Rate Adoptive Intelligent Transaction Caching In Distributed Mobile Environments Using Learning Automata

Mahdi Bazarganigilani

Charles Sturt University, Australia
Mahdi62b@yahoo.com

Abstract

Rapid advances in cellular communications, wireless networks and satellite services, leading to the emergence of moving computing systems. Mobility can be problematic when the data is kept and need to be calculated in the databases with moving clients. Disconnection of the mobile unit with wireless networks and the information centre is rampant. When the mobile client is disconnected the information is disrupted and processes and operations are done from scratch. In this paper, a method for predicting the future of information that the user uses them is suggested. Users' data are saved in cache of each client and are available during disconnected times. Such caching schemes avoid too much overhead due to loss of connection. This paper employs a novel method to suggest the nearest records based on vicinity of the interests to the user. Moreover, this work employs an efficient mechanism to update items in cache storage. This avoids from stale data to be served on each client.

Keywords: *Distributed Mobile Environment, Mobile Transaction Caching, Cache Consistency, Data Suggestion*

1. Introduction

In mobile data base system, caching and suggestion of feature transactions and data play an important role in reducing the communication between server and clients. Reducing such connections leads to immense saving in valuable bandwidth. Caching possible items in each client improves the autonomousness of ach moving unit (MU). In mobile environments, Each MU interacts with other mobile supportive stations (MSS). The main control of data is accomplished by MSSs. MSSs interact with main DBMS to commit or rollback transactions [1-5]. Caching items in MSS results in high improvement in performance [6]. In Pro-Motion model [7], authors have suggested a way, which send the results to MUs. Each MU receives compacts based on its request and sends back the committed compacts for validation to MSS. MSS then caches the validated data. In other similar works the MU sends the transaction to MSS and it commits the transaction. In this way, by moving the MU , their transaction processes may incurred on other MSSs. This environment supports such continues process of moving MU and exchanging transaction execution [8]. While such models just cache the necessary data on MSS devices. It incurs more overhead on MSSs. On the other hand, it can not effectively reduce the connections overhead in wireless networks. On the other hand, in clustering methods [8], the data is distributed in several clusters on connected distributed MSSs. MU devices connect to each MSSs and can effectively cache entire cluster based on their request. In discounted times, the MU interacts with cached cluster and it improves the performance in wireless environment. However, there should be mechanism to maintain the cache consistency. In this way, a TTL property is devoted to each cached item for keeping its maintenance and validity in caching storage [9].

This study proposes a novel approach for suggesting data items to each customer based on semantic similarity to other customers. We employ the fact that users which have previously used similar records, more likely to employ similar transactions. For evaluating the semantic similarity for each couple of customers, we use the concept of Distributed learning automata [10]. Moreover, we employ a mechanism [11] to adopt the caching maintenance for each item in cache item set. We conclude our approach saves much bandwidth in comparison to other similar works.

2. Previous Works

In much recent works, Huang, et. al., [12] introduced a novel method to discover the most frequent queries for caching based on previous frequent query which requested. They mapped each query to it OLAP definition [13] and then encoded OLAP queries for each customer. They suppose, there is some I/O over head in caching mechanism in data warehouse system. Based on that consumption they evaluate if they should connect directly to the server or use cache mechanism. Moreover, they employ an incremental approach [14] for updating the database of frequent queries for each user. Consequently, they obtain the most frequent OLAP levels for each customer based on rule mining algorithm. Such algorithms incur so much overhead on each client. Finding the mining rules based on an changing transaction incur too much overhead on each client which is not suitable in many wireless networks with constraint of processing and energy on each client.

Mershad, and Artail introduced COACS system [15] In COACS, elected query directory (QD) nodes cache submitted queries and use them as indexes to data stored in the nodes that initially requested them (CN nodes). In this case each request makes the RN (requesting node) to look for the nearest (QD). The request can be forwarded to other neighbors QDs upon unsuccessful hit or invalid data. Finally, when a miss occurs, it should be fetched directly from database. While such system obtains a high hit ratio rate, but they have very security drawbacks. Saving each submitted query in various OD nodes are susceptible to many security holes. Some queries are very important and hackers can access them by just hacking one of QDs. Another problem is overhead which incur on wireless connections. However, authors concluded their system is very efficient when the cost of connection to base server is high. While, they achieve such goal, they incur much overhead to wireless connections and expose many queries to various vulnerable QDs.

Since COACS did not implement a consistency strategy Mershad, and Artail, implemented another system [11]. It makes the cache items consistent with their version at the server. This is accomplished by adopting the update rates at server with client request at moving devices for each data item. In this way, each request for any query make the QD to compute the local request rate and updating request rate for that particular item. They employ two thresholds, which both are greater than 1. If the division of updating rate on local requests is greater than the last threshold, updates are not sent to the QD and consequently it causes a miss .On the other hand, while the ratio is grate than 1 and less than the first threshold, updates are sent to QD and a miss occurs. If the ratio is less than 1, a hit is counted. While this system inherits the drawback of previous work, it has less hit ratio percentage. It has also more overhead on wireless communication due to caching consistency implementation.

In another work [16], Mershad, and Artail extended the COACS by semantically comparing each submitted request with all cached queries. The semantic analysis process includes trimming the request into fragments and joining the answers of these fragments to produce the answer of the request. Authors achieved a high hit ratio. However, the system has

security drawbacks. Moreover, the items in cache can be stale since there is no mechanism for updating the caching item sets in QDs.

While proposed system are concluded very effective an efficient, each system has its own drawbacks. Furthermore, there is no prediction in caching item sets. All possible queries are saved in various QDs. This also results in less autonomous characteristic for each client.

Khoozani, et. al., [17], introduced a novel method to fetch the relevant records while each request. They compute the nearest neighbors and fetch the most frequent records, which have the most near neighbor. They define the neighbor as the moving agent, which has the most similarity based on the previous fetched records. The main drawback of their work is not employing any intelligent techniques to obtain the most accurate neighbors. They solely rely on ranking the records and computing the cosine similarity between such ranked records. Moreover, they do not employ any cache persistency mechanism for avoiding stale items in cache. They do not achieve a very high hit ratio in comparison to other similar works.

3. Database Mobile Distributed Environments

In this section, we briefly describe the structure of distributed systems, which we propose our systems. In mobile distributed environment, each MU should connect to MSS to fetch the relevant data. Such connections are through wireless networks. Each request is delivered as a transaction. Each transaction contains several update or write on main database. DBMS on MSS assure the correctness of execution each transaction. That means the integrity of each update for each transaction. Any error in execution results in roll back of the transaction.

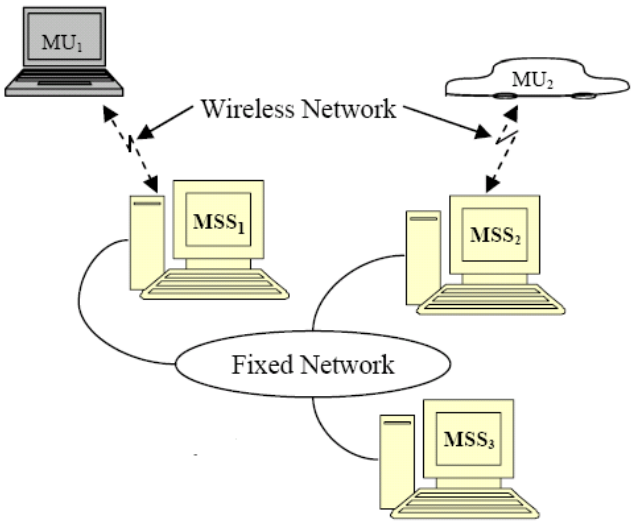


Figure 1. Mobile Distributed Wireless Database Systems

Each transaction is committed in two ways. When a moving unit requests a transaction, the information is present in MU cache. It executes the transaction from the cache. In another condition, the entire data may not be present at client's cache. In this way, the request is transferred to MSS and the transaction is executed on MSS.

On the other hand, transaction can be executed in two ways. Transaction may be sent to MSS and executed on them and the results are sent back to the MUs. In this way, it results in massive overhead on MSSs. Another method, which each request is handled by MU's DBMS. In this way, the relevant data should be transferred to MUs from MSSs. The main problem in

wireless connection is that any disruption may result in role backing the transaction. Therefore, caching item sets and prediction the relevant data for transferring to MUs' caching system is the most efficient way. On the other hand, the overhead on MSSs is decreased. In this paper, we employ the second approach to transfer the relevant data to each MU. We use the most accurate intelligent techniques to identify which data item sets should be transferred to each MU per request. Predicting and suggestion the most relevant items based on MU's usage are the most critical point in wireless database transactions. Employing the accurate technique for suggestion result in high effectiveness and hit ration of transaction caching scheme.

In proposed system, each MSS contains some agents regarding to each MU devices. MUs may change the position and moving to are of other MSSs [17].

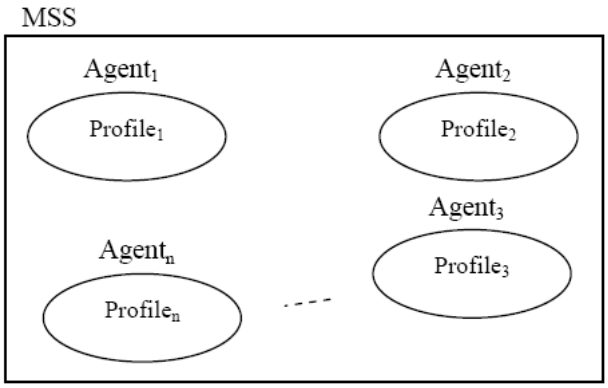


Figure 2. Mobile Distributed Wireless Database Systems

Each MSS devote to each MU a unique ID. This unique ID is used for consecutive interactions between the MU and its MSS service. MU may disconnect from the service and on the consecutive connection establishment. MSS distinguish the MU from its ID. If the MU has passed to other MSSs, the MSSs can retrieve the MU's profile by connecting to previous MSS.

Each MSS uses its unique ID and a MU's ID to assign each MU with a traceable ID.

$$Agent\_ID = MSS\_ID + MU\_ID \qquad (1)$$

Each agent contains a table which saves indexes of each fetched record. Agents can connect to each other by using MSSs. MSSs also connect to each other by wired network. The overhead time is small since it is covered by wired network. Such information is employed for ranking each record based on corresponding MU 's usage and interest. If a particular record is fetched by a MU more frequently and in more recent time, it is ranked higher for caching. Such records aids to define the category of entities based on that record. Such ranks also used to identify the near MU which has fetched the similar records based on frequency and recency.

On each request for fetching the new data from each MU, the corresponding MSS should retrieve the requested data and also suggest the most accurate data based on the current

situation of other MSS and similar agents. The suggestion period is accomplished using following steps:

1. Identifies near neighbors to current MU.
2. Ranks the neighbors using intelligent techniques
3. Identifying the relevant category for the most ranked records.
4. Suggests records from the gained category based on near neighbors.

As described earlier the first step is to identify the potential neighbors to current MU agent. Neighbor is declared as one agent, which have the most similar fetching records and transaction similar to current MU's agent. Consequently to rank such agent, a distributed learning automata (DLA) is employed. Beigy, et. al., [10] suggested DLA for suggestion system using semantic similarity relation between two identities. This model works very effective in interactive and autonomous environment. For each correlation and similarity between two agents, they receive reward. O the other hand, other agents get penalty.

Each record in database is devoted to particular category. Each category can be vertical division of particular database table. The suggestion algorithm selects the category from the records with highest validity and ranks. Consequently, records which exist in more close neighbors are selected from that particular category.

4. Index Structure And Ranking

As described, each MU's agent saves an index for each record which fetched for the MU. The values in this index are computed using previous usage of current records. A sample index is shown in Table 1.

Table 1. Index Structure

Agent (A)	
T_C	Table Code
R_C	Record Code
T_R	Last Date Sending Time
W_R	Weight Request
W_U	Weight Update
$Vote$	Vote
$R = \lambda_U / \lambda_R$	Ratio of Server Update rate to local request rate.

In table I, T_C represent the unique identifier of each table. R_C shows the unique identifier of each record which is the primary key. T_R represents the last time the record is fetched from database. Agents usually delete the records which have this time passed considerably from current time. W_R represents the frequency of access to current record while W_U denotes just

to updating frequencies. $Vote$ is calculated using previous parameters. This parameter shows the final importance of current record to the agent. Records with higher Vote value have more chance to be fetched in near feature sine they have more frequently visited or updated in more recent times.

R represents the ratio of server updates to current request rates of the agent. This parameter is used to adapt the rate of local updating information of each cache to server updates for each MU. While sending the suggested records to each MU, if this ratio is greater than a threshold, the record will not be sent since it is not a valid item for that record on a particular moment. This parameter is described in next sections in more details.

$Vote$ for each record is calculated based on T_R , W_R and W_U . Each one is assigned with a coefficient [17].

$$Vote_i = (T_R * 0.6) + (W_R * 0.25) + (W_U * 0.15) \quad (2)$$

5. Distributed Learning Automata

Learning Automata is an abstract model for interaction between environments, it has some finite actions which interacts with environment and select the best one according to the feedback from the environment.

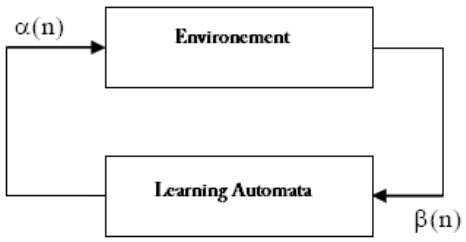


Figure 3. Interaction between Environment and Learning Automata

Figure1, depicts the relation between environment and learning automata.

Environment can be represented by a triple statement $E = \{\alpha, \beta, c\}$ in which α represents the input actions $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ and β represents the output actions $\beta = \{\beta_1, \beta_2, ...\beta_m\}$ also C represents the penalty factors $c = \{c_1, c_2, ..., c_r\}$. β Can have two values $p = 0$ for penalty factor and $p = 1$ for positive reaction. In static structure the penalty factors would remain fixed .While, in variable structure learning automata P represents the set of probabilities of each action $p = \{p_1, p_2, ..., p_r\}$ and also $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ is the learning algorithm. In this kind of automata if the action α_i would be selected in the step n and it resulted in a positive reaction the probability

of action α_i would be increased and other actions' would be decreased. If the action receives a suitable response from the environment the probabilities would be changed as follow.

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)p_j(n) \quad \forall j \qquad j \neq i \qquad (3)$$

In which a is the encouragement factor, the sum of all new probability would be 1 after the changes.

A DLA is a network of learning automata in which they cooperate with each other, every time just one automata is active, the number of action one automata can perform is the same as the number of automatas connected to it, every action triggers the peer automata. A DLA represented with a graph, an edge (LA_i, LA_j) shows the action α_j^i in LA_i triggers the LA_j .the probability of actions would be represented $p^k = \{p_1^k, p_2^k, ...p_{r_k}^k\}$ in which p_m^k is the probability of action α_m^k which triggers LA_m. For more information on DLAs please refer to references [10, 18].

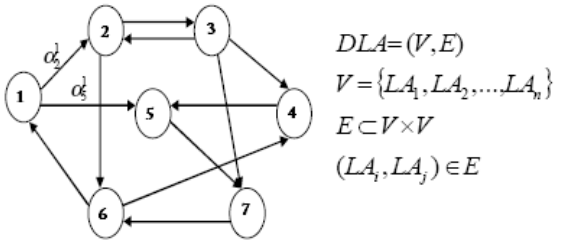


Figure 4. A DLA with 7 Learning Automata's

DLA is very effective intelligent technique to identify the relation between agents based on past correlations. We have employed DLA in a web page classification purpose [10]. Such pages are requested based on Poisson distribution. Such requests determine the correlation between identities each time. The correlations are updated based on each new request which denote any relation. This problem is applicable to context of this work. Each fetched record for each MU change the correlation between the current MU and other MUs. Such LAs score each connectivity based on previous fetched record. Each time such connectivity can be rewarded. O the other hand, other connectivity will be penalized. Such networks of LA can very well represent the relation among agents in different time periods. Such DLA can also effectively show and distinguish the changing of agents toward each other by passing time. On each request for fetching new records for each agent, the index table is updated. This update results in changes in record votes. Such changes affect the DLA and its relation to other agents. Such DLA represent the current relation among various agents in different period of times.

6. Record Suggestion and Identifying the Nearest Neighbors

As described in previous section. One of steps to suggest records on each request is identifying the closest neighbors. Neighbors do not imply a physical concept of neighbor.

That means, it is not close to current MU' agent in distance. Rather, it is near to current agent based on previous usage of similar records.

To compute the nearest neighbors, we employ the Vote parameter in each index for current agent. We compute the Euclidian distance of by employing Vote and Last date sending time. In this way, the Euclidean distance is calculated for each couple agents.

$$Euclidean(A,j)=\sqrt{\sum_{i=1}^{K} TR_{A,i} * \left(Vote_{A,i} - Vote_{j,i}\right)^2} \quad (4)$$

The above formula gives the Euclidean distance of Agent A and j. K is the number of common records and A is the current agent. $TR_{A,i}$ is the last sending date time for record i on agent A. If votes have more distances and they have been sent in more recent times, the distance of couple agents gets higher. We compute this distance for each agent in comparison to current agent.

For each agent, we consider a LA to compute its similarity and distance to other agents in different periods of times. Employing the LA will represent the similarity more accurate for such request distribution. Such request hits each agent on different time. Such hits is according to Poisson distribution. To effectively compute the similarity of agents on particular time, we employ DLA and obtain each similarity in proportion to its distance gained in formula 4.

suppose $p^k = \left\{p_1^k, p_2^k, ...p_{r_k}^k\right\}$ is the probability vector of LA_k which devoted to agent k and p_m^k is the probability of action α_m^k and r is the number of agents. For each agent the Euclidean distance is computed. The edge $D_k \to D_m$ (representing the similarity between agent k and m) of LA_k is updated based on distance value.

$$p_m^k(n+1) = p_m^k(n) + a_m^k\left[1 - p_m^k(n)\right]$$
$$p_{ij}^k(n+1) = \left(1 - a_m^k\right)p_j^k(n) \quad j \neq m \quad \forall j$$
$$a_m^k = \frac{E_m^k}{1+E_m^k}$$
$$E_m^k = -\frac{\left(p_m^k \log p_m^k + \left(1-p_m^k\right)\log\left(1-p_m^k\right)\right)}{Euclidean(k,m)} \quad (5)$$

The above value for E_m^k denotes the relation between agent k and m. More values imply more relevancies. As it is clear the encouraging factor changes when every probability among agents is updated. The algorithm of web relation is as follow:

1-Creates a DLA for the current agent

2- Initializes the probability vectors based on distances. Less distance values have more probability

3- For every user do the bellow steps is accomplished

4- Updates $D_k \to D_m$ based on similarity distance from formula 5.

7. Data Categories And Suggestion Algorithm

For more accuracy of suggestion algorithm, the records are selected from predefined categories. Such categories are usually are vertical partition of tables. For example in an Employee table, one partition could be employee which are "Manager". Each table in data base is partitioned to some categories. Such categories are considered with help of an expert. Each category may have subcategories. The following figure shows a sample categorization for a library system. This system includes three main tables as. Members, Books and Employees. For example, one category is the all books and its subcategories are English or Spanish books. Another category is all the employees which have subcategories such as Students or Staff [17].

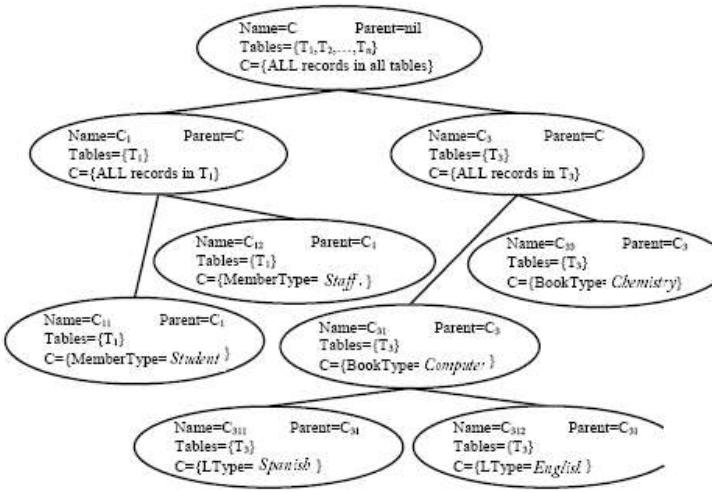


Figure 5. Data Classifcation

Suggestion algorithm is accomplished from categories most related to the record with highest Vote value. The suggestion algorithm include following steps:

1. Selects from index table the record number R_C and table code T_C with highest Vote value.
2. From R_C chooses the suitable category.
3. Selects the records from this category which have not been used recently.
4. Ranks records based on following formula.

$$Rank\left(R_m^i\right)=\sum_{k=1}^{c} p_k^m \qquad (6)$$

The records from the selected category which have more near neighbors are ranked higher.

8. Consistency Implementation

As described, we implement a cache consistency approach similar to works in [11]. However, we show our approach is much more efficient in term of bandwidth consumption

and response times. Mershad and Artail propose a consistency caching system which adopts updating in caching client based on rate of update in real database server. Their system is well works under their previous implementation of COACS system [15]. As described previously, their system has drawbacks in regard of saving all queries in QDs. Each miss causes to traversing all QDs or even direct update from server. Such scheme vastly consumes inbound bandwidth in wireless network. Moreover, there is security concern about this implementation. The most crucial drawback for such system is lack of autonomousness of each client. Clients rely on other client to execute their own query.

We assume both update rate and caching rate follow random process and the variables are exponential [20]. The probability density function is as follow:

$$P_R(t) = \lambda_R e^{-\lambda_R t}, P_U(t) = \lambda_U e^{-\lambda_U t} \qquad (7)$$

Two cases are considered. In C1, $\lambda_U / \lambda_R \leq 1$ and all updates are sent to caching index tables. In C2, if $\lambda_U / \lambda_R \geq \Gamma$, updates are not sent to server. $\lambda_U / \lambda_R \leq \gamma$ updates are sent to server. The two scenarios are shown in Figure 6 [11].

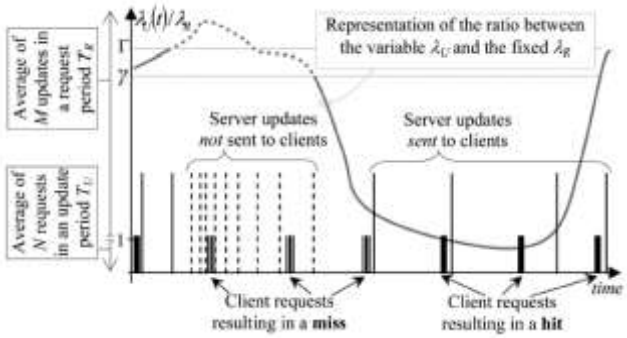


Figure 6. Illustration of Different Zones while both Rate Changes [11]

Both the bandwidth gain Gb and response time gain Gt are influenced by the number of data requests issued by requesting client relative to the number of data updates that occur at the server [11]. For analyzing the efficiency and gain time and bandwidth, we consider the following definitions:

- HC: is the average number of hops between MSSs in wired network. It applies when a packet is sent between the server and the random node.
- T(in) : is the time delay in wired network for each neighboring hop.
- T(out): is the time delay in wire-less network.
- SD: size of data items.
- SR: size of each request.

Computing the bandwidth gain in C2S1, each request cause a hit in cache system while updates are sent to the caching client.

$$Gb_{C2S1} = SR * HC + SD * HC - M * SD * HC$$

Computing the bandwidth gain in C2S2 , since the server will not send the updates and a miss occurs each time. Therefore, the gain is zero.

$$Gb_{C2S2} = 0$$

In C1 , each request causes a hit and one update for the server which reduces the bandwidth gain.

$$Gb_{C1} = N * (SR * HC + SD * HC) - SD * HC$$

We then compute the response time gain for each condition as follow,

$$Gt_{C2S1} = T(out) + HC * T(in) + T(out) - M * (HC * T(in) + T(out))$$

$$Gb_{C2S2} = 0$$

$$Gb_{C1} = N * (T(out) + HC * T(in) + T(out)) - HC * T(in) + T(out)$$

Calculating the probabilities similar to [11].

$$P_R(C1) = P_R(\lambda_u \prec \lambda_R) = \int_0^\infty \int_{t_R}^\infty P_R(t_R) P_U(t_U) dt_U dt_R$$

$$= \frac{\lambda_R}{\lambda_U + \lambda_R}$$

$$P_R(C2S2) = \frac{\lambda_R}{\lambda_U + \gamma\lambda_R} \left(\begin{array}{l} 1 + (\Gamma\lambda_R / \lambda_U)^{\left(\frac{\lambda_U + \Gamma\lambda_U}{\lambda_U}\right)} - \\ (\gamma\lambda_R / \lambda_U)^{\left(\frac{\lambda_U + \Gamma\lambda_U}{\lambda_U}\right)} \end{array} \right) +$$

$$\frac{\lambda_U}{\lambda_U + \Gamma\lambda_R} \left((\gamma\lambda_R / \lambda_U)^{\left(\frac{\lambda_U + \Gamma\lambda_R}{\lambda_U}\right)} - (\Gamma\lambda_R / \lambda_U)^{\left(\frac{\lambda_U + \Gamma\lambda_R}{\lambda_U}\right)} \right)$$

$$P_R(C2S1) = \frac{\lambda_R}{\lambda_U + \lambda_R} - P_R(C2S2)$$

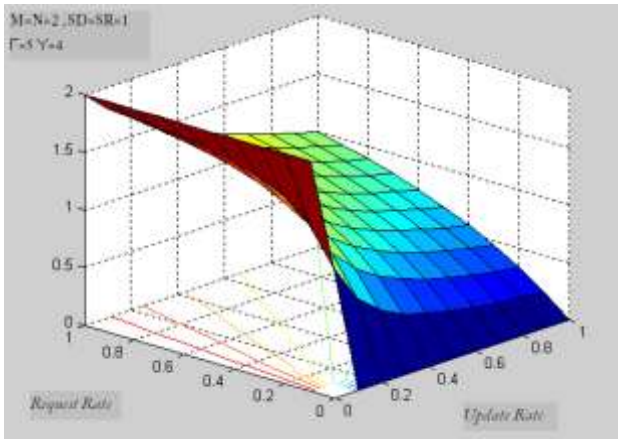


Figure 7. Bandwidth Gain Versus Update and Request Rate

Figure 7 illustrates the bandwidth gain with fixed parameters. As shown, the bandwidth gain is always positive and this is the main advantage of our design. Moreover, the figure shows as the request rate increases the bandwidth gain improves. Moreover, by increasing λ_R / λ_U ratio, the bandwidth gain is improved.

9. Evaluation Results

This section provides the evaluation result to show the performance of the proposed approach. The evaluation results are implemented using SQL/Server and C#.NET framework. A system including 95 users and dataset of 2000 different products are selected. There were 25 different categories. Users may select each product and can see a statistic and customers' interests for that particular product. Such information is updated periodically.

In this experiments the selections are took place randomly. It is obvious if a group of humans with intelligence select them, there is more likelihood to have more accurate results.

Number of users: 5
Requests of each user: 1000
Index length of each user: 150 records
Output records after each prediction: 20

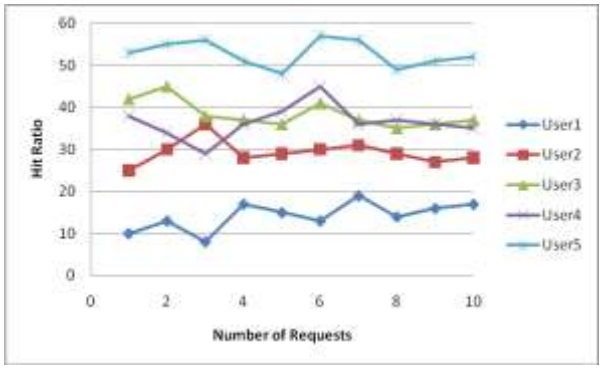


Figure 8. Hit Ratio for Each User

The first experiment is repeated with different number of requests and Index lengths.

Number of users: 5
Requests of each user: 2000
Index length of each user: 200 records
Output records after each prediction: 20

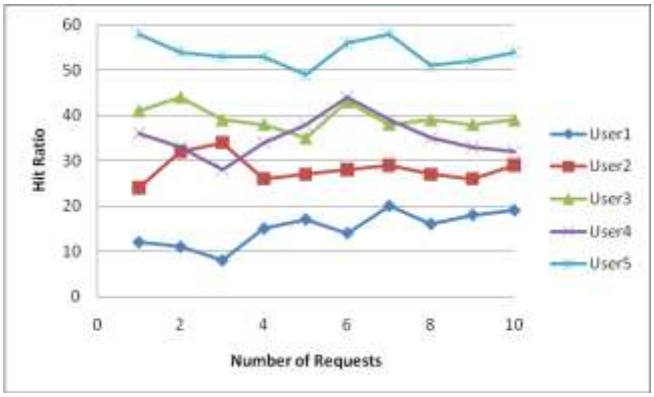


Figure 9. Hit Ratio for Each User

The results show higher performance than the similar work by Khoozani [17]. This is due to the usage of an intelligent machine such as learning automata.

In another experiment, we evaluate the effectiveness of using the rate adoptive algorithm. This module is implemented using .NET framework and installed on the proposed caching system. We have used the same parameter similar to Figure 7.

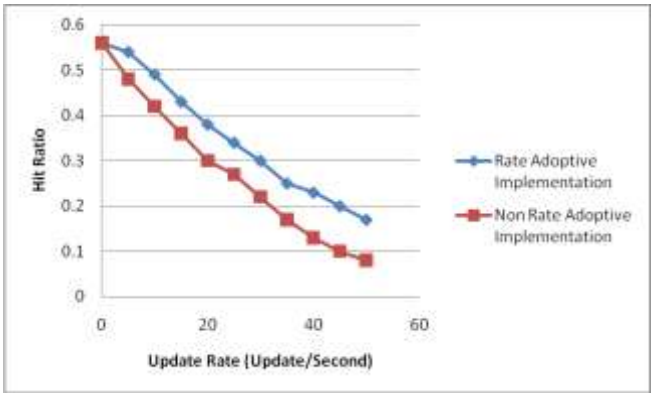


Figure 10. Hit Ratio Comparision for Rate Adoptive Imlementation

Evaluation results show higher performance and accuracy in comparison to Mehrshad work [11].

Another important factor is query request rate. We analyze our approach based on the rate of Query updates per minute.

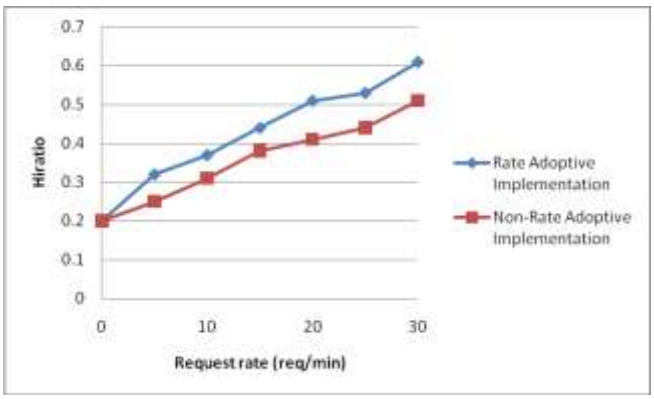


Figure 11. Hit Ratio Comparision for Rate Adoptive Imlementation

Again, evaluation results show comparable performance and accuracy in comparison to Mehrshad work [11].

In another experiment, we investigate the effect of different values for γ. Two situations are considered, One $\lambda_R / \lambda_U = 1$ and another $\lambda_U = 20\lambda_R$.

Simulation results are shown in Figure 12.

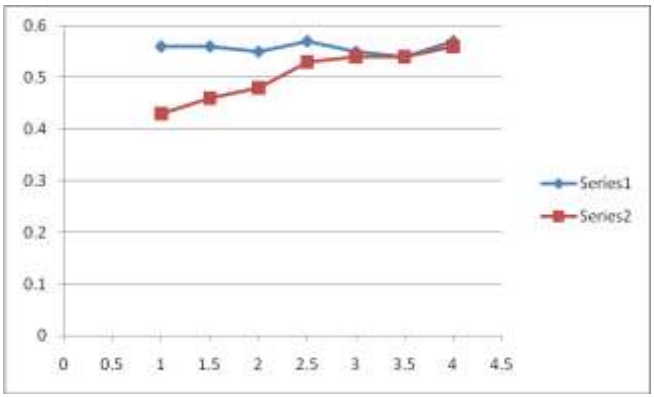


Figure 12. Hit Ratio for Different Update and Request Rates, Series1 Shows $\lambda_R / \lambda_U = 1$ and Series 2 Depicts $\lambda_U = 20\lambda_R$

This is obvious by increasing γ, less data are fetched from outside of the caching scheme and the hit ratio is increased. Moreover, more update rates results in less hit ratio due to massive updates and server update mismatches which result in a miss in the proposed caching scheme.

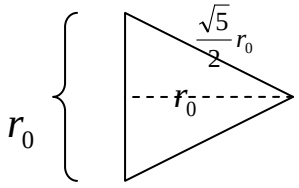
10. Conclusions

In this work, we introduced a new suggestion algorithm for transactions caching in distributed database systems. Moreover, we implemented a mechanism to adopt the rate of records updating at server and client. We employed an intelligent machine such as learning automata to improve the suggestion module of our caching scheme. The results show higher

performance and accuracy of our proposed system in comparison to other similar works. Moreover, our approach is more secure by using mobile supportive station. In this regard the burden of caching scheme is on such stations and no moving agent. Such scheme results in higher performance and security.

Appendix: Expected Number of HOPS between MSS

We assume a rectangular topology with area $a * a$ and uniform distribution of nodes. Two nodes can form a direct link if the distance X between them is less than or equal to r_0, where r_0 is the maximum node transmission capacity distance. However, this distance implies an straight line. The final value is $\sqrt{5}r_0$ which the length of indirect line using triangular formula is. We assume there is third node between such nodes which have at most the same distance such as r_0.



Assuming the topography filled with a sufficient number of nodes that are uniformly distributed, the expected distance to the corner, is calculated as [11],[21]:

$$E[X_{Corner}] = \int_0^a \int_0^a \frac{1}{a^2}\sqrt{x^2 + y^2}\, dxdy = 0.76 * a$$

Which declares the mean of all possible distance in the area. To obtain the number of hops. This value should be divided by $\sqrt{5}r_0$. Therefore, $\frac{0.76 * a}{\sqrt{5}r_0}$ is the expected number of hops.

References

[1] E. Pitoura and B. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environments", Proceeding in 15thInternational Conference on Distributing Computing Systems, **(1995)**.
[2] G. F. Forman and J. Zahorjan, "The Challenges of Mobile Computing", IEEE Computer, **(1994)** April.
[3] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing", Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, **(1996)**.
[4] N. Santos, L. Veiga and P. Ferreira, "Transaction Policies for Mobile Networks", Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE, **(2004)**.
[5] N. Prabhu, V. Kumar, I. Ray, and G.-C. Yang, "Concurrency Control in Mobile Database Systems", Proceedings of the 18th International Conference on Advanced Information Networking and Application, IEEE, **(2004)**.
[6] M. H. Dunham, A. Helal and S. Balakrishnan, "A mobile transaction model that captures both the data and movement behavior", Mobile Networks and Applications, vol. 2, **(1997)**, pp. 149–162.
[7] G. D. Walborn and P. K. Chrysanthis, "PRO-MOTION: Management of Mobile Transactions", Proceeding of the ACM Symposium on Applied Computing, **(1999)**.
[8] Y. Chung, B. Bhargava, M. Mahoui and L. Lilien, "Autonomous Transaction Processing Using Data Dependency in Mobile Environments", Department of Computer Sciences Purdue University, **(2002)**.
[9] E. Pitoura and B. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environments", Proceeding in 15thInternational Conference on Distributing Computing Systems, **(1995)**.
[10] H. Beigy and M. R. Meybodi, "Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path

Problem", International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, World Scientific Publishing Company, to appear.
[11] K. Mershad and H. Artail, "SSUM: Smart Server Update Mechanism for Maintaining Cache Consistency in Mobile Environments", IEEE Transactions on Mobile Computing, vol. 9, no. 6, **(2010)** January, pp. 778-795.
[12] S. Huang, B. Lin and Q. Deng, "Intelligent Cache Management for Mobile Data Warehouse Systems", In Proceedings of J. Database Manag., **(2005)**, pp. 46-65.
[13] C. Sapia, "PROMISE - Modeling and Predicting User Query Behavior in Online Analytical Processing Environments", FORWISS Technical Report FR-2000-001, **(2000)** June.
[14] J. Fong, H. K. Wong and S. M. Huang, "Continuous and incremental data mining association rules using frame metadata model", Knowledge-Based System, vol. 16, no. 2, **(2003)** March, pp. 91-100.
[15] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme and N. Sulieman, "COACS: A Cooperative and Adaptive Caching System forMANETS", IEEE Trans. Mobile Computing, vol. 7, no. 8, **(2008)** August, pp. 961- 977.
[16] K. Mershad and H. Artail, "CODISC: Collaborative and distributed semantic caching for maximizing cache effectiveness in wireless networks", Journal of Parallel and Distributed Computing, vol. 71, Issue 3, **(2011)** March, pp. 495-511.
[17] M. H. Khoozani, A. Tajvidi and M. E. Shiri, "Information prediction for caching in mobile transactions", Computer Engineering Conference, Lahijan, Iran, **(2007)**.
[18] M. R. Meybodi and H. Beigy, "Solving Stochastic Path Problem Using Distributed Learning Automata", Proceedings of The Sixth Annual International CSI Computer Conference, CSICC2001, Isfahan, Iran, **(2001)** February 20-22, pp. 70-86.
[19] M. Bazarganigilani and A. Syed, "Web Page Classification Using Distributed Learning Automata and Partitioning Graph Algorithm", dexa, 2010 Workshops on Database and Expert Systems Applications, **(2010)**, pp. 302-304.
[20] O. Bahat and A. Makowski, "Measuring Consistency in TTLBased Caches", Performance Evaluation, vol. 62, **(2005)**, pp. 439-455.
[21] C. Bettstetter and J. Eberspacher, "Hop Distances in Homogeneous Ad Hoc Networks", IEEE Proc. 57th IEEE Semiann. Vehicular Technology Conf., vol. 4, **(2003)** April, pp. 2286-2290.