Middleware Support for Time-Elastic Database Applications*

António Casimiro
FCUL, Lisboa, Portugal
casim@di.fc.ul.pt

Marco Vieira
DEI-CISUC, Coimbra, Portugal
mvieira@dei.uc.pt

Henrique Madeira
DEI-CISUC, Coimbra, Portugal
henrique@dei.uc.pt

1. Introduction

A database is a collection of data describing the activities of one or more related organizations [3]. The software designed to assist in maintaining and using databases is called database management system, or DBMS. A very important notion in DBMS is the concept of transaction [2], which is a set of commands that perform a given action and take the database from a consistent state to another consistent state. DBMS have a long tradition in high dependability, particularly in what concerns data integrity and availability aspects. However, and in spite of the importance of timeliness properties in some applications domains, there are no mechanisms implemented in typical commercial DBMS able to provide timeliness guarantees in the execution of transactions or at least to detect timing failures thereof.

The goal of TACID (Timely ACID transactions in DBMS) [4] is to bring timeliness properties to the typical ACID transactions, putting together classic database transactions and recent achievements in the field of real-time systems and distributed transactions. Three transaction classes are considered in TACID: a) with no temporal requirements, that is, normal ACID transactions; b) with strict temporal requirements, meaning that the transaction has to be concluded within a specified time frame, otherwise the system must detect the timing failure and take some action upon it; c) with relaxed temporal requirements, meaning that a time frame is specified and expected to be met with a given probability, while the transaction will always execute.

In this fast abstract we address some work done in the context of TACID, concerned with the development of middleware support for the execution of relaxed timed transactions. These transactions are particularly relevant in database applications of a so-called time-elastic class, which will dynamically select appropriate deadlines for the executed transactions, following indications provided by the middleware. The estimation of these deadlines is based on probabilistic approaches, using temporal information collected from the execution of previous transactions.

*This work was partially supported by LaSIGE and by the FCT project POSC/EIA/61568/2004 (TACID).

2. Time-Elastic Applications

The time-elastic class of applications includes those that are able to adapt timing constraints during execution. In this case, the collection of information about timing failures and the temporal execution of transactions can be used to feed a monitoring component or to tune specific application parameters in order to adapt its behavior to the actual load conditions of the system. The application may decrease the transaction submission rate, increase the transactions deadline if possible, or postpone the execution of transactions to a latter time. Clearly, these applications can be based on transactions with relaxed temporal requirements. Examples of this class include databases that control mobile communication systems, where connection establishment can tolerate some delays (or may be refused) and billing transactions can be postponed, or continuous manufacturing processes such as chemical processes.

In these applications, the occurrence of timing failures does not compromise the correctness of the database application. Typically, what is important is to secure that the number of timing failures stays below a given bound, over an interval of mission. In other words, what is important to ensure is that a certain QoS is maintained, expressed in terms of a timing bound to be secured with a given (usually stable) coverage (probability that the timing failure does not occur). Therefore, provided that the application is aware of the actual coverage of the timing assumption, it may undertake one of the above-mentioned adaptation strategies, if necessary, when this coverage changes.

3. TACID Architecture

In a typical database environment the client application communicates with the server through a database interface layer (e.g., Oracle Call Interface). This layer is specific for each DBMS and is responsible for managing all the communication with the database server. In order to enable the provision of support for timed transactions, we add a wrapping layer, through which all the communications between the client application and the database interface layer must

go by, which performs the necessary measurements and timing failure detection. Additionally, we also provide in this layer the middleware support for time-elastic applications.

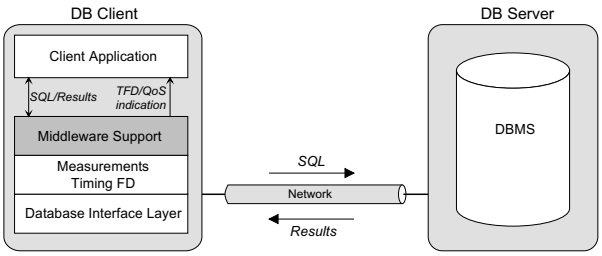

Figure 1. Basic TACID architecture.

Figure 1 shows the basic architecture we propose for adding middleware support in the client interface layer. The application communicates with this layer using an interface that supports the typical SQL requests and delivers the corresponding results. On the other hand, the interface also provides indications about timing failures, sent in the form of exceptions, and about changes in the QoS (the coverage of the assumed timing bound), by explicitly calling a QoS handler of the application. In the wrapping layer, we assume that there exist some mechanisms, or services, to measure the duration of the transactions and to detect their timing failures, such as those provided by a TCB [5].

4. Middleware Support Layer

The middleware to support time-elastic database applications has been developed in the Java language. The programmer must create a class that extends the `TimeElastic` class and define a handler to receive QoS change indications.

Each transaction is defined by specifying its type (strict, relaxed or ACID), a time interval (which is ignored for ACID transactions), the query string and a unique identifier (chosen by the programmer).

Another class, `QoSCoverage`, is provided to allow the specification of QoS requirements and some parameters that are necessary to control how and when indications about QoS changes are sent to the application. An object of this class can be associated to a transaction of the strict or relaxed types. It will monitor the execution times of its associated transaction and, based on probabilistic methods (as described in [1]), it will be able to estimate the achievable coverage for a given time bound. When creating a `QoSCoverage` object, it is necessary to define a timeliness bound and the (initially) required coverage for this bound. During execution, when this object detects a sensible QoS change (the sensibility is also defined at object creation time), it will call the application handler (which is unique for the application), providing the identifier of the transaction and a new time bound, or a new coverage, depending on the selected operation mode (also defined at creation time). It is up to the application programmer to decide what to do with these new bounds.

A very simple example, illustrating the use of this middleware, is provided below.

```
public class TEAppExample extends TimeElastic{
   public void handler(double val, String t_id,
                       boolean OperationMode){
      // perform adaptation procedures
   }
}

long timeout = 50;   // 50ms
double cov   = 0.95; // coverage of 95%
double dev   = 5;    // sensibility of 5ms
int samples  = 30;   // for probabilistic analysis

TEAppExample tea = new TEAppExample();
TACIDTransaction trans = new TACIDTransaction(RELAXED,
   timeout, "SELECT COUNT(*) FROM testtable", "id");
QoSCoverage qos = new QoSCoverage(trans.getTimeout(),
   cov, dev, interval, MODE_TIME, 1);
TACIDConnection conn = new TACIDConnection(DBMSconn,
   trans, TACIDConnection.CLIENT_SIDE);
tea.addTransaction(conn, qos);

TACIDResultSet res = null;

try{
   res = tea.executeTransaction("id");
}catch(SQLException e1){
   System.out.println("Unable to execute query!");
}catch(TACIDTimeoutException e2){
   System.out.println("Transaction timed out");
   if(tea.getTransactionType("id") ==
      TACIDTransaction.RELAXED){
      res = tea.getLastResults("id");
         // Wait for transaction results
   }
}
```

References

[1] A. Casimiro and P. Veríssimo. Using the Timely Computing Base for Dependable QoS Adaptation. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems*, pages 208–217, New Orleans, USA, Oct. 2001.

[2] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Jim Gray, 1993.

[3] R. Ramakrishnan. *Database Management Systems*. McGraw Hill, second edition, 1999.

[4] TACID (Timely ACID Transactions in DBMS) web page. http://gbd.dei.uc.pt/view_project.php?id_p=55.

[5] P. Veríssimo and A. Casimiro. The timely computing base model and architecture. *IEEE Transactions on Computers - Special Section on Asynchronous Real-Time Systems*, 51(8):916–930, Aug. 2002.