Volume 3, Issue 10, October 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Evaluating the Query for a Mobile Database System through Dongle Transaction Model

A.Priya*
Research Scholar,
Research & Development Centre,
Bharathiar University, Coimbatore 641046, Tamil Nadu, India

Dr. R.Dhanapal
Director,
Research & Development,
Procademia, Chennai 600 024, Tamil Nadu, India.

Abstract*— In the rapid exploring technology the mobile devices are used for accessing information from anywhere and anytime. The query evaluation will be done by sending the continuous query to the server in mobile environment. The server will obtain a continuous changing result for those queries. Each and every query will produce an accurate and timely outcome from the server to the mobile user. The mobile devices are monitored at each stage of its outcomes. The queries can be sent the request through point-to-point channels or broadcast channels. The objective of this paper is to discuss the about the evaluation of the query for a mobile database system through the Dongle Transaction Model. The Mobile Transaction Testing and Query is measured and analysed for both approaches such as Kangaroo Model & Dongle Transaction Model to compare their retrieval access time with the no. of records. The database server is a MS SQL Server 2008 Query Analyser and the DBMS is tested with the MS Test Unit Testing framework in Visual Studio 2008.

Keywords* — Mobile Database, Dongle Transaction Model, Mobile Support Stations, Mobile Host, Dongle Data.

I. INTRODUCTION

In the recent advances technology the wireless communication networks are used in mobile database systems. The mobile database system provides the database applications environment to prove the mobile transactions. This advance technology helps the mobile environment to retrieve the information from the database server and sends it back to the mobile users [1]. The users always access the database through a wireless network. Basically, there is a server which holds all the information send by the mobile user called mobile unit. The user with a wireless connection to the information network will maintain a fixed position in the network. In mobile environment, all the clients are very dynamic and can be extremely volatile. The database representing information in one unit will be moved from client-to-server and server-to-client. All the mobility has distinguishing feature for the mobile computing system. Consequently, the answer to a Mobile unit will be sent by the mobile server independently. The query may depend on the location of the mobile host (MH) or the Mobile Unit (MU) that was issued at the time of representation of the particular database.

Therefore, to evaluate the query MS SQL Server 2008 Query Analyser and MS Test Unit Testing framework in Visual Studio 2008 is required. Each and every query sent by the client (Mobile Host) will be processed by the Mobile Server and send it back to the client. The information will be retrieved by the number of records to which is accessed at the particular time. The Mobile Transaction Models such as Kangaroo Model and Dongle Transaction Model has been considered for the implementation.

II. MOBILE COMPUTING ENVIRONMENT ON MOBILE DATABASE

Mobile computers have become increasingly prevalent as professionals discover the benefits of having their electronic work available at all times [2]. Developing distributed applications that make effective use of networked resources from a mobile platform, however, is difficult for four reasons.

- First, mobile computers do not have a permanent connection into the network and are often disconnected for long periods of time.
- Second, when the computer is connected, the connection often has low bandwidth and high latency and is prone to sudden failure, such as when a physical obstruction blocks the signal from a cellular modem.
- Third, since the computer may be forced to use different transmission channels depending on its physical location, the performance of its network connection can vary dramatically from one session to another.

- Finally, depending on the nature of the transmission channel, the computer might be assigned a different network address each time that it connects.

In short, any distributed application that works on a mobile platform must deal with unforgiving network conditions. With the rapid development of internet and wireless network technology the computer-centered mobile computing technology has been widely used and developed, so that people can access to any required information at anytime and anywhere. In the mobile computing environment, each mobile computing device has its own database system [3]. All the database systems in the mobile computing environment created a Mobile Database Cluster (called "MDBC" for short).

The traditional distributed database technology has been difficult to support mobile computing, because of the mobility, frequent disconnection, network diversity, non-symmetry of network communications in mobile computing environment, and the mobility, self-command, distribution, heterogeneity and other features of the mobile database in the MDBC [4]. In order to effectively access the database in mobile computing environment, mobile database technology came into being. Mobile database technology as an extension of the distributed database technology has many strengths of distributed database and unique features. Mobile database system not only can manage datum and information of the local terminals, but also can connect the central database to process all kinds of transactions, so as to meet the needs of information processing of different industries and users in the mobile environment and greatly improve the timeliness of information. In mobile database, the transactions launched by the mobile computer go by the name of mobile transaction. Mobile transaction usually belongs to long-lived transaction because of limited communication bandwidth and frequent disconnection. In the processing of mobile transactions, the location change of mobile computers will bring about complex switchover problem of region. In addition, the processing of mobile transaction is more prone to error. Mobile transaction processing has become a challenging study field because of the above characteristics of mobile transaction. In order to improve the processing efficiency of mobile database system, a new mobile database model based on agent was designed. Mobile database system not only can manage datum and information of the local terminals, but also can connect the central database to process all kinds of transactions, so as to meet the needs of information processing of different industries and users in the mobile environment and greatly improve the timeliness of information.

In mobile database, the transactions launched by the mobile computer go by the name of mobile transaction. Mobile transaction usually belongs to long-lived transaction because of limited communication bandwidth and frequent disconnection. In the processing of mobile transactions, the location change of mobile computers will bring about complex switchover problem of region [5]. A mobile database can be recognized as a kind of distributed database that supports mobile computing. It is important to allow mobile hosts get hold of data that may be distributed across multiple mobile databases running on neighbouring mobile devices. The schemas of these highly decentralized databases may therefore be different but could overlap, in that it may possess common attributes and even common relations. When trying to view the collection of such databases found in a wireless network as one logical database, it is suitable and natural to consider it as a federated database. The mobile database system in a MANET is a dynamic distributed database system which is composed of some Mobile Hosts (MHs). The MHs in a MANET can be classified by their capabilities to SMH and LMH.

A Client or Small Mobile Host (SMH) is a node with reduced processing, storage, communication, and power resources. A Server or Large Mobile Host (LMH) is a node having a larger share of resources. The database technology allows employees using handheld to link to their corporate networks, download data, work offline, and then connect to the network again to synchronize with the corporate database [6]. For example, with a mobile database embedded in a handheld device, a package delivery worker can collect signatures after each delivery and send the information to a corporate database at day's end.

The present database systems do not provide special facilities for specific update operations in a mobile computing environment. Some of the commercially available Mobile relational Database systems are:

- IBM's DB2 Everywhere 1.0
- Oracle Lite · Sybase's SQL: These databases work on Palm top and hand held devices (Windows CE devices) providing a local data store for the relational data acquired from enterprise SQL databases.

The main constraints for such databases are relating to the size of the program as the handheld devices have RAM oriented constraints. The commercially available mobile database systems allow wide variety of platforms and data sources. It also allows users with handheld to synchronize with Open Database Connectivity (ODBC) database content, and personal information management data and email from Lotus Development's Notes or Microsoft's Exchange. These database technologies support either query-by-example (QBE) or SQL statements.

III. DONGLE TRANSACTION MODEL ARCHITECTURE USING DONGLE AGENT

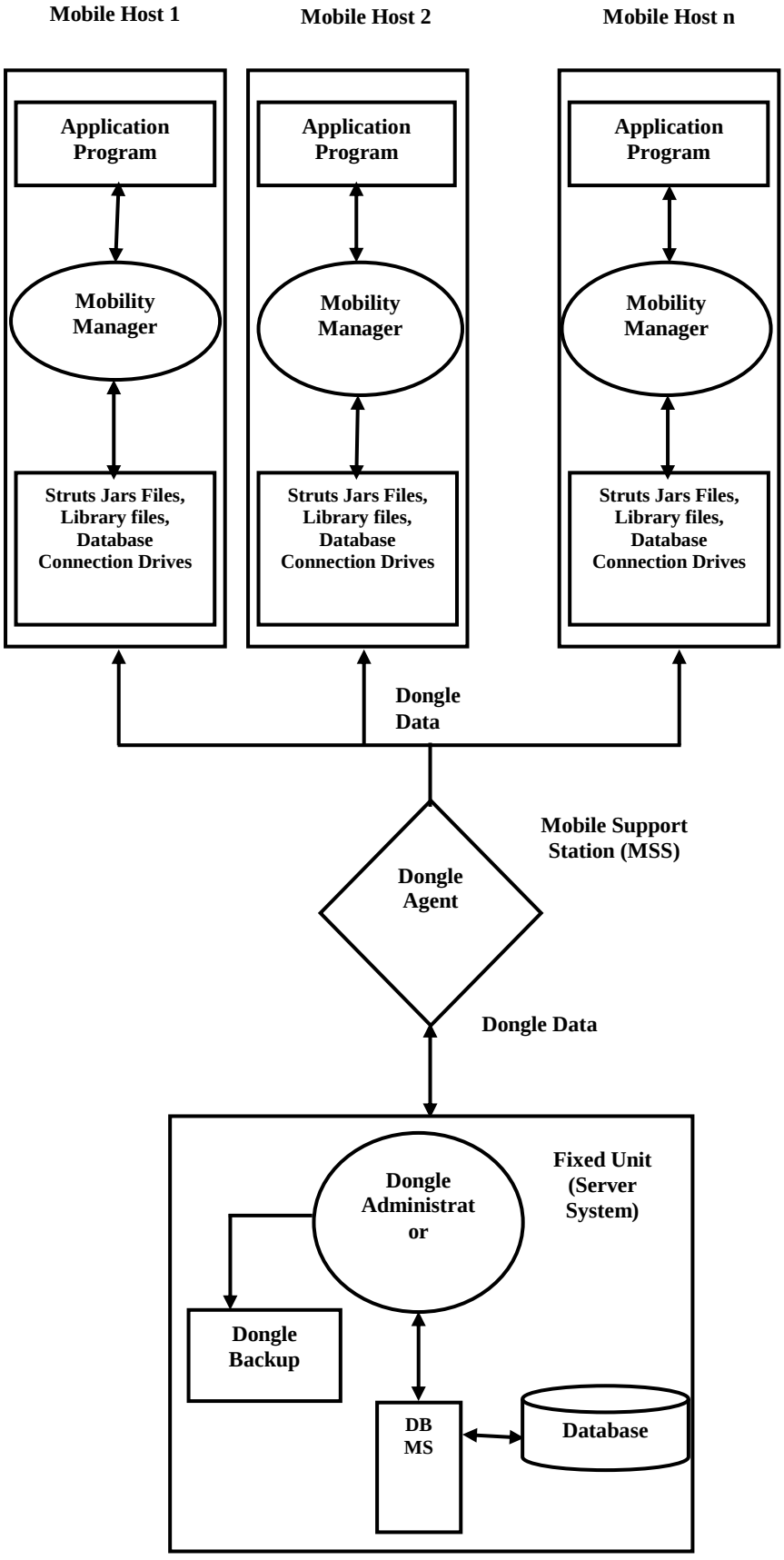

Fig. 1 Dongle Transaction Model Architecture using Dongle Agent

The Dongle Transaction Model was developed by the tools of Front-End and Back-End. This model uses an Apache Struts Web Framework as Front-End Tool and My SQL Enterprise Backup as Back-End Tool for the development of the retrieval of Information and accessing the data.

The Dongle Transaction Model Architecture consists of three parts namely,

(i) Dongle Administrator,
(ii) Dongle Agent &
(iii) Mobility Manager.

In Figure 1, the Dongle Administrator is available on the Server System to hold all the Dongle Agents details and Mobility Manager's Transactions. The Dongle Back-up is placed adjacent to the Dongle Administrator, which has the capable of taking all the back-up's for the Server System. The Database will have the Dongle Agent's connectivity details and also the access rights provided to each Dongle Agent. The DBMS can perform the transaction for the Dongle Data that is given by the Mobility Manager through the Dongle Agent.

This system represents an agreement between the Fixed Unit (Server System) and the Mobile Host where the database server delegates control of some data to the Mobile Host to be used for local transaction processing [7]. The Fixed Unit need not to be aware of the operations executed by individual transactions on the Mobile Host, but, rather it sees the periodic updates done by the Mobility Manager through a *Dongle Data* for each of the data items manipulated by the mobile transactions. The management of *Dongle Data* is performed by the *Dongle manager* on to the Fixed Unit and the *Dongle Data* on each Mobile Host cooperatively. *Dongle Data* are obtained from the database by requesting when a data demand is created by the MH. A Mobile Agent called *Dongle Agent*, a stationary server front-end called *Dongle Administrator*, and an intermediate array of *Mobility Manager* helps to manage the flow of updates and data between the other components of the system.

The *Dongle Data* is then recorded to the *Dongle Backup* and transmitted to the Mobile Host to provide the data and methods to satisfy the needs of transactions executing on the MH. For the Fixed Unit two or more Mobile Unit can be connected to perform the Mobile transactions. The Server System will always be considered as the Root Node. Each Dongle Data will be processed through the Struts-Config.Xml Files and to perform updates submitted by transactions run by applications executing on the Mobile Host. If the *Dongle Manager* is the only means to access the database, every item in the database can be considered implicitly locked by the root transaction [8]. When an item is needed by a MH, the *Dongle Manager* can read the data value and immediately release any actual (i.e. server imposed) locks on the data item, knowing that it will not be accessed by any transaction unknown to the *Dongle Manager*.

IV. EVALUATING THE QUERY ON THE DONGLE TRANSACTION MODEL

Let us consider transactions T1 and T2, where T1 starts initially and T2 starts after committing but before the termination of T1. To increase the concurrency, the system allows T2 to start before the termination of T1. Once a transaction is committed, then it means that the changes made by the particular transaction are updated in the Database Server (DBS) and this updated value is provided to all the other systems on request of the value [9]. When an agent requests for the data, one of these instances, is provided, depending on the type of transaction that requests for the data. It represents a version of data item as, where '*X*' is the data item and *i* takes the value 1 to n (n depends on number of transactions applied on T_i); '*ts(i)*' is the timestamp of the mobile transaction T_i that has written the version of the data item. '*ts(i)*' stands for the present timestamp of data item version *used* in version selection to process a read operation on '*X*'.

It also implies that the transaction T_i, which has updated the data item, has been successfully committed at mobile agent but is yet to be terminated at DBS. DBS assigns timestamps to the data items when a transaction accesses them, and is assumed to be synchronized across the system. The data version written by the mobile transaction T_j, where *j* takes the value 1 to n (n depends on number of transactions applied on T_j) which has been terminated successfully at DBS. The subscript 'zero' indicates the successful termination of T_j at DBS [10]. As said earlier, in the new version of data item created by the committed mobile transaction T_k at Mobile agent that has created the new value at time *ts(k)*, but is not yet terminated ($T_j < T_k$ in timestamp) at DBS, where *k* takes the value 1 to n (n depends on the number of transactions applied on T_k).

Case 1: To increase the availability of any system, concurrency should be provided in both read and write areas of the system.

Consider a DBS with one version of data item '*X*' and two versions of data item '*Z*'. Versions represent the transaction T_i which updated *X* and *Z* has been most recently terminated *i* takes the value 1 to n (n depends on number of transactions applied on T_i). The version indicates that there is a transaction T_k that is committed but is yet to be terminated, where *k* takes the value 1 to n (n depends on the number of transactions applied on T_k).

The data item versions present at mobile agent indicate that earlier the transaction T_j being executed at Mobile agent has requested a read operation on data item '*Z*' and a write operation on data item '*X*', where *j* takes the value 1 to n (n depends on number of transactions applied on T_j) which has been terminated successfully at DBS. DBS assigned the most up-to-date version of data items to the read operation. Hence, T_j read data item versions (committed version) and (terminated version) available at DBS. After obtaining write permission on data item '*X*', it writes its version. Version at DBS indicates that there existed a transaction T_k that has committed at Mobile agent but not yet terminated. In our model, in order to maintain exactly two versions of a data item, it can't be assigned to write permission on a data item to any

transaction if there is another transaction that holds the write permission on that data item. The concurrent write operations on a data item conflict. Also, a transaction abort after acquiring write permission might result in cascading aborts [11]. It is also possible that both transactions can simultaneously commit at Mobile agent resulting in the existence of more than one committed version of data item. Therefore, Mobile agent cannot obtain a write permission and write its version of '*Z*' but can only read the version. To prove the above property is true, another constraint defined by Case 2 is included.

Case 2: For instance, there exists only one committed transaction in a DBS.

There exist two versions of data items '*X*' and '*Z*' at DBS. The data items and represent that there exist a committed but yet to be terminated transaction T_k. The data items and represent a transaction T_i that is most recently terminated at DBS. The transaction T_j being executed on Mobile agent reads the version at Mobile agent. DBS assigns the write permission to transaction T_j at Mobile agent on data item '*X*' even though there exists a committed but yet to be terminated transaction T_k at DBS that has a committed version [12].

This is possible because the transaction T_k is committed and therefore assured of successful termination. The versions of a data item written by such transaction are up-to-date and can be used by another transaction. Since transactions can be terminated only at DBS, it can keep track of the order in which transactions need to be terminated. Thus, DBS can gives write permission on a data item when there exists another transaction that is committed on the same data item but yet to be terminated. Hence, T_j at Mobile agent is assigned the write permission on data item '*X*' with version. It then writes its own version of data item. The transaction T_j then commits at Mobile agent and sends the version to DBS to terminate.

Case 3: Whenever a transaction wants to read a data item then the committed version of the data item with the largest timestamp less than or equal to the timestamp of the transaction is selected.

That is, if there exist versions: T2 and T3 at DBS and transaction *T*3 at Mobile agent or at DBS requesting data item *X* having a timestamp *ts(T3) > ts(T2)*, then *T*3 is given for execution, otherwise T2 is executed. A read protocol should adhere to the following: T_i requests a read lock on the data item *X*, DBS grants or corresponding to whether the version T2 or version T3 is selected in accordance with the read rule; and the read lock version satisfies the specified constraints [13]. Transaction T_i reads the selected version of *X*. The write protocol should adhere to the following: T_i requests a write lock on data item *X*, DBS grants, if there are no conflicts then, T_i creates a new version for data item *X*. Rules to be satisfied for acquiring a lock:

- A read lock for an instance it can be acquired by a system, (even if its write lock is held by another process j) only if the timestamp of i is greater than the timestamp of j.
- A write lock can be obtained only if no system contains any write locks to the currently requesting process.

Adhering to these rules helps maintain serialize ability of the system, hence avoiding cascading aborts.

V. EXPERIMENTS AND RESULTS

The primary goal of the present research work is to empower the Transaction of a Mobile Database using the Mobile Agent. The Dongle is implements in each and every client to access those data that has been stored on the server's database. This will lead to a faster transaction of the data using the mobile network [14]. The graphical representations explain about the comparison of the executable features of the two models such as the Dongle Transaction Model's & Kangaroo Model are categorized.

A. SYSTEM TESTING

The Mobile Transaction Testing and Query is measured and analysed for both approaches such as Kangaroo Model & Dongle Transaction Model to compare their retrieval access time with the no. of records. The database server is a MS SQL Server 2008 Query Analyser and the DBMS is tested with the MS Test Unit Testing framework in Visual Studio 2008. Four computers were configured as Mobile Support Stations sites and one as mobile server.

B. EXPERIMENTAL RESULTS

The vertical axis shows the transaction's access time (in milliseconds) for both the models such as "Dongle Transaction Model" and "Kangaroo Transaction" [15]. The horizontal axis represents the number of records to be executed at the time of I/O performance. All the Data in X axis and Y axis are represented in unit of hundreds. The x axis represents the number of records on which the transaction was carried out, and the y axis represents the number of access made to the database server for performing the transactions.

The following Table 1 and Figure 2 explain about the I/O Performance with Sequential Retrievals for No. of Records for two Transaction Models such as Dongle Transaction Model & Kangaroo Model. The Dongle Transaction Model was compared with the Kangaroo Model and the results were obtained. This shows that Dongle Transaction Model shows high Performance when compared to the Kangaroo Model.

TABLE 1: I/O PERFORMANCE WITH SEQUENTIAL RETRIEVALS

No of Records	Dongle Transaction Model	Kangaroo Model
5	0.3	0.6
10	1.2	1.64
15	2	2.44
20	2.67	2.96
25	3.01	3.69
30	3.76	4.11
35	4.22	4.98
40	5.12	5.99

The I/O Performance with Sequential Retrievals shows that the total Access Time taken for the retrievals of the record.

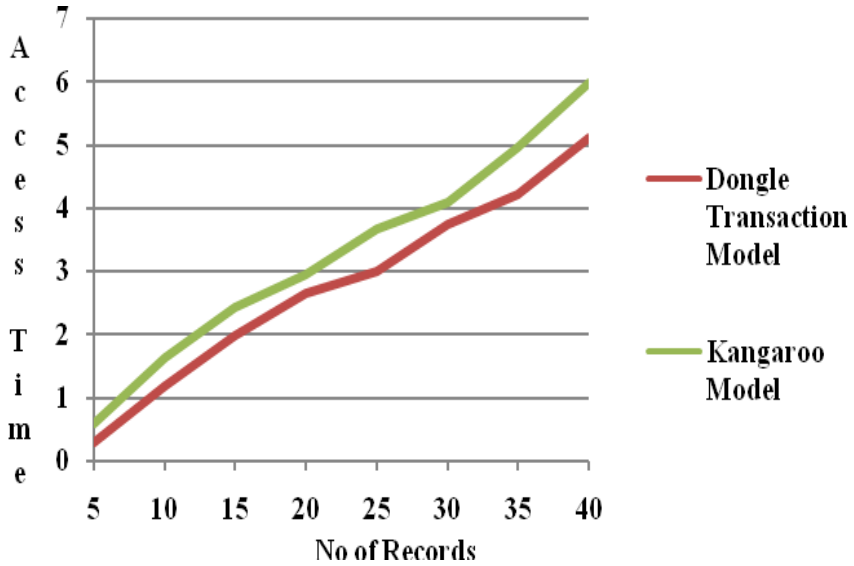

Fig. 2 I/O Performances with Sequential Retrievals

The No. of Records Retrieved by the Dongle Transaction Model is always very fast and more than comparing with the Kangaroo Model. The I/O Performance with Random Retrievals shows that till reaching a particular number of transactions of records, the Dongle Transaction Model and the Kangaroo Model performs almost takes the same access time, after which the access time stabilize in the Dongle Transaction Model, while the Kangaroo model shows periodic increase.

TABLE 2
I/O PERFORMANCE WITH RANDOM RETRIEVALS

No of Records	Dongle Transaction Model	Kangaroo Model
5	0.7	0.7
10	1.4	1.5
15	2.6	2.7
20	2.8	3.16
25	3.01	3.44
30	3.419	3.89
35	3.724	4.13
40	3.94	4.93

This shows that the I/O Performance with Random Retrievals in Dongle Transaction Model is faster than the Kangaroo Model as shown in Table 2 & Figure 3.

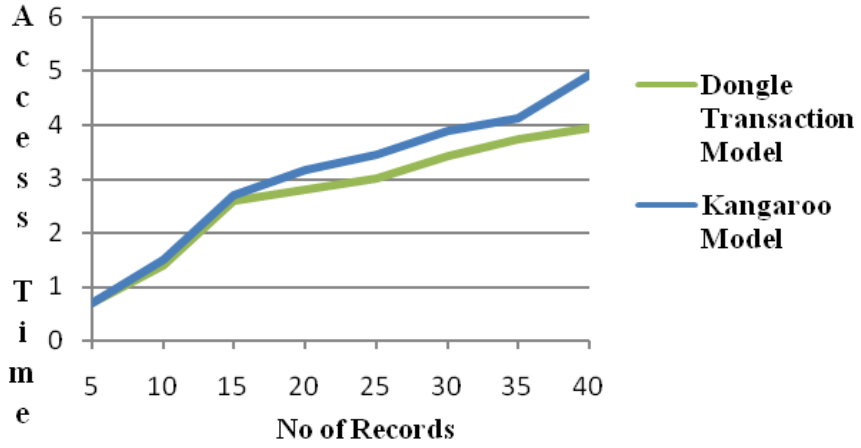

Fig. 3 I/O Performances with Random Retrievals

The I/O Performance with Insertion, Deletion and Modification Transactions shows that No. of Records used for Transactions in the Dongle Transaction Model is faster than the Kangaroo Model. The Table 3 & Figure 4 shows the I/O Performance with Insertion Transactions.

TABLE 3
I/O PERFORMANCE WITH INSERTION TRANSACTIONS

No. of Records	Dongle Transaction Model	Kangaroo Model
5	0.2	0.5
10	0.8	1.22
15	1.34	1.78
20	1.56	1.9
25	1.98	2.31
30	2.33	2.56
35	2.67	2.89
40	3.16	3.37

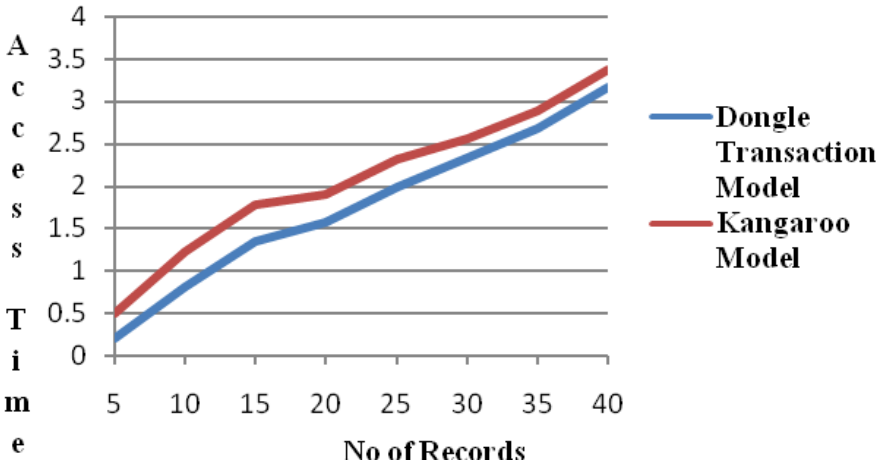

Fig. 4 I/O Performances with Insertion Transactions

The Table 4 & Figure 5 shows the I/O Performance with Deletion Transactions. In which the Dongle Transaction Model is faster than the Kangaroo Model to delete no. of transactions done for the no. of records.

TABLE 4
I/O PERFORMANCE WITH DELETION TRANSACTIONS

No of Records	Dongle Transaction Model	Kangaroo Model
5	0.8	1.5
10	1.4	2.4
15	2.3	3.35
20	2.9	4.6
25	3.7	5.35
30	4.9	6.55
35	5.2	6.8
40	5.3	7.01

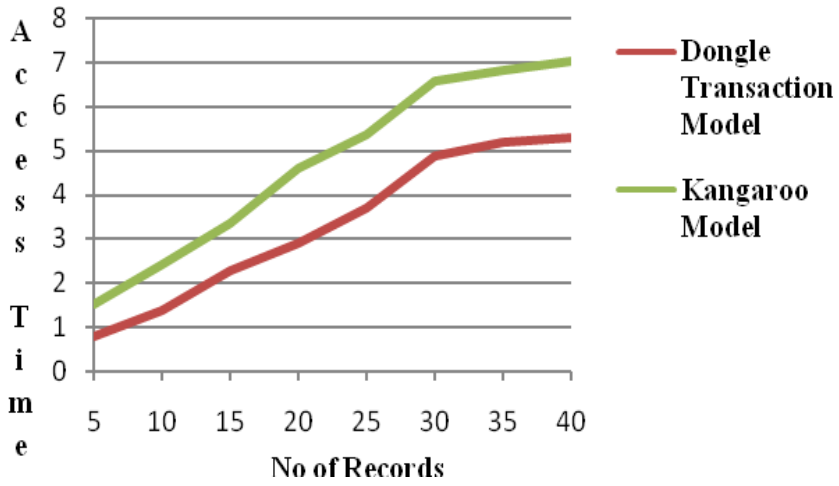

Fig. 5 I/O Performances with Deletion Transaction

The Table 5 & Figure 6 shows the I/O Performance with Modify Transactions. In which the Dongle Transaction Model is faster than the Kangaroo Model to modify the no. of records in no. of transactions.

TABLE 5
I/O PERFORMANCE WITH MODIFY TRANSACTIONS

No of Records	Dongle Transaction Model	Kangaroo Model
5	1.01	1.7
10	1.7	2.59
15	2.3	3.35
20	3.14	4.6
25	4.33	5.35
30	4.71	6.55
35	5.02	7.23
40	5.22	7.982

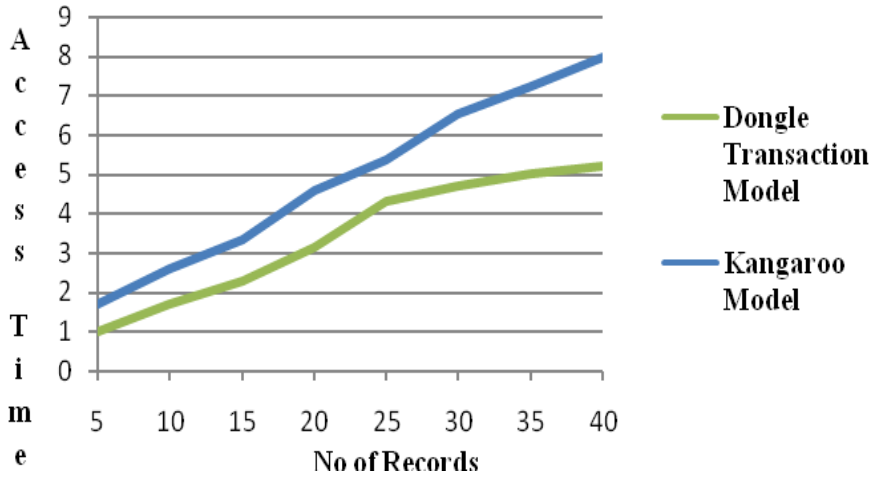

Fig. 6 I/O Performance with Modify Transactions

VI. CONCLUSION AND FUTURE WORK

The Dongle Architecture uses the Fixed Unit as the Server System and Mobile Unit called Mobile Host as the Client System. There is a Mobile Support Station which acts as the intermediate for Client and the Server Systems to transfer the data. The Mobile Client/Mobile Database and the Back-End Database Server communicate with each other through a wireless network. The Dongle Transaction Model is implemented in each and every client to access those data that has been stored on the server's database, which leads faster transaction for mobile network. The graphical representation shows the clear comparison features of the two models such as the Dongle Transaction Model & Kangaroo Model. The Retrieval Access Time and the No. of Transaction Records are tested with the four computers were configured as Mobile Support Stations sites and one as mobile server. The analysis report shows that the Dongle Transaction Data Model can be implemented in the Cellular Systems for the next generation.

REFERENCES

[1] A.Priya, Dr.R.Dhanapal, S.Dinesh, "A Method of Implementing Dongle Transaction Model in Mobile Transaction Systems using Mobile Agents", European Journal of Scientific Research, ISSN 1450-216X Vol. 90 No 4 November, 2012, pp.536-549.

[2] A.Sripriya, Dr.R.Dhanapal, T.P.Vijayalakshmi , "Mobile Database Transaction Using Mobile Agents", 2010 Journal of Computing Press, USA, Volume 2, Issue 12, December 2010, ISSN 2151-9617, pp. 88-93.

[3] Jing Li, Jianhua Wang, "A New Architecture Model of Mobile Database Based on Agent", IEEE Computer Society – 2009.

[4] J. Holliday, D. Agrawal, and A. El Abbadi, "Disconnection Modes for Mobile Databases," J. Wireless Netw., pp. 391–402, 2002.

[5] Shibnath Mukherjee, "A Modified Kangaroo Model for Long Lived Transactions over Mobile Networks", First International Workshop on Database Technology and Applications - 2009.

[6] D. Saha and N. Chowdhury, "A Method for Secure Query Processing in Mobile Databases", 2007, Advance online publication: Engineering Letters, 14:1, EL_14_1_20.

[7] Sanjay Kumar Madria, Mukesh Mohania and John F. Roddick, "A Query Processing Model for Mobile Computing using Concept Hierarchies and Summary Databases", 1999.

[8] James Winly Jayaputera, "Mobile Query Processing Incorporating Server and Client Based Approaches", A Thesis Submitted for Clayton School of Information Technology, Monash University, September, 2008.

[9] Saeid Maleki, Mohammad Ebrahim Shiri & Samaneh Hekmatmehr, "The Management of Mobile Transaction with the use of Data's Feedback", International Conference on Information and Multimedia Technology - 2009.

[10] R. Malladi and Karen C. Davis. "Applying Multiple Query Optimization in Mobile Databases." Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), IEEE Computer Society, pp 294-303.

[11] Samidha Dwivedi Sharma, D.r R.S.Kasana, "Mobile Database System: Role of Mobility on the Query Processing", International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010.

[12] Agustinus Borgy Waluyo, Bala Srinivasan, David Taniar, "Mobile Query and Processing in Mobile Database Environment", 2002.

[13] Ziyad.T.Abdul-Mehdi & Ramlan Mahmod, "Security Management Model for Mobile Databases Transaction Management" 2007.

[14] Serrano-Alvardo, P., C. Roncancio and M. Adiba, 2004, "A Survey of Mobile Transactions", Kluwer Academic Publishers Distributed and Parallel Databases (DAPD), 16, pp 1-38.

[15] Zulfu orenc, "A Study of Kangaroo Transaction Model for Mobile Transaction Management", A thesis submitted to the graduate school of informatics - 2004.

