AMDB – An Approach for Sharing Mobile Databases

José de Aguiar Moraes Filho **Angelo Brayner**

University of Fortaleza (UNIFOR) – Fortaleza – CE – Brazil

{jaguiar.mia, brayner}@unifor.br

Abstract. *We define a Mobile Database Community (MDBC) as a dynamic collection of autonomous mobile databases in which each database user can access databases in the community through a wireless communication infrastructure. New participants may join to an MDBC as they move within communication range of one or more hosts which are members of the MDBC. MDBC participants may also transiently disconnect from the network due to communication disruptions or to save power. This paper describes an agent-based architecture, called AMDB (Accessing Mobile Databases), which enables such communities to be formed opportunistically over mobile database hosts in ad hoc configurable environments. The AMDB architecture is fully distributed and exploits physical mobility of hosts and logical mobility of database queries and their results across mobile hosts.*

1. Introduction

Advances in portable information devices and wireless network technology have led to the development of a new computing paradigm, namely, mobile computing. According to this paradigm, users carrying portable devices are able to access services provided through a wireless communication infrastructure, regardless of their physical location or movement patterns. The database technology has also been impacted by the mobile computing paradigm. For example, in an environment with support for mobile computing, a varying number of mobile computers can be interconnected through a wireless communication infrastructure, on each of which resides a database system. In other words, a dynamic collection of autonomous mobile databases, interconnected through a wireless communication infrastructure, can be formed in such an environment. For now on, that collection of mobile databases will be denoted Mobile Database Community (MDBC). Databases in an MDBC are mobile, autonomous and distributed. Moreover, they can be heterogeneous. It is important to note that in an MDBC, the notion of any-where and any-time data access plays a key role.

We propose an architecture for sharing heterogeneous, autonomous, distributed and mobile databases in an ad hoc and dynamically configurable environment [7]. The architecture, denoted AMDB (Accessing Mobile Database), is based on the concept of mobile agents [6]. It is fully distributed and exploits physical mobility of hosts and logical mobility of database queries (or transactions) and their results across mobile hosts. Moreover, it provides the support to form MDBCs opportunistically over mobile database hosts and enables sharing of mobile databases. Furthermore, query processing and transaction processing in an MDBC are addressed. Regarding the query processing issue, it is proposed a query mechanism to perform queries on databases in an MDBC. The proposed query processing is dynamic and adaptive. For the transaction processing issue, a transaction model for mobile transaction and a commit protocol are proposed.

This paper is structured as follows. In section 2, the proposed architecture is described and analyzed. Section 3 describes the proposed query processing mechanism for the AMDB architecture. Section 4 describes the proposed transaction mechanism. Section 5 concludes the paper.

2. The AMBD Architecture

The AMDB architecture [3, 8] is based on the concept of mobile software agents. Software agent is a software artifact that executes users' tasks in autonomous way [6]. The proposed architecture has two classes of agents: **stationary agents** and **mobile agents**.

The **stationary-agent** class has two types of agents: manager agents and wrapper agents. *Manager agents* are responsible for managing the local computational resources of a mobile computer through continuously monitoring of computational resources (e.g., used memory, cpu time, etc.) at mobile unit. These resources can be used to perform mobile agents' tasks. *Wrapper agents* provide an interface between mobile unit users and the AMDB platform. For that reason, wrapper agents have the following functionalities. Wrapper agents provide a common data representation of the local data stored at mobile computers. XML [11] is used as a common data representation. In fact, at each mobile unit exists an XML schema (called MDBC Schema), which contains data descriptions about all mobile database participants of a given MDBC. Additionally, a wrapper agent running on a given mobile unit is also responsible for temporarily transferring one or more services to another mobile computer of the community in order to improve performance or whenever a local critical situation occurs (for example, temporary service unavailability due insufficient local memory for processing a query). In this case, any mobile agent visiting the mobile unit of the wrapper agent should be redirected to the unit where the service is provided.

Mobile agents of the **mobile-agent** class are responsible for implementing the logical mobility property in the AMDB architecture. The basic feature of mobile agents is to carry database access code and access results while migrating between MDBC members. There are three types of mobile agents: Runner, Loader, and Broker agents. *Runner agents* are directly responsible for performing tasks required by mobile unit users in remote mobile databases belonging to an MDBC. Such tasks can represent data queries or data updates. *Loader agents* have the functionality of carrying a database query result back to the mobile unit that has required the query. *Broker agents* should gather the schemas of mobile databases of an MDBC, when a mobile unit joins an MDBC or when it is explicitly requested (schema evolution). Moreover, a broker agent can define a mobile unit as a temporary storage device for partial query results.

Figure 1 depicts an abstract model of the AMDB architecture. In order to describe how the proposed architecture works, suppose that a mobile unit, $MU_{M,}$ wants to form an MDBC. For that, it declares itself as initial coordinator of the new community to its wrapper agent. After that, the wrapper agent of MU_M sends information about the new community to all computers inside the area covered by the wireless communication network in which MU_M is connected. When news members begin to join the new MDBC, the central coordinator role is not necessary anymore. This coordinator function will be performed local and collaboratively by the wrapper agents at each mobile unit which is member of the community.

Now, suppose that a mobile unit MU_L wants to join to the MDBC, which was initially created by MU_M. First of all, it is necessary an explicit declaration of MU_L in order to see the schemas of all databases which are member of the community. This functionality is executed by the local wrapper agent, which, in turn, creates a broker (mobile) agent and gives to it the task of querying local database schema at all mobile units, which are hosts of database systems. The broker agent will roam unit by unit and will collect the local schemas. Local database schemas will be provided by local wrapper agents. When the broker agent of MU_L returns to MU_L, it brings the schemas of the participating databases and passes them away to wrapper agent of MU_L. With the database schemas of each database of the MDBC, users or application programs at MU_L are able to access data stored at those databases through an interface provided by local wrapper agents. Therefore, users or application programs submit queries to the local wrapper agent. In order to integrate databases belonging to an MDBC, the multidatabase system (MDBS) approach is used. Thus, queries over them should be specified in an extension to the XQuery [12], called MXQuery [9]. In fact, MXQuery is a multidatabase language (MDL) to integrate heterogeneous databases, where XML is the common model for representing conceptual schemas of databases in an MDBC. Therefore, the key goal of using MXQuery is to allow queries over several databases in an integrated way without attempting to integrate them by means of a global schema.

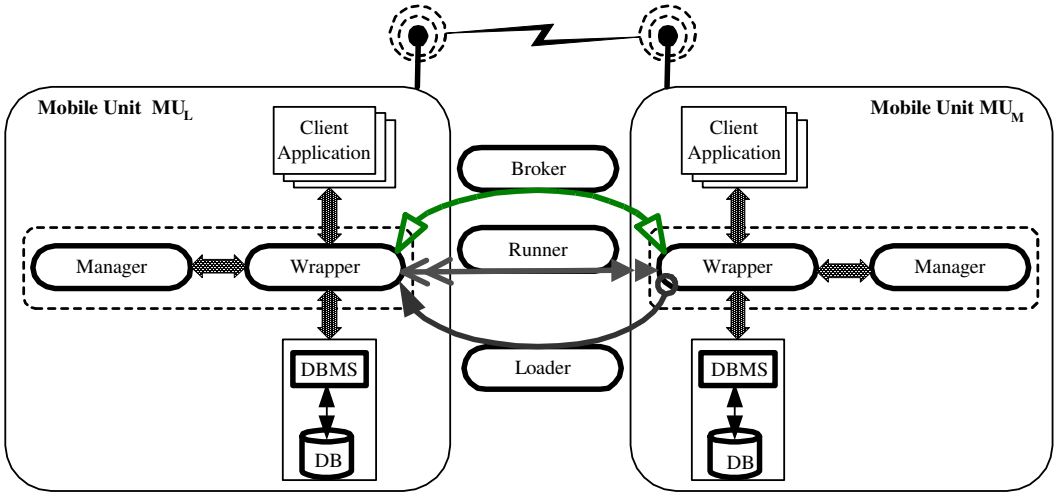

Figure 1. Abstract Model of the AMDB Architecture

3. Query Processing

A query in the AMDB platform is processed in six distinct phases. When the wrapper agent of MU_L (Figure 1) receives a query statement, it creates a runner agent at MU_L and passes to it the MXQuery expression corresponding to submitted query. After that, the runner agent begins to process the query. We call it **Pre-Processing** phase. In this phase, a query execution plan (QEP) is generated and represented by an operator tree. Tree nodes represent XML formal semantics algebraic operators [12] or data sources. The root represents the final query result.

During the execution of the **Decomposition** phase, a query is divided into fragments. Each fragment can represent recursively a sub-tree whose locality attribute refers to a mobile unit, in which the query fragment has to be executed. After that, the **Context Recognition** phase is processed. During this phase, runner agent's query

processor verifies, for each fragment, whether or not the local DBMS has resources to execute the fragment. In this phase is applied a cost model.

During the execution of the **Clone Generation** phase, the runner agent may create clones. A clone for each query fragment is created with the functionality of executing the fragment. The **Fragment Execution** phase is started when a runner agent (or its clones) migrates to the mobile units in which query fragments of a global query should be executed. On arriving at a given mobile unit, the runner agent or a clone submits to the local wrapper agent the query fragment. The wrapper agent, in turn, translates the fragment into the native query language of the local DBMS. After that, it submits the query to the local DBMS and returns the results back to the runner or clone.

During the execution of the **Post-Processing** phase, the clones send to the runner agent the results of the query fragments. The runner agent, in turn, starts the execution of operations which could not be executed by any local DBMS. In this case, the runner agent performs operations corresponding to the high-level nodes of the QEP. Our strategy uses adaptive operators such as XJoin [10] and Ripple Join [5] to perform global operations. Once the execution has finished, the runner agent projects the result to the user. Statistic information is also gathered during the execution of this phase.

4. Transaction Processing

The proposed transaction model is based on the use of semantic knowledge to relax the notion of absolute transaction atomicity. Supported by this new concept of atomicity, we propose a new correctness criterion, denoted Semantic Serializability, for the execution of concurrent transactions in MDBCs. The Semantic Serializability model was initially proposed in [4] and was extended for MDBCs in [2]. The key idea of the approach is to "see" an MDBC as a collection of disjoint sets of objects, each of which representing a single mobile database. We call those disjoint sets of objects *semantic units (SU)*. Operations over objects of a given *SU* represent an atomic unit in a mobile transaction. We have shown in [2] that the proposed transaction model provides a high degree of inter-transaction concurrency of mobile transactions and ensures consistency of the local databases belonging to an MDBC. We refer the reader to [2] for an in-depth discussion about transaction management in mobile databases and in the AMDB architecture.

5. Conclusion

In this work, we define the concept of dynamically configurable database communities in MANET environments. According to that concept, a dynamic collection of autonomous mobile databases, interconnected through a wireless communication infrastructure, can be opportunistically formed. In order to provide a platform to enable such collections, called MDBCs, to be formed opportunistically in ad hoc configurable environments, an architecture is described. The proposed architecture, denoted AMDB, supports physical mobility of hosts and logical mobility of database queries (or transactions) and their results across mobile hosts. Moreover, the proposed architecture has important additional properties for coping with database mobility, such as: *(i)* it does not require changes to the core of the underlying database systems, and; *(ii)* it supports opportunistic creation of database communities over a varying number of mobile and autonomous database hosts. A query engine for processing queries in AMDB architecture is described as well. The proposed engine implements a query

optimizer which has the ability for adapting a query execution plan dynamically according to the execution scenario. Finally, we propose a transaction model for increasing concurrency among mobile transactions in MDBCs. The proposed model enforces consistency of databases belonging to an MDBC and provides a high degree of inter-transaction parallelism. A prototype of the AMDB architecture has been implemented based on IBM AGLETS, a Java-based platform.

We conclude this paper by emphasizing that sharing information among multiple heterogeneous, autonomous, distributed and mobile data sources stored in nodes of a MANET has emerged as a strategic requirement for several applications.

References

1. Brayner, A. and M. Filho, José A. "Sharing Mobile Databases in Dynamically Configuration Environments". Advanced Information Systems Engineering. LNCS 2681. 2003. pages 724-737.
2. Brayner, A. and M. Filho, José A. "Increasing Mobile Transaction Concurrency in Mobile Database Communities". Proceedings of the V Wireless Communication Workshop (WCSF03). 2003.
3. Brayner, A. and M. Filho, José A. "AMDB: An Approach for Sharing Mobile Databases in Dynamically Configuration Environments". Proceedings of the XVII Brazilian Symposium on Databases (SBBD). 2002.
4. Brayner, A. and Härder T. “Global Semantic Serializability: An Approach to Increase Concurrency in Multidatabase Systems”. Cooperative Information Systems. LNCS 2172. 2001. pages 301-315.
5. Haas, Peter J. and Hellerstein, Joseph M. “Ripple Joins for Online Aggregation”. Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data. 1999. pages 287-298.
6. Lange, D. B. and Oshima M. “Programming and Deploying Java Mobile Agents with Aglets”. Addison-Wesley. Massachusetts, USA. 1998. ISBN 0201325829.
7. Macker, J. P. and Corson, M. S. “Mobile Ad Hoc Networks and the IETF”, Internet Engineering Task Force, Mobile Ad Hoc Network (MANET) Working Group. Available online at http://www.ietf.org/html.charters/manet-charter.html.
8. Moraes Filho, José Aguiar. “AMDB – An approach for Sharing Mobile Databases”. Master Dissertation. Universidade Fortaleza (UNIFOR). 2003.
9. Soares, M. and Brayner, A. “MXQUERY: An XQuery-like Multidatabase Language”. Submitted for publication. 2004.
10. Urhan, T. and Franklin, Michael J. “XJoin: A Reactively-Scheduled Pipelined Join Operator”. IEEE Data Engineering Bulletin. Vol. 23 No. 2. June, 2000. pages 27-33.
11. XML. Available Online at http://www.w3c.org/XML.
12. XQuery 1.0 Formal Semantics. Available online at http://www.w3.org/TR/query-semantics/.