Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 7 (2): 165-174, 2007
ISSN 1812-5654

A Threshold Based Dynamic Data Allocation Algorithm-A Markov Chain Model Approach

[1]Mitat Uysal and [2]Tolga Ulus
[1]Department of Computer Engineering, Dogus University, Kadikoy 34722 Istanbul, Turkey
[2]Department of Management Information Systems, Boğaziçi University, Bebek 80815 İstanbul, Turkey

Abstract: In this study, a new dynamic data allocation algorithm for non-replicated Distributed Database Systems (DDS), namely the threshold algorithm, is formulated and proposed. The threshold algorithm reallocates data with respect to changing data access patterns. The proposed algorithm is distributed in the sense that each node autonomously decides whether to transfer the ownership of a fragment in DDS to another node or not. The transfer decision depends on the past accesses of the fragment. Each fragment continuously migrates from the node where it is not accessed locally more than a certain number of past accesses, namely a threshold value. The threshold algorithm is modeled for a fragment of the database as a finite Markov chain with constant node access probabilities. In the model, a special case, where all nodes have equal access probabilities except one with a different access probability, is analyzed. It has been shown that for positive threshold values the fragment will tend to remain at the node with the higher access probability. It is also shown that the greater the threshold values are, the greater the tendency of the fragment to remain at the node with higher access probability will be. The threshold algorithm is especially suitable for a DDS where data access pattern changes dynamically.

Key words: Distributed databases, dynamic data allocation, Markov chain

INTRODUCTION

Developments in database and networking technologies in the past few decades led to advances in distributed database systems. A DDS is a collection of sites connected by a communication network, in which each site is a database system in its own right, but the sites have agreed to work together, so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site (Date, 1990; Özsu and Valduriez, 1991).

The primary concern of a DDS is to design the fragmentation and allocation of the underlying database. Fragmentation unit can be a file where allocation issue becomes the file allocation problem. File allocation problem is studied extensively in the literature, started by Chu (1969) and continued for non-replicated and replicated models (Apers, 1988; Casey, 1972; Grapa and Belford, 1977; Mahmoud and Riordan 1976; Morgan and Levin, 1977; Ramamoorthy and Wah, 1983; Whitney, 1970). Some studies considered dynamic file allocation (Ames, 1977; Smith, 1981; Wah, 1979; Wang and Chen, 2005; Ahn and Kim, 2005).

Data allocation problem was introduced when Eswaran (1974) first proposed the data fragmentation. Studies on vertical fragmentation (Babad, 1977; Ceri *et al*., 1989; Hoffer, 1976; Navathe *et al*., 1984), horizontal fragmentation (Ceri *et al*., 1983) and mixed fragmentation (Chang and Cheng, 1980; Cheng *et al*., 2002; March, 1983; Sacca and Wieder hold, 1985; Sacco, 1986; Zhang and Orlowska, 1994) were conducted. The allocation of the fragments is also studied extensively (Ahmad *et al*., 2002; Apers, 1988; Bakker, 2000; Chang, 2002; Kwok *et al*., 1996; So *et al*., 1999; Zhou *et al*., 1999; Gorawski *et al*., 2005).

In these studies, data allocation has been proposed prior to the design of a database depending on some static data access patterns and/or static query patterns. In a static environment where the access probabilities of nodes to the fragments never change, a static allocation of fragments provides the best solution. However, in a dynamic environment where these probabilities change over time, the static allocation solution would degrade the database performance. Initial studies on dynamic data allocation give a framework for data redistribution (Wilson and Navathe, 1986) and demonstrate how to perform the redistribution process in minimum possible time (Rivera-Vega *et al*., 1990). In (Brunstrom *et al*., 1995), a dynamic data allocation algorithm for non-replicated database systems is proposed, but no modeling is done to analyze the algorithm. Instead, the paper focused on load balancing issue.

Corresponding Author: Mitat Uysal, Department of Computer Engineering, Dogus University, Kadikoy 34722 Istanbul, Turkey

This study proposes a new dynamic data allocation algorithm for non-replicated distributed databases and analyzes the algorithm using a finite-state Markov chain. Present study is based on the research conducted by Ulus, (1999). In this study, horizontal, vertical or mixed fragmentation can be used. Allocation unit can even be as small as a record or an attribute.

THE THRESHOLD ALGORITHM

In some cases, due to extra storage space need, it could be very costly to use the optimal algorithm (Ulus, 1999) in its original form. For a less costly algorithm, the solution is to decrease the need for extra storage space. The proposed threshold algorithm in this paper serves this purpose.

Let the number of nodes be n and let X_s denote the access probability of a node to a particular fragment. Suppose the fragment is stored in this particular node (i.e., it is the owner node). For the sake of simplicity, let X_d denote the access probability of all the other nodes to this particular fragment. The owner does local access, whereas the remaining nodes do remote access to the fragment.

The probability that the owner node does not access the fragment is (n-1) X_d. The probability that the owner node does not perform two successive accesses is $((n-1) X_d)^2$. Similarly, the probability that the owner node does not perform m successive accesses is $((n-1) X_d)^m$. Therefore, the probability that the owner node performs at least one access of m successive accesses is $1-((n-1) X_d)^m$.

Table 1 shows the probabilities that the owner node performs at least one access out of m successive accesses, where x_s ranges from 0.1 through 0.9 and where m is 5, 10, 25, 50 and 100. The values in the table are truncated to five decimal digits.

According to the table, the probability that the owner node with the access probability of 0.1 performs at least one access of ten successive accesses is 0.65132. It is trivial from the table that as the access probability of owner node increases, so as the probability that at least one local access occurs in m accesses.

Applying the same idea, a new threshold based algorithm (or threshold algorithm) can be proposed. In threshold algorithm, only one counter per fragment is stored. Figure 1 shows fragment I together with its counter. Comparing it to the optimal algorithm, this radically decreases the extra amount of storage space to just one value compared to an array of values in the optimal algorithm.

In the threshold algorithm, the initial value of the counter is zero. The counter value is increased by one for each remote access to the fragment. It is reset to zero for a local access. In other words, the counter always shows the number of successive remote accesses. Whenever the counter exceeds a predetermined threshold value, the ownership of the fragment is transferred to another node.

Table 1: The probability that at least one local access occurs in m accesses

χ_s	m = 5	m = 10	m = 25	m = 50	m = 100
0.1	0.40951	0.65132	0.92821	0.99485	0.99997
0.2	0.67232	0.89263	0.99622	0.99999	1.00000
0.3	0.83193	0.97175	0.99987	1.00000	1.00000
0.4	0.92224	0.99395	1.00000	1.00000	1.00000
0.5	0.96875	0.99902	1.00000	1.00000	1.00000
0.6	0.98976	0.99990	1.00000	1.00000	1.00000
0.7	0.99757	0.99999	1.00000	1.00000	1.00000
0.8	0.99968	1.00000	1.00000	1.00000	1.00000
0.9	0.99999	1.00000	1.00000	1.00000	1.00000

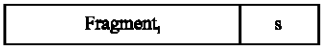

Fig. 1: Any fragment i in threshold algorithm

1. For each (locally) stored fragment, initialize the counter values to zero. (Set $s_i = 0$ for every stored fragment i).
2. Process an access request for the stored fragment.
3. If it is a local access, reset the counter of the corresponding fragment to 0 (if node j accesses fragment i, set $s_i = 0$). Go to step 2.
4. If it is a remote access, increase the counter of the corresponding fragment by one. (If fragment i is accessed remotely, set $s_i = s_i + 1$).
5. If the counter of the fragment is greater than the threshold value, reset its counter to zero and transfer the fragment to the remote node. (If $s_i > t$, set $s_i = 0$ and the fragment to remote node)
6. Go to step 2.

Fig. 2: Threshold algorithm

At this point, the critical question is which node will be the fragment's new owner. The algorithm gives very little information about the past accesses to the fragment. In fact, throughout the entire access history only the last node that accessed the fragment is known. So, there are two strategies to select the new owner. Either it is chosen randomly, or the last accessing node is chosen. In the former, the randomly chosen node could be one that has never accessed the fragment before. So picking the latter strategy is heuristically more reasonable.

Initially, all fragments are distributed to the nodes according to any method. A threshold value t is chosen. Afterwards, any node j, runs the threshold algorithm given in Fig. 2 for every fragment i, that it stores.

Threshold algorithm overcomes the volley of a fragment between two nodes provided that a threshold value greater than one is chosen. The algorithm guarantees the stay of the fragment for at least (t+1) accesses in the new node after a migration. In other words, it delays the migration of the fragment from any node for at least (t+1) accesses.

An important point in the algorithm is the choice of threshold value. This value will directly affect the mobility of the fragments. It is trivial that as the threshold value increases, the fragment will tend to stay more at a node; and as the threshold value decreases, the fragment will tend to visit more nodes.

Another point in the algorithm is the distribution of the access probabilities. If the access probabilities of all nodes for a particular fragment are equal, the fragment will visit all the nodes. The same applies for two nodes when there are two highest equal access probabilities.

MARKOV CHAIN MODEL OF THRESHOLD ALGORITHM

General Case: Let there be n nodes ($n \in Z^+$), denoted by 0 through (n -1). Let the threshold value be t ($t \in Z \cup \{0\}$). For simplicity, suppose the access probabilities of the nodes are discrete random variables. Assume the nodes have access probabilities X_0 through X_{n-1} for a particular fragment, subscripts showing the node index. The following is satisfied for the access probabilities where $X_i \in [0, 1]$ for all i = 0,.., n -1.

$$\sum_{i=0}^{n-1} X_i = 1$$

Figure 3 shows the finite state diagram of the system described.

In the diagram, two numbers determine the name of each state; first number denotes the node name where the fragment is currently stored, and the second number denotes the successive remote access counter. For example, when the system is in state 00, this means that the fragment is currently stored in node 0, and the current successive remote access counter is 0 (which implies that either last access performed on the fragment is local or the fragment has just migrated to node 0).

There are (t+1) states per node. In all these states, the fragment is stored in that particular node. These states correspond to the different values of successive remote access counter for the node.

The state transition probabilities are given next to each transition indicated by the arrows. For example, for the state 00 there are several incoming and outgoing transitions. One transition is both incoming and outgoing with a probability of x_0. This transition implies that with a probability of x_0, node 0 accesses the fragment, and the counter, that is already zero, is reset to zero and the fragment stays at node 0. As a result, the system does not change a state. Besides this transition, there is only

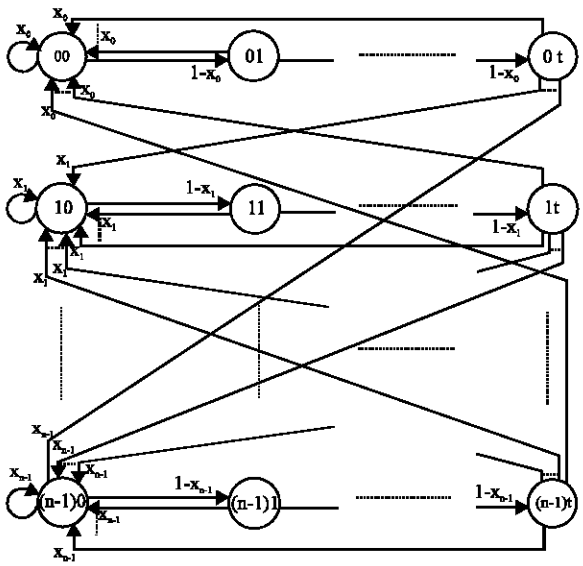

Fig. 3: Finite-state diagram of the system in general case

one outgoing transition to state 01 with a probability of $(1-x_0)$. This transition implies that with a probability of $(1-x_0)$ a remote node accesses the fragment, and the counter is increased by one, to one. As a result, the fragment still stays at node 0, but a state change from 00 to 01 takes place. Besides these two transitions, there are two groups of incoming transitions all with a probability of x_0. One group of transitions comes from the states 01 through 0t. A local access causes these transitions. As a result of these transitions, the counter is reset to zero and the fragment still stays in node 0, but it leads to a state change from the previous state (01 through 0t) to 00. The other group of transitions comes from the states 0t through (n-1)t. Before these transitions, the fragment is in a node other than node 0 and the counter is t. The transition occurs when node 0 accesses the fragment. As a result, the counter value exceeds the predetermined threshold value and the fragment is transferred to the ownership of node 0. Hence a state change from the previous state (0t through (n-1)t) to 00 occurs.

Figure 3 shows a Markov chain due to its memory less property. It is memory less because, for any state the system can enter, the next state entered depends solely on the current state of the system. Furthermore, this Markov chain has discrete-time, finite-state, irreducible, aperiodic and recurrent properties. It is discrete-time, because the state transitions occur in discrete times (when an access to the fragment is performed) and state transition duration is negligible. It is finite-state, because the number of states is finite. It is irreducible, because every state can be reached from every other state. This

Markov chain is aperiodic, because for every state, the entrance to the same state is not periodic. This Markov chain is recurrent, because it is finite-state and irreducible (Kleinrock, 1975).

Let π be a 1 by n probability vector whose elements π_k, show the steady state probability that the system is in state k.

$$\pi = \begin{bmatrix} \pi_0 & \pi_1 & \dots & \pi_k & \dots & \pi_{(n-1)} \end{bmatrix}_{1xn}$$

Let P be the n by n state transition probability matrix whose elements p_{ij}, show the state transition probability from state i to state j.

$$P = \begin{bmatrix} p_{00} & p_{01} & \cdots\cdots & p_{0(n-1)} \\ p_{10} & p_{11} & \cdots\cdots & p_{1(n-1)} \\ \vdots & \vdots & :: p_{ij} :: & \vdots \\ p_{(n-1)0} & p_{(n-1)1} & \cdots\cdots & p_{(n-1)(n-1)} \end{bmatrix}_{nxn}$$

Equation 1 defines the steady state of a discrete-time, finite-state, irreducible, aperiodic and recurrent Markov chain. Given the state transition probability matrix P, the system determines the steady-state probability vector π (Kleinrock, 1975).

$$\pi = \pi P \tag{1}$$

Readjusting Eq. 1 and 2 is obtained.

$$(P\text{-}1)' \pi' = 0 \tag{2}$$

are n equations and n unknowns. But since one of the equations is linearly dependent on the others, one more equation is needed to solve the system (Kleinrock, 1975). Last equation is, the one that shows the summation of the steady state probabilities, given by Eq. 3.

$$\sum_{i=0}^{n-1} \pi_i = 1 \tag{3}$$

Replacing the first equation in Eq. 2 by Eq. 3 and 4 is obtained.

$$Q\pi' = r \tag{4}$$

In Eq. 4, Q, π' and r are as follows.

$$Q = \begin{bmatrix} 1 & 1 & \dots & 1 \\ p_{01} & p_{11} - 1 & \dots & p_{(n-1)1} \\ \vdots & \vdots & \vdots & \vdots \\ p_{0(n-1)} & p_{1(n-1)} & \dots & p_{(n-1)(n-1)} - 1 \end{bmatrix}_{nxn}$$

$$\pi' = \begin{bmatrix} \pi_0 \\ \pi_1 \\ : \\ \pi_{(n-1)} \end{bmatrix}_{nx1} \qquad r = \begin{bmatrix} 1 \\ 0 \\ : \\ 0 \end{bmatrix}_{nx1}$$

These equations can be adapted to system in Fig. 3. For the threshold algorithm model in general case of Fig. 3, let π_g be the 1 by n probability vector and P_g be the n by n state transition probability matrix. They are as follows.

$$\pi_g = [\pi_{00}\ \pi_{01} \dots \pi_{0t}\ \pi_{10}\ \pi_{11} \dots \pi_{1t}\ \dots\ \pi_{(n-1)0}\ \pi_{(n-1)1}\ \dots\ \pi_{(n-1)t}]$$

$$P_g = \begin{bmatrix} x_0 & 1-x_0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ x_0 & 0 & 1-x_0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_0 & 0 & 0 & \dots & 1-x_0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ x_0 & 0 & 0 & \dots & 0 & x_1 & 0 & 0 & \dots & 0 & \dots & x_{n-1} & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & x_1 & 1-x_1 & 0 & \dots & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & x_1 & 0 & 1-x_1 & \dots & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & x_1 & 0 & 0 & \dots & 1-x_1 & \dots & 0 & 0 & 0 & \dots & 0 \\ x_0 & 0 & 0 & \dots & 0 & x_1 & 0 & 0 & \dots & 0 & \dots & x_{n-1} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & \dots & x_{n-1} & 1-x_{n-1} & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & \dots & x_{n-1} & 0 & 1-x_{n-1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 & \dots & x_{n-1} & 0 & 0 & \dots & 1-x_{n-1} \\ x_0 & 0 & 0 & \dots & 0 & x_1 & 0 & 0 & \dots & 0 & \dots & x_{n-1} & 0 & 0 & \dots & 0 \end{bmatrix}$$

Notice that π_g elements have two indices. First index is the node name and the second index is the successive remote access counter. And finally, Q_g is as follows.

$$Q_g = \begin{bmatrix}
1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 \\
1-x_0 & -1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
0 & 1-x_0 & -1 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1-x_0 & -1 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & x_1 & x_1-1 & x_1 & x_1 & \dots & x_1 & x_1 & \dots & 0 & 0 & 0 & \dots & 0 & x_1 \\
0 & 0 & 0 & \dots & 0 & 0 & 1-x_1 & -1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 1-x_1 & -1 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 1-x_1 & -1 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & x_{n-1} & 0 & 0 & 0 & \dots & 0 & x_{n-1} & \dots & x_{n-1}-1 & x_{n-1} & x_{n-1} & \dots & x_{n-1} & x_{n-1} \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 1-x_{n-1} & -1 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 1-x_{n-1} & -1 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1-x_{n-1} & -1
\end{bmatrix}$$

After solving for π_g vector in Eq. 4, the probabilities, that the fragment is in a particular node, are calculated as follows for all i = 0,.., (n-1) where O_i denotes the probability that the fragment is in node i (here notice that the node names are used as subscripts in calculation).

$$O_i = \sum_{j=0}^{t} \pi_{ij} \qquad (5)$$

Since the number of unknowns, namely the equilibrium probabilities in the general case is very large, it is very hard to investigate the general case situation. For the sake of simplicity, a special case, that will decrease the number of unknowns to just two, will be examined.

Special case: Assume an n node DDS. Assume further that one particular node denoted by s has an access probability of x_s to a particular fragment of DDS. Suppose all the other nodes denoted by d_1 through d_{n-1} have the equal access probabilities of x_d to the same fragment. The following equation is satisfied for the access probabilities where $x \in [0,1]$ and $x_d \in [0,1]$

$$x_s + (n-1)x_d = 1$$

The finite-state diagram of this system is given in Fig. 6.

In the Fig. 4, states s0 through st corresponds to node s that has an access probability of x_s to the fragment. For the rest of the nodes d_1 through d_{n-1}, there are the states d_i0 through d_it, (n-1) of each.

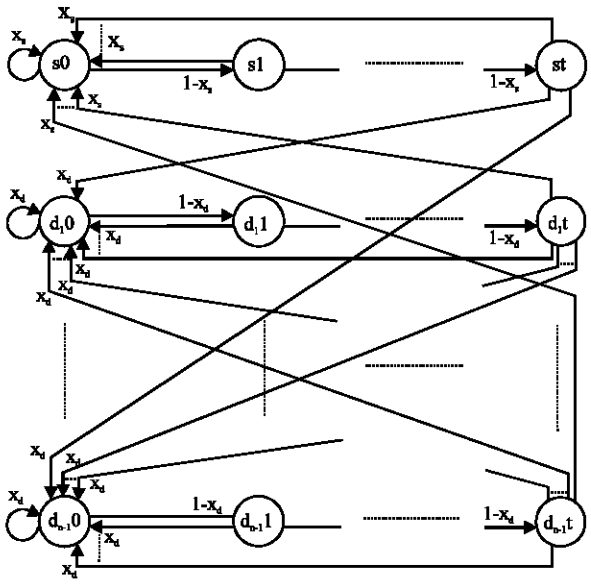

Fig. 4: Finite-state diagram of the system in special case

Lemma 1: For the system of Fig. 4, the steady state probabilities of all nodes, except node s, corresponding to a particular threshold value t, is equal. In other words,

$$\pi_{d_1 f} = \pi_{d_h f}$$

where h shows any node index varying from 1 to (n-1) and f shows any threshold value varying from 0 to t.

Proof: (Ulus.,1999).

The finite-state diagram of the system after Lemma 1 is given in Fig. 5.

In Fig. 5, states s0 through st corresponds to node s that has an access probability of x_s to the fragment. For the rest of the nodes d_1 through d_{n-1}, there are the states d0 through dt as a corollary to Lemma 1.

For the threshold algorithm model of Fig. 7, let π_m be a 1 by n(t+1) steady state probability vector and let P_m be the n(t+1) by n(t+1) state transition probability matrix. They are as follows.

$$\pi_m = [\pi_{s0} \;..\; \pi_{st} \;\; \pi_{d0} \;..\; \pi_{dt} \;....\; \pi_{d0} \;..\; \pi_{dt}]$$

$$P_m = \begin{bmatrix}
x_s & 1-x_s & 0 & ... & 0 & 0 & 0 & 0 & ... & 0 & ... & 0 & 0 & 0 & ... & 0 \\
x_s & 0 & 1-x_s & ... & 0 & 0 & 0 & 0 & ... & 0 & ... & 0 & 0 & 0 & ... & 0 \\
::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: \\
x_s & 0 & 0 & ... & 1-x_s & 0 & 0 & 0 & ... & 0 & ... & 0 & 0 & 0 & ... & 0 \\
x_s & 0 & 0 & ... & 0 & x_d & 0 & 0 & ... & 0 & ... & x_d & 0 & 0 & ... & 0 \\
0 & 0 & 0 & ... & 0 & x_d & 1-x_d & 0 & ... & 0 & ... & 0 & 0 & 0 & ... & 0 \\
0 & 0 & 0 & ... & 0 & x_d & 0 & 1-x_d & ... & 0 & ... & 0 & 0 & 0 & ... & 0 \\
::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: \\
0 & 0 & 0 & ... & 0 & x_d & 0 & 0 & ... & 1-x_d & ... & 0 & 0 & 0 & ... & 0 \\
x_s & 0 & 0 & ... & 0 & x_d & 0 & 0 & ... & 0 & ... & x_d & 0 & 0 & ... & 0 \\
::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: \\
0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & ... & 0 & ... & x_d & 1-x_d & 0 & ... & 0 \\
0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & ... & 0 & ... & x_d & 0 & 1-x_d & ... & 0 \\
::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: & ::: \\
0 & 0 & 0 & ... & 0 & 0 & 0 & 0 & ... & 0 & ... & x_d & 0 & 0 & ... & 1-x_d \\
x_s & 0 & 0 & ... & 0 & x_d & 0 & 0 & ... & 0 & ... & x_d & 0 & 0 & ... & 0
\end{bmatrix}$$

It can be easily seen that in π_m vector the elements π_{d0} through π_{dt} repeat themselves (n-1) times. The dimension of the system can be decreased as shown in Lemma 2.

Lemma 2: Let π_r be a 1 by 2(t+1) steady state probability vector and let P_r be the 2(t+1) by 2(t+1) state transition probability matrix as shown below.

$$\pi_r = [\pi_{s0} \;..\; \pi_{st} \;\; (n-1)\pi_{d0} \;..\; (n-1)\pi_{t0}]$$

$$P_r = \begin{bmatrix}
x_s & 1-x_s & 0 & .. & 0 & 0 & 0 & 0 & 0 & .. & 0 & 0 \\
x_s & 0 & 1-x_s & .. & 0 & 0 & 0 & 0 & 0 & .. & 0 & 0 \\
:: & :: & :: & :: & :: & :: & :: & :: & :: & :: & :: & :: \\
x_s & 0 & 0 & .. & 1-x_s & 0 & 0 & 0 & 0 & .. & 0 & 0 \\
x_s & 0 & 0 & .. & 0 & 1-x_s & 0 & 0 & 0 & .. & 0 & 0 \\
x_s & 0 & 0 & .. & 0 & 0 & 1-x_s & 0 & 0 & .. & 0 & 0 \\
0 & 0 & 0 & .. & 0 & 0 & x_d & 1-x_d & 0 & .. & 0 & 0 \\
0 & 0 & 0 & .. & 0 & 0 & x_d & 0 & 1-x_d & .. & 0 & 0 \\
:: & :: & :: & :: & :: & :: & :: & :: & :: & :: & :: & :: \\
0 & 0 & 0 & .. & 0 & 0 & x_d & 0 & 0 & .. & 1-x_d & 0 \\
0 & 0 & 0 & .. & 0 & 0 & x_d & 0 & 0 & .. & 0 & 1-x_d \\
x_s & 0 & 0 & .. & 0 & 0 & 1-x_s & 0 & 0 & .. & 0 & 0
\end{bmatrix}$$

The system of equations given by $\pi_m=\pi_m P_m$ and $\pi_r=\pi_r P_r$ are the same

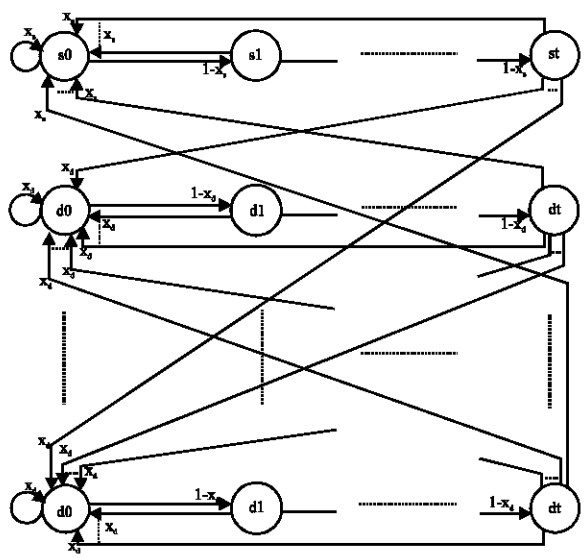

Fig. 5: Simplified finite-state diagram of the system in special case

Proof: (Ulus, 1999)

Theorem 1: Assume that the fragments of a DDS are allocated to n nodes, denoted by 0 through (n-1). Assume all nodes have equal access probability of x_d to a particular fragment except node 0, which has a different access probability of x_s where $x_s \in [0, 1]$ and $x_d \in [0, 1]$. When the threshold algorithm with a threshold t is used, the fragment will be in node 0 with the probability O_s given by

$$O_s = \frac{x_d(1-x_d)^t\left[1-(1-x_s)^{t+1}\right]}{x_d(1-x_d)^t + (1-x_s)^{t+1}\left[1-(1-x_d)^t\right]} \qquad (6)$$

where $x_s \neq 0$, $x_d \neq 0$ and $x_s \neq 1$.

Proof: (Ulus, 1999)

Theorem 2: Assume that the fragments of a DDS are allocated to n nodes, denoted by 0 through (n-1). Assume all nodes have equal access probability of x_d to a particular fragment except node 0, which has a different access probability of x_s where $x_s \in [0, 1]$ and $x_d \in [0, 1]$. When the threshold algorithm with a threshold t is used, the fragment will be in the nodes other than node 0 with the probability O_d given by

$$O_d = \frac{(1-x_s)^{t+1}\left[1-(1-x_d)^{t+1}\right]}{x_d(1-x_d)^t + (1-x_s)^{t+1}\left[1-(1-x_d)^t\right]} \qquad (7)$$

where $x_s \neq 0$, $x_d \neq 0$ and $x_s \neq 1$.

Proof: (Ulus, 1999)

Equation 6 gives the probability that the fragment is in node 0, whereas Eq. 7 gives the probability that the fragment is in the other nodes. Since the fragment is either in node 0 or in a node other than node 0, the sum of O_s and O_d is 1.

RESULTS

Let us investigate Eq. 6 and 7. Since, $O_s+O_d=1$, investigating only O_s is sufficient.

In Eq. 6, the parameters are x_s, x_d and t. In other words, the probability that the fragment is in node 0 is determined by the access probability of node 0, the access probability of the other nodes and the threshold value. Furthermore, the number of nodes, n, is another parameter, since it specifies the relationship between x_s and x_d with the following formula.

$$x_s + (n-1)\, x_d = 1$$

Now, let us find how a change in the access probabilities and the threshold value effect the probability that the fragment is in any node.

Change in Access Probability: The relation between x_s and x_d is given by the following equation.

$$x_s + (n-1)\, x_d = 1$$

Since x_s and x_d are access probabilities, the following inequalities are satisfied.

$$0 \leq x_s \leq 1 \text{ and } 0 \leq x_d \leq 1$$

When n is held constant, x_s and x_d are inversely proportional. So, it is sufficient to investigate only the change in x_s of O_s.

Lemma 3: When $X_s = 1$, $O_s = 1$.

Proof: When $x_s = 1$, all π_{sj} values of O_s given by Eq. 5 are 0 except π_{s0} value. π_{s0} value is 1 which makes O_s value 1 as well.

Lemma 4: When $x_s = 0$, $O_s = 0$.

Proof: When $x_s = 0$, all π_{sj} values of O_s given by Eq. 5 are 0 which makes O_s value 0 as well.

Lemma 5: O_s is strictly increasing with respect to x_s in the interval of (0,1).

Proof: Let us investigate the change in O_s with respect to x_s. The partial derivative of O_s with respect to x_s gives the change in O_s with respect to x_s. The partial derivative is as shown below where O_1 and O_2 are the nominator and the denominator of O_s, respectively.

$$\frac{\partial O_s}{\partial x_s} = \frac{(t+1)(1-x_s)^t\left[x_d(1-x_d)^t O_2 + \left[1-(1-x_d)^t\right]O_1\right]}{\left[O_2\right]^2}$$

It is obvious that the partial derivative is positive for all $x_s \in [0,1]$. Therefore, O_s is strictly increasing with respect to x_s in (0,1).

Figure 6 shows the behaviour of O_s as a function of x_s in a five-node system. Figure 6 is drawn for three different threshold values, 0, 3 and 10.

For the threshold of 0, O_s is a linear function of x_s with a slope of 1. This means that when the threshold is 0, the access probability of a node directly gives the steady-state probability that the fragment is in the corresponding node.

For threshold values of 3 and 10, notice the change in steepness of the curve.

Change in threshold value: Threshold t can take only non-negative integer values. Let us investigate under which circumstances O_s is increasing or decreasing with respect to t.

Lemma 6: The following holds for the change in O_s with respect to t, provided that $x_s \neq 0$, $x_d \neq 0$ and $x_s \neq 1$:

- When $x_s = x_d$, O_s is constant with respect to t.
- When $x_s > x_d$, O_s is increasing with respect to t.
- When $x_s < x_d$, O_s is decreasing with respect to t.

Proof: To investigate the behaviour of O_s with respect to the threshold, the partial derivative of O_s with respect to t should be examined. But since O_s is defined only for non-negative integer values of t, it is not continuous for t. Therefore it is not possible to find the partial derivative of O_s with respect to t. Instead, to investigate the sign of the difference $O_s(r+1) - O_s(r)$, for any positive integer r, would be sufficient. If the sign of this expression is positive, the probability will be increasing. Otherwise it will be decreasing.

For simplicity, let us substitute a and b given by the equations $a = 1-x_s$ and $b = 1-x_d$ in Eq. 6. The difference will be as follows.

$$O_s(r+1) - O_s(r) = \frac{(1-b)a^{r+1}b^r\left[b-a+a^{r+2}(1-b)-b^{r+2}(1-a)\right]}{\left(a^{r+1}(1-b^r)+b^r(1-b)\right)\left(a^{r+2}(1-b^{r+1})+b^{r+1}(1-b)\right)}$$

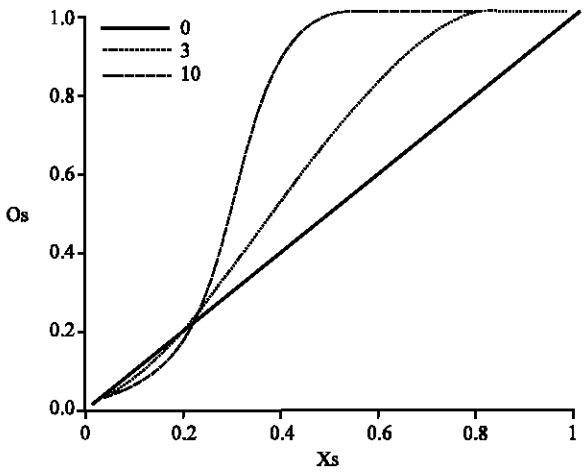

Fig. 6: O_s as a function of x_s in a five-node system for thresholds 0, 3 and 10.

In this expression, all the terms except $(b-a+a^{r+2}(1-b)-b^{r+2}(1-a)$ in the nominator are positive provided that $x_s \neq 0$, $x_d \neq 0$ and $x_s \neq 1$. Only the sign of this term determines the sign of the whole expression. Let D denote this term and let us substitute a and b expressions back in. The result is as follows.

$$D = x_s\left[1-(1-x_d)^{r+2}\right] - x_d\left[1-(1-x_s)^{r+2}\right]$$

Let us multiply and divide D by $x_s x_d$ and readjust it. The expression takes the following form.

$$D = x_s x_d\left[\frac{\left[1-(1-x_d)^{r+2}\right]}{x_d} - \frac{\left[1-(1-x_s)^{r+2}\right]}{x_s}\right]$$

Applying Eq. C.2, D is found as follows.

$$D = x_s x_d \sum_{i=0}^{r+1}\left[(1-x_d)^i - (1-x_s)^i\right]$$

The sign of D depends on the relation between x_s and x_d. According to this:

- If $x_s = x_d$, D is zero. Therefore, when $x_s = x_d$, O_s is constant with respect to t.
- If $x_s > x_d$, D is positive. Therefore, when $x_s > x_d$, O_s is increasing with respect to t.
- If $x_s < x_d$, D is negative. Therefore when $x_s < x_d$, O_s is decreasing with respect to t.

Lemma 7: $\lim_{t\to\infty} O_s = 1$ provided that $x_d \neq 0$.

Proof: Readjusting Eq.6, the following formula is obtained:

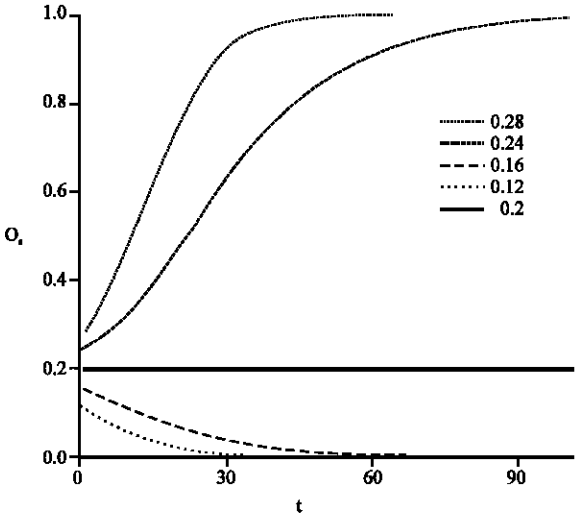

Fig. 7: O_s as a function of t in a five-node system for x_s values of 0.28, 0.24, 0.16, 0.12 and 0.2

$$O_s = \frac{x_d\left[1-(1-x_s)^{t+1}\right]}{x_d + \frac{(1-x_s)^{t+1}\left[1-(1-x_d)^t\right]}{(1-x_d)^t}}$$

Using this formula, provided that $x_d \neq 0$:

$$\underset{t\to\infty}{\text{Limit}}\, O_s = \frac{x_d \times 1}{x_d + \frac{0\times 1}{0}} = \frac{x_d}{x_d} = 1$$

Figure 7 shows the behavour of O_s as a function of t in a five-node system. Figure 7 is drawn for five different access probabilities x_s of 0.28, 0.24, 0.2, 0.16 and 0.12.

For 0.28 and 0.24, O_s converges to one. This is because $x_s > x_d$. Noticing the change in steepness of two curves, it converges faster for greater access probabilities. For 0.2, O_s is constant at 0.2. This is because $x_s = x_d$. In this case, the access probability of a node directly gives the steady-state probability that the fragment is in the corresponding node.

For 0.16 and 0.12, O_s converges to zero. This is because $x_s < x_d$. Noticing the change in steepness of two curves, it converges faster for smaller access probabilities.

CONCLUSIONS

In this study, a new dynamic data allocation algorithm, namely threshold algorithm, for non-replicated DSSs is introduced. In the threshold algorithm, the fragments, previously distributed over a DDS, are continuously reallocated according to the changing data access patterns. The node in which a fragment is stored is considered the owner of that particular fragment. When its owner in the past few successive accesses, specified by the threshold value, never accesses a fragment, the ownership of the fragment is transferred to another node.

The threshold algorithm is modeled using a finite-state Markov chain. To simplify the model, a special case where the access probabilities of the nodes are all equal except a single node is examined. The equilibrium probabilities for a fragment in any node are obtained in terms of access probabilities and the threshold value. The behavior of a fragment, in reaction to a change in access probabilities or to a change in threshold value, is investigated. It is shown that the fragment tends to stay at the node with higher access probability. As the access probability of the node increases, the tendency to remain at this node also increases. It is also shown that as the threshold value increases, the fragment will tend to stay more at the node with higher access probability.

Threshold algorithm can be used for dynamic data allocation to enhance the performance of non-replicated DDSs. For further research, the algorithm can be extended to use on the replicated DSSs as in (Sistla *et al*., 1998; Wolfson *et al*., 1995, 1997).

REFERENCES

Ahmad, I., K. Karlapalem, Y.K. and S.K. Kwok, 2002. So, Evolutionary algorithms for allocating data in distributed database systems. Distributed and Parallel Databases, 11: 5-32.

Ahn, K. and D.H. Kim, 2005. Implementation of a database management system for the comprehensive use of severe accident risk information. Progress in Nuclear Energy, 46: 57-76.

Ames, J.E., 1977. Dynamic file allocation in a distributed database system. Ph.D. Thesis, Duke University, Durham .

Apers, P.M.G., 1988. Data allocation in distributed database systems. ACM Transactions on Database Systems, 13: 263-304.

Babad, M.J., 1977. A record and file partitioning model. Comm. ACM., 20: 22-31.

Bakker, J.A., 2000. Semantic partitioning as a basis for parallel i/o in database management systems. Parallel Computing, 26: 1491-1513.

Brunstrom, A. S.T. Leutenegger and R. Simha, 1995. Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads. In: IEEE Proc. Fourth Int. Conf. Inf. Knowl. Man., Baltimore, MD., pp: 395-402.

Casey, R.G., 1972. Allocation of copies of a file in an information network. In: Proc. AFIPS Spring Joint Computer Conf., Atlantic City, pp: 617-625.

Ceri, S., S.B. Navathe and G. Wiederhold, 1983. Distribution design of logical database schemas. IEEE Trans. Software Engineering, 9: 487-503.

Ceri, S., B. Pernici and G. Wiederhold, 1989. Optimization problems and solution methods in the design of data distribution. Information Systems, 14: 261-272.

Chang, S.K. and W.H. Cheng, 1980. A methodology for structured database decomposition. IEEE Trans. Software Engineering, 6: 205-218.

Chang, C.T., 2002. Optimization approach for data allocation in multidisk database. Eur. J. Operational Res., 143: 210-217.

Cheng, C.H., W.K. Lee and K.F. Wong, 2002. A genetic algorithm-based clustering approach for database partitioning. IEEE Trans. Systems Man and Cybernetics Part C-Applications and Reviews, 32: 215-230.

Chu, W.W., 1969. Optimal file allocation in a multiple computer system. IEEE Trans. Computers, 18: 885-889.

Date, C.J., 1990. An Introduction to Database Systems Vol. I, 5th Edn., Addison-Wesley: Reading.

Eswaran, K.P., 1974. Placement of records in a file and file allocation in a computer network. In: Proc. IFIP Cong. on Information Processing, Stockholm, Sweden, pp: 304-307.

Gorawski, M. and R. Chechelski, 2005. Parallel telemetric data warehouse balancing algorithm. Proceedings 5th Intl. Conf. on Intelligent Syst. Design and Applied, 8: 387-392

Grapa, E. and G.G. Belford, 1977. Some theorems to aid in solving the file allocation problem. Comm. ACM., 20: 878-882.

Hoffer, J.A., 1976. An Integer programming formulation of computer database design problems. Information Science, 11: 29-48.

Kleinrock, L., 1975. Queueing Systems Vol. I: Theory, John Wiley and Sons: New York.

Kwok, Y.K., K. Karlapalem and I.M.P. Ng Ahmad, 1996. Design and evaluation of data allocation algorithms for distributed multimedia database systems. IEEE J. on Selected Areas in Communications, 14: 1332-1348.

Mahmoud, S. and J.S. Riordan, 1976. Optimal allocation of resources in distributed information networks. ACM Transaction on Database Systems, 1: 66-78.

March, S.T., 1983. Techniques for structuring database records. ACM Computing Surveys, 15: 45-79.

Morgan, H.L. and K.D. Levin, 1977. Optimal program and data locations in computer networks. Comm. ACM., 20: 315-321.

Navathe, S.B., S. Ceri, G. Wiederhold and J. Dou, 1984. Vertical partitioning algorithms for database design. ACM Transaction on Database Systems, 9: 680-710.

Özsu, T. and P. Valduriez, 1991. Principles of Distributed Database Systems, Prentice-Hall: Englewood Cliff.

Ramamoorthy, C.V. and B.W. Wah, 1983. The isomorphism of simple file allocation. IEEE Trans. Computers, 23: 221-231.

Rivera-Vega, P.I. R. Varadarajan and S.B. Navathe, 1990. Scheduling data redistribution in distributed databases. In: IEEE Proc. 6th Intl. Conf. Data Eng., pp: 166-173.

Sacca, D. and G. Wiederhold, 1985. Database partitioning in a cluster of processors. ACM Transaction on Database Systems, 10: 28-56.

Sacco, G., 1986. Fragmentation: A technique for efficient query processing. ACM Transaction on Database Systems, 11: 113-133.

Sistla, A.P., O. Wolfson and Y. Huang, 1998. Minimization of communication cost through caching in mobile environments. IEEE Trans. Parallel Distributed Systems, 9: 378-390.

Smith, A.J., 1981. Long-term file migration: Development and evaluation of algorithms. Comm. ACM., 24: 512-532.

So, S.K., I. Ahmad and K. Karlapalem, 1999. Response time driven multimedia data objects allocation for browsing documents in distributed environments. IEEE Trans. Knowledge and Data Engineering, 11: 386-405.

Ulus, T., 1999. Data Allocation algorithms in distributed database systems (In Turkish), Ph.D. Thesis, Istanbul University, Istanbul.

Wah, B.W., 1979. Data management in distributed systems and distributed data bases, Ph.D. Thesis, University of California, Berkeley.

Wang, S. and H.L. Chen, 2005. Near-Optimal Data Allocation over multiple broadcast channels, Computer Communications, (In Press).

Whitney, V.K.M., 1970. A study of optimal file assignment and communication network configuration in remote access computer message processing and communication systems, Ph.D. Thesis, University of Michigan, Ann Arbor.

Wilson, B. and S.B. Navathe, 1986. An analytical framework for the redesign of distributed databases. in Proceeding of the 6th Advanced Database Symposium, Tokyo, Japan, pp: 77-83.

Wolfson, O. and S. Jajodia, 1995. An algorithm for dynamic data allocation in distributed systems. Information Processing Letters, 53: 13-119.

Wolfson, O. and S. Jajodia, 1997. An adaptive data replication algorithm. ACM Transaction on Database Systems, 22: 255-314.

Zhang, Y. and M.E. Orlowska, 1994. On fragmentation approaches for distributed database design. Information Science, 1: 117-132.

Zhou, S., H.M. Williams and K.F. Wong, 1999. Data placement in shared-nothing database systems. High Performance Cluster Computing, 2: 440-453.