An Interoperability Approach Based on Asynchronous Replication Among Distributed Internet Databases

Douglas D. J. de Macedo[1], Hilton W. G. Perantunes[2], Luiz F. J. Maia[1], Eros Comunello[2], Aldo von Wangenheim[1,2] and M.A.R. Dantas[1,2]

Federal University of Santa Catarina (UFSC)
Department of Informatics and Statistics (INE)[2]
Post-Graduate Program in Knowledge Engineering and Management (PPGEGC)[1]
Florianopolis, Brazil 88040-900
{macedo,william,maia,eros,awangenh,mario}@inf.ufsc.br

Abstract

Nowadays, there is a growing interest in telemedicine in Brazil, because the country is the fifth largest country in the world and there is a huge gap between the population size and the number of available doctors. Institutions and the government are considering solutions for the integration and availability of data produced by several different systems. In this paper we present a contribution that is characterized by an asynchronous replication strategy between distributed medical databases. The contribution, named PGR, is an enhancement to the PostgreSQL database software that introduces lightweight asynchronous operations. In addition, we have implemented a multi-master engine with partial replication and hybrid fragmentation in order to allow interoperability between different telemedicine systems. Our early results indicate that the model and its implementation have reached a successful level in terms of performance and interoperability.

1. Introduction

The eight million, five hundred and eleven thousand, nine hundred and sixty-five square kilometers of Brazil is completely covered by wired and wireless telecommunication facilities. Therefore, an individual in the middle of the Amazon region or in the middle of the Swamp region of the country can get such help as a diagnosis assessment from a specialist doctor. This scenario has clear advantages in terms of territorial covering and usage of hospitals and clinical equipment. This strategy is broadly presented in some researches (e.g., [6]), but different efforts are necessary to meet local conditions. Because of this, in the south of Brazil, in the state of Santa Catarina, an effort was established in the project RCTM (Santa Catarina Telemedicine Network) [5].

Both the design and the implementation of this telemedicine effort is executed by the Cyclops Project [16] in the Federal University of Santa Catarina (UFSC). The project offers a telemedicine portal where services such as a second doctor assessment, collaborative diagnosis, and available hospital procedures.

The RCTM project concept consists of a model in which hospitals from all cities of the state produce patient exams, e.g., electrocardiogram, computed tomography, magnetic resonance, X-ray angiography and nuclear medicine.

These exams are immediately sent to a centralized database located in the University Hospital. This centralized database is accessed by doctors that execute the necessary diagnoses. This project has proved to be a success and many other states are interested to implement a similar approach. In addition, as a national plan, the idea is to interconnect all databases, avoiding that the need for a patient to go through many entrances and similar exams in different places across the country.

Interesting and challenging problems arise from this national plan, and representative examples are scalability issues and telemedicine database replication. This fact leads directly to the motivation of this research, which can be summarized as design and implementation of an asynchronous replication strategy between distributed medical databases. In addition, an important fact is that the final proposal would be based upon an *open source* paradigm, which could be used by all peers of the national project.

The paper is organized as follows. Section 2 shows related work. A brief overview related to theoretical aspects and issues of data replication in distributed system configurations are shown in section 3. In section 4, the PGR (*PostgreSQL Replication*) proposal is presented, where there is a deep discussion of the issues and how to tackle these problems, aiming at a high performance. Experimental results from the present research are pointed out in section 5, where there are some comments related to these experiments. Finally, in section 6 we show conclusions and future work.

2. Related Work

It was mentioned in the previous section that the national plan for medical database interoperability would consist of several peers across the country. In addition to that, it is within the scope of the project to increase the software and experience exchange. As a result, our proposal, presented in section 4, is an implementation based on *open source* code. However, in this section we present some efforts in both segments (i.e., *open* and *not open*).

Examples of *open source* data replication software packages are: PGPool [14], PGCluster [10], Postgres-R [11] and Slony-l [9]. On the other hand, IBM DB2 Replication [12], Oracle (RAC) Real Application Clusters [13], and SyBase Replication Server [15] are classic *closed source* software packages.

Table 1 shows the replication type and mode, high availability, and load-balancing characteristics of several software packages.

The Slony-1 [9] is a system based on a single master node to serve multiple slave machines. This software was designed to act as central backup node in a configuration where all other nodes are subject to data loss. As a result, the package implements pessimistic algorithms with asynchronous replication [9]. Slony-1 is not the answer for a project where a specific hardware configuration could be unavailable.

The middleware PGPool [14] acts in the middle of a PostgreSQL node a data client. PGPool offers services such as connection control and pooling, replication, and load and query balancing. The package creates in real time a backup node. The main characteristic operations of PGPool are being a central pool server and not working perfectly with data replication.

The Postgres-R is an extension of PostgresSQL that focus on an efficient, fast and consistent replication for database clusters [11]. Thus, this package is used for load balancing and high availability features in a distributed database configuration. However, the major drawback of this system is the necessary synchronous operation among the nodes.

A multi-master solution, called PGCluster [10], allows synchronous replication among PostgresSQL nodes. The latency for the duplicated data is low, and there is high availability of data because of its multi-master characteristics. However, this software requires high bandwidth as a mandatory infra-structure. This is an issue to be considered for the national project (or even for a state alone), since the scenario one faces presents heterogeneous connection facilities.

The Telemedicine Portal from the RCTM project utilizes PostgresSQL. Therefore, one interesting solution proposal would be to improve some limitations of the PostgresSQL package, namely the high cost for interoperability between distributed telemedicine databases. This new lightweight interoperability strategy is presented in section 4, and we named it PGR. Table 1 illustrates some characteristics of the PGR proposal.

Table 1. Database packages characteristics

Name	Replication Type	Replication Mode	High Available	Load Balance
Slony-l	Single-Master	Asynchronous	Yes	No
PGPool	Multi-Master	Synchronous	No	Yes
Postgres-R	Multi-Master	Synchronous	Yes	Yes
PGCluster	Multi-Master	Synchronous	Yes	Yes
PGR	Multi-Master	Asynchronous	Yes	No

3. Data Replication

Data replication is an active subject in the research of distributed databases [7, 8, 17]. The main concerns in database clusters are scalability, autonomy and full (or partial) replication [2, 3]. In other words, the number of nodes, with different types of replications, could grow and the system should preserve the same performance.

Traditional techniques for data replication make use of full distributed consistency as an approach to guarantee to a user that the data is available [1]. These techniques are usually known as a pessimistic approach, because the synchronism function is necessary. The pessimistic strategy is suitable for local area networks, where latencies are low and faults are not common.

In contrast, in wide area networks, where communication latencies are high and faults are frequent, another technique should be employed. As [8] mentions, optimistic replication is the right paradigm for long distance networks. Optimistic techniques use asynchronous primitives between nodes [4]. There is no need for synchronization in order to access data. Updates and conflicts are handled in background. The Internet's Domain Name System (DNS) and the CVS software [8] are two examples of optimistic replication. Interesting characteristics when one uses optimistic replication are: a node from a database cluster can continue to work with its own replicas when others nodes fail; a communication link responsible for node interconnection can also be unstable and the system will still be working; high scalability, because of the low-traffic communication between nodes [8]. However, data consistency is the price paid for this relaxation. Data across a database cluster can be inconsistent. As a result, the challenge is to employ an optimistic paradigm assuring data consistency.

One relevant aspect related to data consistency is the number of *replicas* and how to store these copies. In a *master-slave* approach, a master node is responsible for all updates and then these *replicas* are sent to slave nodes. On the other hand, the *multi-master* strategy allows multiple updates in diverse nodes, resulting in complex operation management.

4. The PGR Proposal

Telemedicine applications are very specialized, because strict rules are applied to the program and its utilization. Databases in this area are well known to store a large amount of data. These data are usually composed of: name of patients, doctors and hospitals; images from complex exams (e.g., electrocardiogram, computed tomography, magnetic resonance, X-ray angiography and nuclear medicine). Because of that, the size of these databases can reach up to several *terabytes.*

Typically, databases used in our experiments store information related to studies from patients. A study is a specific set of exams that are used for a diagnosis. Table 2 illustrates a set of exam types and their sizes. Each day several exams for different patients are done, thus the number and size of the database increases on a daily basis. According the table 2, an upward trend in terms of database size is expected.

Table 2. Exams and its typical sizes

Exam Type	Average Type	Number of Items	Total Average
Computed Tomography	500Kb	100	50Mb
X-Ray Angiography	70Mb	8	560Mb
Nuclear Medicine	1Mb	5	3Mb
Magnetic Resonance	250Kb	200	50Mb

The greatest challenge found in distributed database replication for telemedicine is the amount of scattered exams that can occur in the system. In other words, the amount of data is large and there is a high probability of data duplication if an efficient control is not employed. It is important to observe that any strategy to tackle this problem should consider the large amount of data that can not have a full backup in a master-slave topology. In other words, the history from a specific patient could end up being made up of entries in different nodes. Therefore, if an efficient tool is used, no useless replication will be allowed. In addition, any consolidation of a history should spend but a small fraction of the available bandwidth.

The expansion of the RCTM to a national level will require a multi-master approach with some specific characteristics.

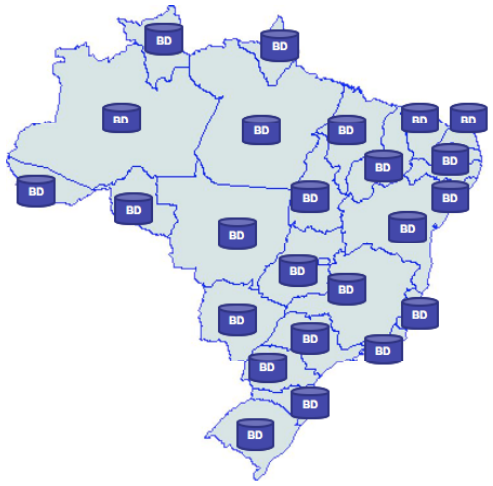

Figure 1. An illustration of a national plan for the medical database

Figure 1 shows the Brazilian states and an ordinary division of the medical database. However, aspects such as the huge difference of existing doctors per region (e.g., in north this number is equal to 4.1%, whereas in south-east 57.7%), exemplifies some concerns in the design of the project. Another concern is the number of existing patients per region. An example that illustrates this problem is the fact that the number of people living in the state of Santa Catarina is similar to the number of people living in the city of Rio de Janeiro alone.

In this context, our proposal is to design and implement a solution, which could be considered a partial replication one in a multi-master topology, adopting asynchronous communication and hybrid fragmentation.

The contribution proposed in this research work was named PGR (*PostgreSQL Replication*). The goal of the PGR proposal is to allow a subset of medical database tables to be monitored in an asynchronous replication fashion. Utilizing these tables, it is possible to execute horizontal and vertical data fragmentation. In other words, it is possible to filter some entries through some query conditions and obtain only relevant information for the fragmentation application.

This fragmentation is necessary, because it is expected that some nodes will not replicate medical images. This fact can be explained by the high cost of bandwidth and storage of an entire database that is to be published to other nodes. Therefore, as we suggest in this research work, a partial replication strategy is employed.

Our proposal is characterized by a structure of relational databases, in each replica of the environment that keeps system control information. In addition, the proposal includes an application module that executes in each master node, which supports databases and is able to establish connections for replication coordination.

The proposal assumes that all monitored relational databases have the same structure. This premise is necessary because the data propagation adopted is the multi-master paradigm. In this approach any replica expects to find in other tables the same fields that exist inside its own table (i.e., names and attributes). Obeying this rule it is possible to make connections and queries in all nodes inside the environment.

If an update to a local database were to occur in a synchronous master configuration, connections to all nodes would be necessary. In contrast, in the PGR approach only information related to the update is stored (inside a table of events). The data thus stored have information about the operation, i.e., when and where it occurred. The strategy adopted by the PGR is a two-phase process.

In a specific moment, a second replica will connect to the local database and retrieve the information about the operation. In other words, after this connection the non-local node will obtain information related to the update done. After getting the information, the node can update its own table. The PGR proposal also stores information about this update for future utilization. In a future request from the remote node to the local one, only actions that happens after that time will be captured by the remote node.

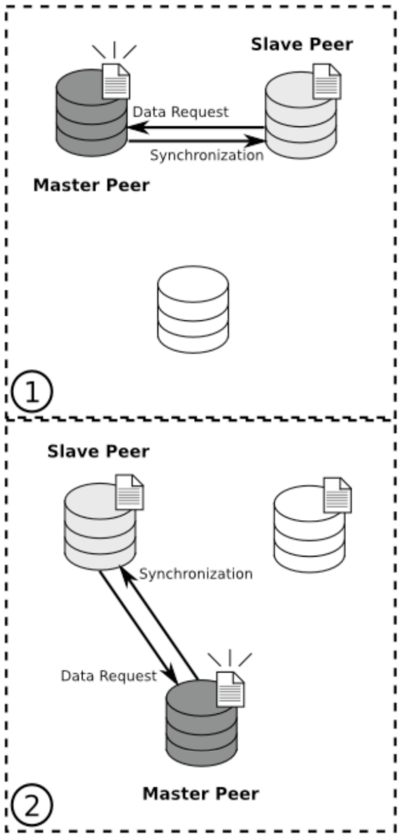

Figure 2 – Replicas communication in the environment

Figure 2 shows the replication data process in the PGR environment, which is realized in a number of *n steps*. This *n steps* update propagation is similar to a master-slave paradigm. One specific node establishes a connection to another node and captures (part 1 of figure 2). In a second moment, illustrated in part 2 from figure 2, another node will take over the master position and will realize the same operation.

All data gathered from updates are stored inside the PGR in tables called *shadow*. A *shadow* table has a similar structure of a monitored table, with one additional column that point to the original database. Querying monitored tables inside the database is done through the original tables and their respective *shadow* tables, considering that it is possible to filter data when one takes into consideration its origin.

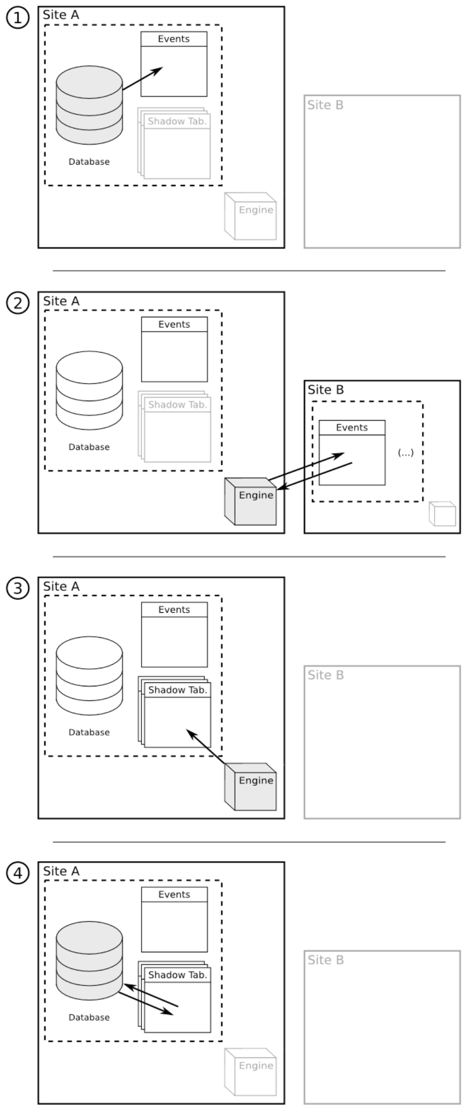

Figure 3 – Communication process and update between nodes.

Figure 3 illustrates how the communication occurs between databases when updates are realized in a replica through four examples. Frame 1 (from figure 3) shows that updates in Replica A are stored inside a PGR event table. This table exists in all replicas and the application access it to obtain information related to the most up-to-date table, as shown in frame 2.

The PGR is responsible for recording all information obtained from other databases in a special replication table. This feature is illustrated in frame 3 and is represented by a *shadow table*.

Finally, a database can use relations from *shadow* tables in queries related to the monitored tables. Therefore, it is possible to gather update information that occurs in the other replicas from the group, as it is presented in frame 4.

5. Experimental Results

In order to realize tests with the PGR proposal, we established the location of several databases and their connections, as figure 4 shows. This scenario reflects a real operation environment where exams, images, videos, and other types of information can be exchanged. Considering the distribution proposed in figure 4 it is possible to realize tests utilizing the PGR.

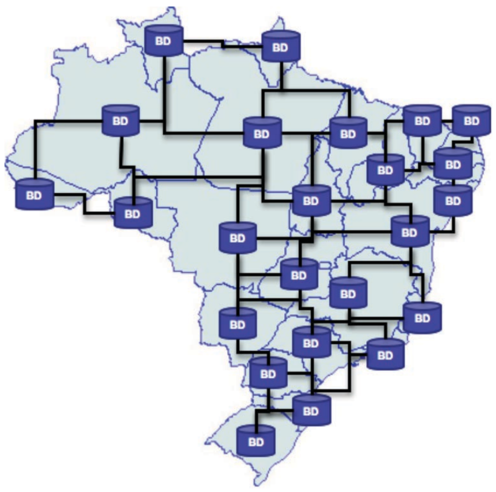

Figure 4. A database distribution and connection scenario for tests

In our empirical tests we have used *chunks* with different sizes and twenty experiments were considered for the data replication between remote entities.

In the first experiment, *chunks* of *binary large objects* (*blob*) were utilized. Aim the second, only plain text was used. Table 3 shows the data size employed in the empirical tests.

Table 3. Chunk sizes

	10	100	1000	10000
Binary Objects	0.34 Mb	26.20 Mb	275.07 Mb	1,712 Mb
Text	8 Kb	16 Kb	104 Kb	1,128 Kb

Twenty replications of medical images and plain records were considered in chunk sizes varying from 10 to 10,000 records. In our experiments, we measured the time to transfer those chunks between two different sites. Each measurement consists of a fraction of second, and was taken by employing a native tool from the operating system that exists inside the site responsible for querying a replica. Therefore, we were able to record the exact time span between the beginning of data transfer (by the controlled engine) and the indication of a successful transfer.

The operation of gathering records was characterized by the use of temporary tables in the database of the site which holds the original data chunk that will be transferred. The controller of the primary site requires a structure of tables to reproduce this in its own database. Afterwards, it copies all data employing the copy procedure existing in the PostgresSQL. The mean time to accomplish these operations, considering text and binary formats, is presented in table 4.

In order to have a better visualization of the data collected from the replication operation, the amplitude normalization formula was used:

$$\frac{|\bar{x} - x_i|}{x_{max}}$$

This normalization was made through the division of the absolute value of the difference between each value and the mean observed value, and the maximum value for each size.

Table 4 Replication Mean Time in seconds

	10	100	1000	10000
Text	0.0048	0.0091	0.0170	0.1836
Binary Objects	0.0665	9.6952	104.4264	627.1346

The result from the previous described division is a number that fits in the interval between 0 and 1. Therefore, we were able to consider variables with different degrees of dispersions. Those values were multiplied by 100 to represent a relative cost of data transfer. Table 5 shows these costs for each kind of record.

Table 5. Replication cost per record

	10	100	1000	10000
Text	18.25	3.40	7.46	7.39
Binary Objects	0.97	0.47	0.59	0.08

It can be observed from the normalized values of figure 5 that the latency for transferring plain text is high for small packet sizes and it is almost the same for chunks greater than 1,000. On the other hand, binary objects have a similar behavior in terms of data transfer cost.

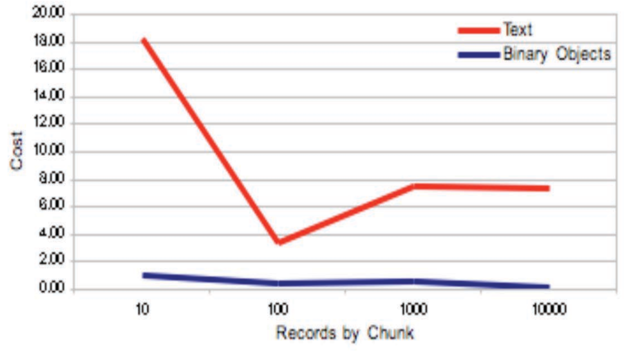

Figure 5. Data transfer cost

6. Conclusions and future work

In this paper we have presented a proposal that is characterized by an interoperability approach based on asynchronous replication among distributed Internet databases.

The contribution, named PGR (*PostgreSQL Replication*), had the goal to allow a subset of medical database tables to be monitored in an asynchronous replication fashion. Utilizing these tables it was possible to execute horizontal and vertical data fragmentation. In other words, it was possible to filter some entries through some query conditions and obtain only relevant information for the fragmentation application.

Experimental tests done using samples of medical data and operational knowledge from the RCTM Project [5], indicate that the lightweight strategy is capable of successfully synchronizing databases from different sites. The partial and optimistic replication, along with a hybrid fragmentation, were used for an experimental environment characterized by a high latency communication cost. This indicates that the proposal is suitable for the telemedicine configuration, with clear advantages when compared to the ordinary PostgresSQL.

As a future work, we are planning to develop a more complex schema to detect data inconsistencies and resolve possible conflicts. In addition, a backup procedure to ensure high availability of the configuration is also another challenge to be tackled.

References

[1] Bernstein, P. A.; Hadzilacos, V.; Goodman, N. Concurrency Control and Recovery in Database Systems. Massachusetts: Addison Wesley, 1987.

[2] Coulon, C.; Pacitti, E.; Valduriez, P., "Scaling up the Preventive Replication of Autonomous Databases in Cluster Systems," VECPAR, LNCS 3402, pp. 170-183, 2004.

[3] Coulon, C.; Pacitti, E.; Valduriez, P., "Consistency management for partial replication in a high performance database cluster," Proceedings of 11th International Conference on Parallel and Distributed Systems. Vol. 1, pp. 809-815, 2005.

[4] Goel, A.; Pu, C.; Popek, G.J., "View consistency for optimistic replication," Proceedings of Seventeenth IEEE Symposium on Reliable Distributed Systems. pp.36-42, 1998.

[5] Maia, R.S.; Wangenheim, A. von; Nobre, L.F., "A Statewide Telemedicine Network for Public Health in Brazil," *19th IEEE International Symposium on Computer-Based Medical Systems, 2006. CBMS 2006.*, pp.495-500, 2006.

[6] Mcneill K. M.; Weinstein R. S.; Holcomb M. J., Arizona Telemedicine Program: Implementing a Statewide Health Care Network, Journal of the American Medical Informatics Association, Volume 5 ,Number 5 , Sep / Oct 1998.

[7] Ozsu, T.; Valduriez, P.: *Principles of Distributed Database Systems.* Ed. 2, Pretince Hall, 1999.

[8] Saito, Y. and Shapiro, M. 2005. Optimistic replication. ACM Computer Surveys (CSUR). Vol. 37, n. 1, p. 42-81, 2005.

[9] Slony-l. Enterprise-level replication system. http://www.slony.info, 2008

[10] PGCluster. The multi-master and synchronous replication system for PostgreSQL. http://pgcluster.projects.postgresql.org, 2008

[11] Postgres-R. Eager multi-master replication for Postgres. http://www.postgres-r.org, 2008

[12] Gu, L.; Budd, L.; Cayci, A.; Hendricks, C.; Purnell, M.; Rigdon, C.; A Pratical Guide to DB2 UDB Data Replication V8. http://www.ibm.com/redbooks, 2008

[13] Oracle. Oracle Real Application Clusters (RAC) 11g: *An Oracle Technical White Paper*. April, 2007. http://www.oracle.com.

[14] PGPool. http://pgpool.projects.postgresql.org., 2008

[15] SyBase. Replication Server: Move and synchronize data across the enterprise with Replication Server. http://www.sybase.com, 2008

[16] Cyclops Group, http://cyclops.telemedicina.ufsc.br/, 2008.

[17] Dantas, M. A. R., Cunha, D. P.: An Experimental Case Study of Replication on Reconciliation in a Wireless Environment. HPCS 2004: 179-182, 2004.