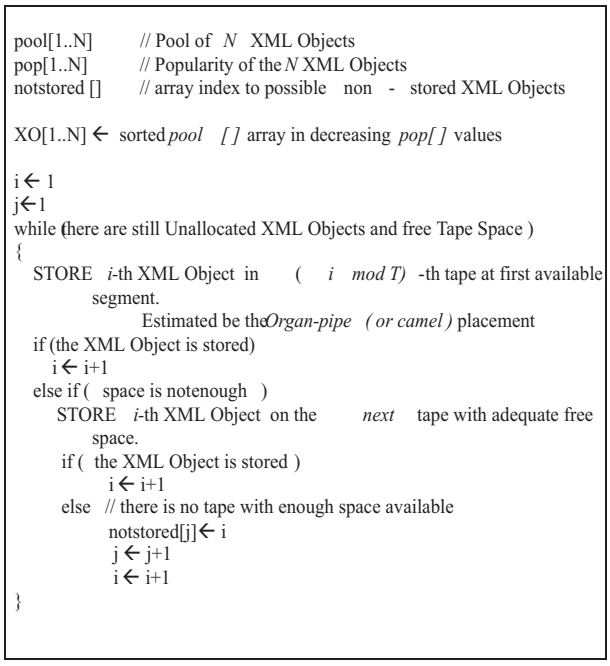

Fig. 5. Constructive Placement Algorithms

5.2 The Placement Algorithms

Our data placement problem is to store N physical entities onto T tapes. We consider the tape topologies described in Section 3 and we perform placement on tapes that rewind to the nearest Zone. In the proposed data placement algorithms the popularity of physical entities to be stored is evaluated by the XML documents frequency of access as identified by the browsing graph edges and the inter-connection between the physical entities within the nodes. The data placement algorithms implemented can be divided into two main categories : *Constructive placement* and *Iterative improvement*. For constructive placement *organ-pipe* and *camel* placements are implemented. Figure 5 presents the generic structure of these algorithms under a tape library storage system with T tapes. The organ-pipe and camel placement policies implemented within a tape consisting of Z zones are summarized as follows :

Organ-Pipe Placement

```
(A) Place the most popular object
    on the middle zone (Z/2) of the tape
(B) Allocate the next two popular objects
    on either side of the middle zone
(C) Go to (B) until all objects are placed.
```

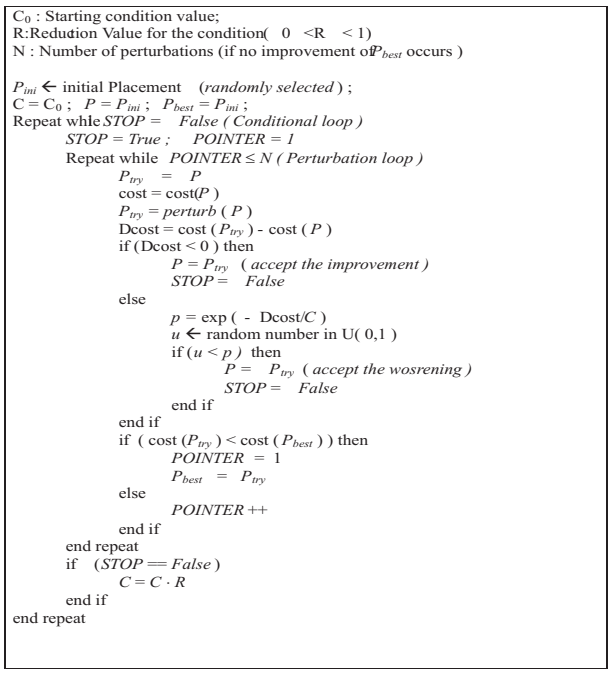

Fig. 6. General Concept of Simulated Annealing algorithm

Camel Placement

```
(A) Divide the tape into two consecutive tapes
    consisting of (Z/2) zones
(B) Implement Organ-Pipe placement alternatively
    on the two consecutive tapes
(C) Go to (B) until all objects are placed.
```

Simulated Annealing The iterative improvement on the other hand, starts with an initial placement determined by a constructive placement procedure and is repeatedly modified in search for cost reduction. If a modification results in a reduction, it is accepted. Otherwise it is rejected. Simulated Annealing is a popular algorithm for combinatorial optimization and a description of the algorithm is given in Figure 6. Next we comment on the modules such as the *initial placement strategy, the perturbation procedure* and the *cost function* :

- The *initial placement* strategy is presented in Figure 7. The initial placement of the physical entities within the tapes can follow either one of the constructive placement methods described above, or a random policy.
- The *rearrangement (perturbation)* of the physical entities within the tapes can be applied following two strategies.

```
pool[1..N]        // Pool of  N  XML Objects
pop[1..N]         // Popularity of the N XML Objects
notstored []      //  array index to possible    non-  stored XML Objects

XO[1..O] ← sorted  pool [ ] array in decreasing pop[ ] values

                    Initial  Placement
i ← 1
j←1
while (there are still Unallocated XML Objects and free Tape Space )
{
        STORE i-th clip  in   ( i  mod T) -th tape at the available segment.
        if (the XML Object is stored)
           i ← i+1
        else if ( space is not  enough   )
                STORE    i- th  XML Object on the  next tape with adequate free
                         space.
           if (the XML Object is stored)
                i ← i+1
           else  // there is no tape with enough space available
                notstored[j]← i
                j ← j+1
                i ← i+1
}
```

Fig. 7. Simulated Annealing Initial Placement

1. Interchange : Select two physical entities of the current configuration randomly, and interchange their positions.
2. Rotation : Make a left (or right) circular shift of current configuration.

– As a measure for the cost of each perturbation expected service time is chosen. If the XML application is consisted of N physical entities and i, j refer to all possible combinations of the entity that lies under the reading head (XML object i) and the one that is requested (entity j) , the Expected Service Time can be evaluated using the following formula :

$$Expected\ Service\ Time\ = \sum_{i=1}^{N} pop(i)pop(j)(s_j s_{rate} + t_j t_{rate})$$

where s_j and t_j are the number of bytes to search and transfer respectively, while s_{rate} and t_{rate} are the search and transfer rates respectively.

6 Simulation - Experiments

Our simulation model consists of the three main modules (Figure 8) :

– *The Data Representation Module :* the browsing graph is constructed based on users access patterns. Representation is performed in both Internal and

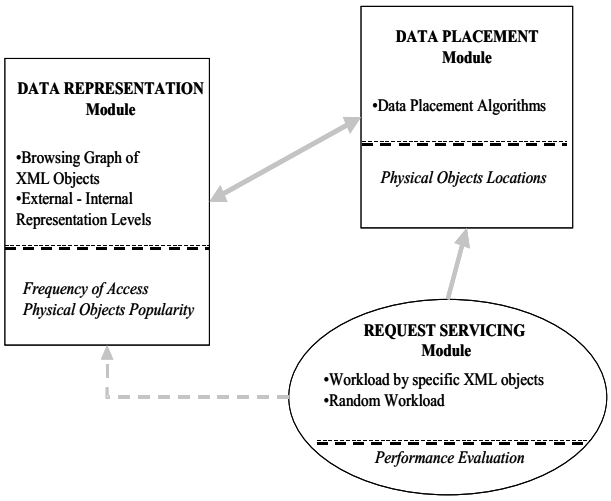

Fig. 8. The Modules of the Simulation model.

External levels as described in Section 2. The access frequencies are evaluated by using the formula given in Theorem 1 and the XML Objects popularity estimation is based on Definition 6.

- *The Data Placement Module :* at each simulation run, one of the data placement algorithms (described in Section 5) is performed according to the selected tape topology. The location of each XML object is identified such that request servicing can be performed.
- *The Request Servicing Module :*
  the requests define the specific workload which could be considered either independent or based on the access frequencies patterns as identified by the browsing graph. The request servicing process is depicted in Figure 9, and performance metrics are evaluated in order to compare the proposed placement policies.
  An XML object is elevated in the cache memory by the following steps:
  1. Specification of the magnetic tape on which the object is stored where the following cases might hold:
     - The tape is off-line and there is a drive available. In this case the tape is loaded on the drive.
     - The tape is off-line and there is no drive available. In this case the request waits and the specified tape is loaded on the first drive that becomes idle.
     - The tape is already loaded on a tape drive. In this case the physical object is read as soon as the service of the previous request referring to that tape is finished.
  2. The next XML object of the node is specified and brought in cache in the way already described above.

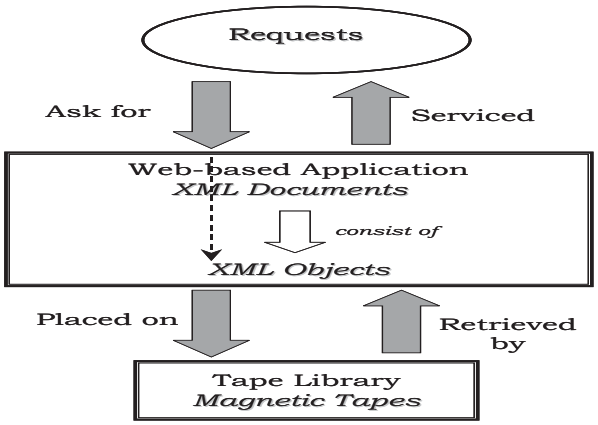

Fig. 9. The request servicing process.

3. The total time spent to elevate all the objects of the XML document is evaluated.

6.1 The Results

Parameter	Value
$NDRIVES$	1
$NROBOTS$	1
NUM_TAPES	2-10
NUM_REQ	100
$PICK\_TIME$	10secs
PUT_TIME	10secs
$MOVE\_TIME$	2secs
$EJECT\_TIME$	8secs
$LOAD\_TIME$	10secs
$SEEK\_OVHD(REW\_OVHD)$	0.1(0.1)sec
$SEEK\_RATE(REW\_RATE)$	192512(197632) KB/s
$XFER\_RATE$	3072 KB/s
$TAPE\_SIZE$	3039232-379904 Segments
$SEGMENT\_SIZE$	2-32KB

Table 1. parameters used for the tape library model

We have run experimentation workloads for Zoned tapes as long as they are widely used and they tend to replace PBOT tapes. The parameters of the model used for these experiments are shown in Table 1. Numerous sets of requests

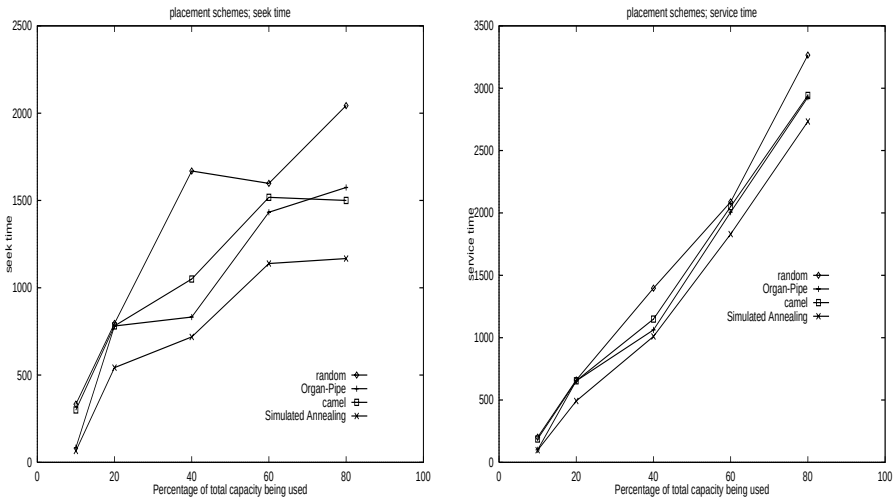

Fig. 10. Percentage of total Capacity being used.

were generated in order to evaluate the proposed placement schemes. Simulation results include both seek and service times and in all cases a browsing graph with a constant number of nodes is used. More specifically, random, organ pipe, camel and simulated annealing placement strategies have been considered and results indicate that simulated annealing considerably improves system's performance.

Figure 10 depicts seek and service times for a storage system with a fixed number of tapes, with respect to the percentage of the available storage space being occupied by XML objects. The percentage varies from 10% to 80% of the total storage system capacity. Simulated annealing considerably improves system's performance while random placement causes high seek and service times. It is interesting to note that both seek and service times increase with the number of physical entities since the increase in the number of physical objects results in more complicated XML objects and therefore more demanding requests. Note that camel placement proves to be as good as organ-pipe when only 20% of the total storage capacity is used and considerably more effective when the storage space usage exceeds 80%. Simulated annealing results in seek times 15-30% compared to those obtained when organ pipe placement was implemented. Furthermore, simulated annealing proved to be better than camel placement since the seek (service) times obtained show an improvement rate of 15% to 80% (12% to 50%). The less XML objects placed on the tapes, the higher the improvement rate is, since there are more potential rearrangements.

Furthermore, Figure 11 depicts the system's performance for a varying segment's size (2-32KB). It is obvious, that when the segment's size is large, there is a lot of unused storage space, particularly in the case of non-multimedia data which usually are small files such as text files. However, this space may prove to be useful when additional information has to be inserted for a specific XML

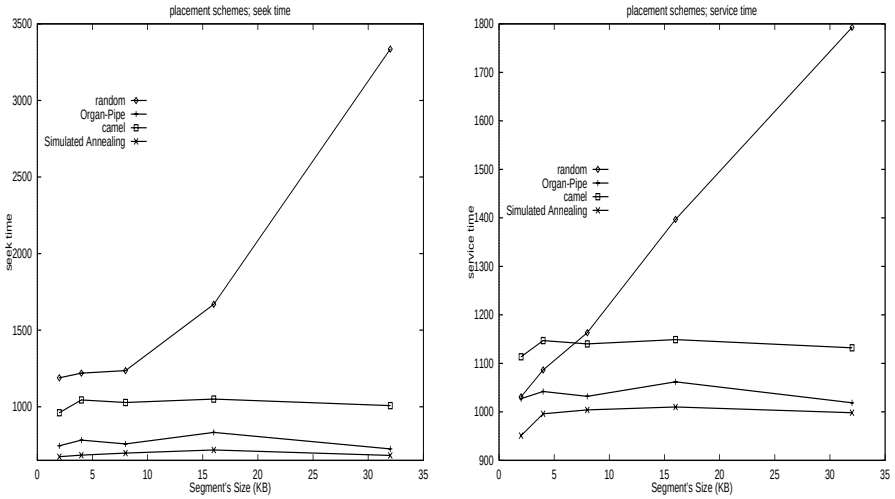

Fig. 11. Placement Policies; Segment's size.

object. On the other hand, small segment's size proves to be more advantageous for read-only requests, since XML documents consist of variable-sized physical entities and therefore no extra bytes are elevated from tertiary storage every time a request is serviced. Figure 11 shows that the random placement policy results in large seek and service times that increase with the segment's size. On the other hand, the performance of the other three algorithms proves not to depend on the segment's size , which can be easily explained, as long as our artificial workload consisted of a lot of large-sized multimedia physical entities. In the latter case, simulated annealing outperforms the other placement strategies. As far as the other two algorithms are concerned the graphs illustrate that organ pipe leads to better system's performance than camel placement. Notice, that simulated annealing results in 10% and 7% improvement of seek and service times respectively when compared to organ pipe placement. The respective values of improvement rates are 31%, 15% when the performance of simulated annealing is compared to that of camel placement.

Finally, graphs of Figure 12 depict the expected seek and service times under random, organ-pipe, camel and simulated annealing, with respect to a varying number of tapes (2 to 10). Simulated annealing again, is proven to be the most beneficial placement policy. More specifically, simulated annealing results in a 10% better seek and service times when compared with organ pipe placement. We notice that when the number of tapes is small simulated annealing improves system's performance (in relation to the seek times) by 30%. The performance diversion between the two placement policies' performance narrows when they are implemented on tape libraries with 6 (22%) and 8 tapes (15%), but even in these cases there is a significant overall improvement. It is obvious that seek times obtained by simulated annealing are much smaller than those obtained

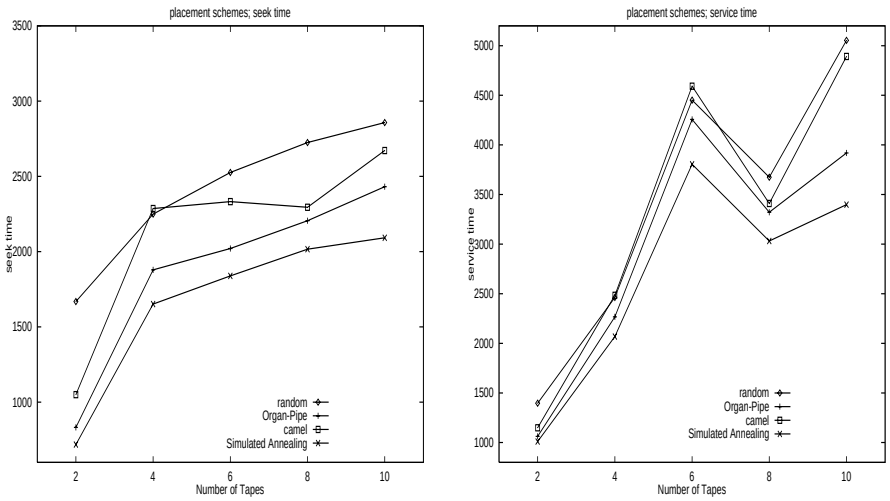

Fig. 12. Placement Policies; Number of Tapes.

by random placement policy. The relative improvement of the seek time varies form 67% (2 tapes) to 27% (4,6,8,10 tapes). There is also an improvement of the service times obtained by simulated annealing placement when compared to those obtained by camel. The improvement rate is 12% for systems with 2 or 8 tapes, 17% for systems consisted of 4,6 tapes and reaches even 30% for ten-tape systems.

Placement Algorithm	Percentage Coverage		Segment Size		Number of Tapes	
	40%	80%	16KB	32 KB	6	8
Organ-pipe	24%	11%	24%	43%	4%	10%
Camel	18%	10%	18 %	37%	-3%	7%
Simulated Annealing	28%	16%	28%	45%	15%	18%

Table 2. Improvement rates between proposed policies and random placement.

Table 2 summarizes on some indicative improvement percentages as resulted for service times under the different storage policies, compared to the random placement. The effectiveness of applying a storage policy is proven since there is always an improvement in both Organ-pipe and Simulated Annealing policies. The Camel approach is the only one that might result in worse seek or service times. Simulated Annealing has been proven to be the best choice in all cases, and it has improved seek and service time figures considerably.

7 Conclusions - Future Work

In this paper, we have considered an object-based model for XML documents representation in relation to the quest of an efficient XML data storage policy. Our representation model included two levels with an associated appropriate structure, the external level had a browsing graph for the navigation among XML documents and the internal level had a tree-like structure for the XML document itself. The notion of the XML object was defined in order to guide the storage policy according to the frequencies of accesses on the considered physical entities. Organ-pipe, Camel and Simulated Annealing placement policies were considered on a tertiary storage model of tape libraries. Simulated annealing has been proven to be the most beneficial placement policy with significant improvement rates in both seek and service times.

Further research should include the implementation of information placement algorithms on other types of Tertiary and Secondary storage systems including optical and magnetic disks. Moreover,we could extend our model in order to exploit all levels of the storage hierarchy in order to improve both response and service times. For example we can propose a data placement approach based on documents' access frequencies and dependencies, in order to "split" the browsing graph among secondary and tertiary storage levels. Thus, in our future hierarchical storage approach, secondary level could serve as a cache for the tertiary level. All objects will be stored in Tertiary Storage initially according to the placement policies mentioned here. Finally, scheduling algorithms based on the proposed representattion models, could also be investigated in order to satisfy certain quality criteria in XML applications.

References

1. S. Abiteboul, P. Buneman, D. Suciu, J. Gray : *Data on the Web : From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers, 1999.
2. E. Bertino, G. Guerrini, I. Merlo and M. Mesiti : An approach to Classify Semi-Structured Objects, *Proceedings of the 13th European Conference on Object-Oriented Programming*, 1999.
3. K. Bohm, K. Aberer, E. Neuhold and X. Yang : Structured document storage and refined declarative and navigational access mechanisms in HyperStorM, *The VLDB Journal*, No. 6, pp. 296-311, 1997.
4. B. Bos : The XML Data Model, *http://www.w3.org/XML/Datamodel.html*, 1999.
5. B. Bos : XML Representation of a Relational Database, *http://www.w3.org/XML/RDB.html*, 1999.
6. Yie-Tarng Chen : Physical storage model for interactive multimedia information systems, PhD Thesis, Department of Electrical Engineering, Purdue University, 1993.
7. A.L. Chervenak: Tertiary Storage-An Elevation of New Applications, PhD Dissertation, University of California at Berkley, 1994.
8. Stavros Christodoulakis, Peter Triantafillou and Fenia Zioga : Principles of Optimally Placing Data in Tertiary Storage Libraries, *23rd International Conference of Very Large Databases (VLDB)*, pp.236-245, Athens, Greece 1997.

9. D. Connolly and J. Bosak : Extensible Markup Language (XML), *http://www.w3.org/XML/*, 1999.
10. D.R. Cox and H.D. Miller : *The Theory of stochastic processes*, Chapman and Hall, 1977.
11. G.A.Gibson, J.S. Vitter, J.Wilkes : Strategic Directions in Storage I/O Issues in Large-Scale Computing, *ACM Computing Surveys*, Vol.28, No.4, pp779-793, 1996.
12. E. Rusty Harold : *XML Bible*, IDG Books Worldwide, 1999
13. B.K.Hillyer and A. Silberschatz: On the Modeling and Performance Characteristics of a Serpentine Tape, *Proceedings ACM SIGMOD'96 Conference*, pp.170-179, 1996.
14. Kien A. Hua, S.D. Lang and Wen K. Lee : A decomposition-based simulated annealing technique for data clustering. *Proceedings of the 13th ACM symposium on Principles of database systems*, Minneapolis, USA, May 1994.
15. Mark Fleischer : Simulated Annealing : Past, Present and Future. *Proceedings of the 1995 Winter Simulation Conference.*
16. D.L. Isaacson and R.W. Madsen : *Markov Chains Theory and Applications*, John Wiley and Sons, 1976.
17. C.C. Kanne and G. Moerkotte : Efficient Storage of XML data, *Technical Report*, University of Manheim, Germany, Dec. 1999.
18. R. Khare and A. Rifkin : XML : A Door to Automated Web Applications, *IEEE Internet Computing*, pp.78-87, July - August 1997.
19. Sunil Prabhakar, Divyakant Agrawal, Amr El Abbadi, Ambuj Singh: Tertiary Storage: Current Status and Future Trends, Computer Science Department, University of California, Santa Barbara, August 1996.
20. Sunil Prabhakar, Divyakant Agrawal, Amr El Abbadi: Impact of Media Exchanges in Robotic Libraries,Computer Science Department, University of California, Santa Barbara, June 1997.
21. H. Maruyama, K. Tamura, N. Uramoto, *XML and Java: Developing Web Applications*, Addison-Wesley Publ. Co, 1999
22. S.Sesardi, D. Rotem and A. Segev : Optimal Arrangements of Cartridges in Carousel Type Mass Storage Systems, *The Computer Journal*, Vol.37, No.10, pp.873-887, 1994.
23. UNC Sunsite: XML sample documents, *http://metalab.unc.edu/pub/sun-info/standards/xml/eg*, 1999.
24. A.Vakali and Y.Manolopoulos: Information Placement Policies in Tertiary Storage Systems, Storage Models for Multimedia Object, *Proceedings of the Hellenic Conference of New Information technologies*, pp.205-214, Oct 1998.
25. J .Widom: Data Management for XML - Research Directions, *IEEE Data Engineering Bulletin*, Special Issue on XML, Vol. 22, No. 3, Sept. 1999.
26. W3C DOM : Document Object Model lvel 1 Specification, *http://www.w3.org/TR/REC-DOM-Level-1*, 2000.