Project no. 004758

GORDA

Open Replication Of Databases

Specific Targeted Research Project

Software and Services

Strategic Research directions

GORDA Deliverable D1.3

Due date of deliverable: 2005/03/31
Actual submission date: 2005/03/31

Start date of project: 1 October 2004 Duration: 36 Months

Universidade do Minho

Revision 1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

1 Summary

This document is builds on deliverables D1.1 (State of the Art) and D1.2 (User Requirements) to update and detail the strategic research directions identified in the DoW. Research in the scope of the GORDA project will be carried in four main areas required to the fulfillment of the main project goals: Architecture, database replication protocols, group communication protocols, and management tools. These are detailed in the next sections. In short, we identify the following strategic research directions:

- An architecture that abstracts interaction between database management systems and replication protocols.
- Replication protocols for interconnected clusters for both performance and high-availability, with emphasis on write scalability.
- Hybridization and development of current database replication protocols based on group communication to provide widely applicable practical solutions.
- Study of the performance and dependability characteristics of database replication protocols in different environments and workloads.
- Recovery protocols that interoperate with the proposed database replication protocols and provide high performance and fault-tolerant reconfiguration.
- Optimization of implementations and interfaces of group communication protocols in current environments and database replication workloads.
- Innovative strategies for adapting, reducing, or masking the cost of total order in group communication systems.
- Interfaces, mechanisms and strategies for autonomic management of replicated database systems.

2 Architecture

Kernel based replication solutions take advantage of internal components of the DBMS and result in an increase of performance in terms of throughput, scalability and response time. However, for the sake of portability database replication must be independent from the underlying database engine, that is, protocols must not rely on any modifications to the target DBMS. Furthermore the internals of the database are mostly not accessible and if they are, they are complex and difficult to change.

Therefore database replication at the middleware level has received a considerable attention in the last years. Such solutions can be maintained independently of the database system, and can potentially be used in heterogeneous settings. The downside of this approach is no

access to concurrency control information at the database engine. Therefore such replication protocols often suffer from duplicating the back-end database logic into the middleware. Since SQL statements visible at the middleware do not necessarily indicate the exact records to be accessed, concurrency control provided is usually at a coarse level (table based). Moreover, some of the existing solutions require transactions to pre-declare specific properties: for instance, tables accessed by a transaction or the type of transaction (update or read-only) has to be known in advance, that is before the transaction's execution.

A major research goal of the GORDA project is to design a replication framework that abstracts the complexity of interfacing replication protocols and database management systems. The resulting architecture and programming interfaces must be realizable either as middleware or in the DBMS kernel thus avoiding the restrictions of both existing solutions. Therefore, the proposed architecture should match the semantics of standard database interfaces, such as JDBC and ODBC, but also the internal semantics of relational database management systems for efficiency. It should also allow that replication is fully transparent: a client application interacts with a database via SQL statements and gets an illusion of accessing a single, highly available database system.

3 Replication Protocols

Little research has considered the applicability of the existing database replication protocols in WANs. Replication here is more complicated due to different characteristics between LANs and WANs, notably much larger communication latency. Some key points to obtain acceptable performance across WANs are to reduce communication overhead as much as possible, to overlap execution with message delay, and to relax consistency requirements. Replication in WANs can be seen as a replication between geographically dispersed database clusters. The main ideas behind this assumption is that transactions are localized, that is, most of the time a particular database cluster processes transactions that access a specific part of the database only. Different consistency and fault-tolerance guarantees can therefore be insured within local clusters and over wide area networks.

A particularly important issue in database replication protocols is to achieve write scalability by making it practical to partition and separately replicate each fragment of data. This addresses a fundamental limitation in database replication: By scaling up the amount of transactions that can be processed by replicas, the amount of storage bandwidth required grows. Dealing with partial replication has a profound impact on replication protocols but also on other system components such as group communication and management tools.

Different database replication protocols based on group communication provide different performance and fault tolerance tradeoffs. As an example, some protocols based on optimistic execution provide better performance with little configuration effort but degrade with certain workloads causing a large number of aborted transactions. In contrast, protocols based conservative scheduling require an additional effort in defining appropriate conflict classes for high

performance but avoid unfairness. In order to allow practical application of protocols it is necessary to precisely characterize and understand their behavior in different environments and workloads. This provides the basis for both manual and autonomic management techniques and tools. It provides also the basis for devising hybrid protocols that can perform adequately in a wide range of environments and workloads by exploiting the best features of each protocol or automatically adapting to the environment and workload.

Replica management and recovery in the context of group communication based replication protocols both in LANs and WANs has not been thoroughly examined. Many of currently proposed replication protocols rely on the state transfer mechanism of group communication to cope with recovery of failed sites, merging of partitions or joining of new sites. Therefore the database sites update and resynchronize their database copies by copying the entire database state to another site. This consumes time and bandwidth, as normal update operations of the database system are interrupted. If the amount of data to be copied is large, the availability requirements of many critical systems might be violated. Obviously it is impractical to perform data transfer as a single atomic step. Furthermore, the database consistency must be ensured after transfer: if some replicas are available for transactions processing during data transfer, the new or recovering site might never be able to catch up with such nodes.

A similar problem occurs while making an online copy of the database for archiving purposes. Traditionally, this is done by isolating the active replica from the system and doing the copy offline. Consequently the following question arises: is it possible to do this online in an efficient way? Since it depends on the update transactions ratio in the workload and the database size, one way to solve the problem would be to consider the workload while designing a state transfer mechanism. Furthermore to minimize the blocking time of the active replicas during the state transfer, we intend to investigate different logging techniques and parallel recovery techniques. For example, only the missed messages or, more specifically, those data items that have been changed during the failure are transfered to the recovering replica. Moreover, only the last versions of updated data items are copied.

4 Group Communication Protocols

As described in the previous section, replication protocols based on group communication are expected to be a major goal in the project. Although there is already a substantial body of research on group communication, there are still many open issues of relevance to the GORDA objectives that will be addressed. In the following, we identify those that we believe are the four most challenging research issues in the area and that set our research agenda.

Group communication for different operational envelopes has quite different implementations. For instance, implementations optimized for wide area are generally quite different from the implementations optimized for local area networks. However, GORDA is targeted to operate in challenging environments that combine clusters of local area networks interconnected by wide area networks. We are interested in the design of adaptive protocols that can

autonomously reconfigure to match the execution environment with minimum management overhead.

Complementing the line of work above, it is interesting to notice that most of the implementations of group communication for local area network were designed more than a decade ago. The properties of local area networks have changed dramatically since then. Issues such as new latency/throughput ratios, the performance and properties of LAN-switches, etc, must now be taken into account to optimize the performance of group communication protocols in today's LANs. We will study how the evolution of local area network has impacted the performance of group communication protocols.

From the set of group communication protocols to be used in GORDA, Total Order protocols are the most fundamental but also the most expensive, both in terms of communication steps and number of messages exchanged. This problem is exacerbated in large-scale systems, where the performance of the algorithm may be limited by the presence of high latency links. To alleviate this performance problem, Optimistic Total Order protocols have been proposed. Such protocols provide to the application an early indication of the estimated total order. The application can use this estimate to perform a number of actions optimistically, which are later committed when the final definitive order is established. The goal is to execute some application steps in parallel with the communication steps of the Total Order algorithm. Different Optimistic Total Order protocols can be found in the literature. Although most of these protocols share the same designation, they offer quite different services. Research will be conducted on how to integrate these different approaches in a single coherent and adaptive component of the GORDA architecture.

Independently of the quality of the implementation of group communication protocols, their use by commercial products, as envisaged by GORDA, requires the services to be exported using interfaces that are industrial standards (formal or "de facto"). One of these interfaces is the Java Naming and Directory Interface (JNDI), which is J2EE compliant. However, it is widely known that the services interfaces may be a major bottleneck in system performance, namely if they require intensive data copying or force multiple context switches. Therefore, we will need to study how to implement standard interfaces without hampering the performance of the group communication support.

5 Autonomic System Management

The main challenges in providing a management system for large-scale replicated databases can be found on two main axes: the complexity and the heterogeneity axis.

The *complexity axis* refers to the large number of layers (i.e. vertical complexity) and entities (i.e. horizontal complexity) present in the underlying managed system. Horizontal complexity of enterprise systems or grid-based systems refers to the fact that those systems could have hundreds and thousands of nodes serving different functionalities. Furthermore, some systems such as Internet-based services are distributed in tiers comprising a web tier for serving web

content, a business tier for handling complex system interactions and a database tier containing the system data. And each tier consists of a number of nodes interconnected in clusters. Moreover, the vertical complexity of a system is induced by the fact that each node has several software packages running together to provide the requested functionality. A consistent software stack runs on each node and comprises the OS, virtual machines (in case of Java for example), middleware products, and the actual application logic, or user data. Each of this stack elements must function correctly and efficiently at all times and it must be monitored and managed independently, while preserving the overall view of the system.

The *heterogeneity axis* refers to the variation in software and hardware entities across the large pool of servers providing the overall system functionality. In grid systems for instance, machines could have different types of OS, of middleware or indeed of DBMS running in a collaborative manner. Approaching all these different entities in a consistent manner is the key to efficient management. Significant research efforts are required to ensure that the GORDA Management system (GMS) will provide the appropriate interfaces, independent of specific database implementations. They must be generic in order to accommodate the common functionality and replication capabilities of most database systems, while at the same time, be flexible enough to allow implementation-specific optimizations be added.

In terms of the capabilities required by the GMS, the research will be on pursuing optimal solutions for dealing with repair and configuration. In detail, the management system should be fault-tolerant with respect to the failure (crash) of the machines on which it executes. It should support at least a basic repair policy consisting in updating the failed managed system to a configuration that conforms to the same architecture description as the configuration in place prior to the occurrence of failure. The management system should be self-healing in the sense that it should deal with failures occurring in the repair management system itself. In other words, repair management should also function to repair itself.

In addition, the management system should be dynamically deployable. In particular, it should be possible to set up repair management on an already running managed system, and it should be possible to dynamically modify the different elements of the repair management system (e.g. its management policy, its repair algorithms, its monitoring structure, etc.). The management system should allow human administrators to monitor the activity of the overall system, and to manually effect repairs, in replacement of the standard repair management behavior. A human administrator should in addition be allowed to dynamically set up a management console on an arbitrary node (i.e. a management console should not be tied to a specific node in the system and should be dynamically deployable, as per the dynamic configuration requirement above).

Finally, the management system should be applicable to legacy systems (possibly with reduced requirements). In general, a legacy application usually appears as a "black box" providing immutable interfaces for activation and monitoring; its internal structure is not accessible. This aspect is related to the provision of common interfaces to accommodate most current implementations.

The management system should enforce a strict separation between mechanism and policy, with the goal of allowing the execution of different management policies, possibly at different levels of abstraction. In particular, it should allow for different configuration and deployment policies, from the exact specification of selected components and locations at which they should be deployed, to the indirect specification of selected components and their locations through high-level management goals and constraints. For instance, one can have different levels of declarations in the specification of deployment and configurations: the selected components and the location at which they should be deployed are explicitly specified; the components and locations are not explicitly specified, but the administrator may specify preferences; the specification is in terms of higher-level goals, e.g. select the components and the locations for maximal availability or maximal performance. The operations of the management subsystem should interfere minimally with the behavior of the managed system. Thus, the repair management system should preserve the correct operation of the managed system, and the performance interference between the nominal behavior of the system and its behavior under management should be minimal.