



Project no. 004758

GORDA

Open Replication of Databases

Specific Targeted Research Project

Software and Services

# User Case Studies and Requirements Report

## GORDA Deliverable D1.2

Due date of deliverable: 2005/01/31
Actual submission date: 2005/02/15
Revision 1.1 date: 2006/12/30
Revision 1.2 date: 2007/05/17
Revision 1.3 date: 2008/05/09

Start date of project: 1 October 2004 Duration: 42 Months

Continuent

**Revision 1.3**

| **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) | |

## Contributors

Edward Archibald, Continuent
Emmanuel Cecchet, Continuent
José Pereira, U. Minho
Robert Hodges, Continuent
Rui Oliveira, U. Minho

**Abstract**

This report identifies relevant use cases and requirements embodying the key issues in database replication in current information systems. The selected use cases stem essentially from Continuent's experience and its users base. These include current client deployments as well as short term prospective scenarios for which base requirements are known. The report starts by describing each use case in detail and then presents a set of requirements generally applicable.

# Contents

# Chapter 1

# Introduction

An explicitly stated aim of the GORDA project is to "foster research, development and deployment of database replication solutions". While the majority of the GORDA participant institutions are from the academic community, it is becoming clear that this community of participants is learning to strike a healthy balance between providing solutions for ***interesting*** problems and providing solutions for ***real world*** problems.

The use cases and system requirements presented in this document reflect this balance of the ideal and the practical and lay a substantial foundation for the elaboration and validation of the GORDA system architecture, implementation and deployment The GORDA flexibility will provide Continuent with a framework to address all client scenarios and requirements.

The document starts by presenting the use cases for a set of industry standard benchmarks that are widely considered to be representative of workloads encountered in real world E-Commerce and OLTP applications. Then we present several real-world case studies and model a set of use cases based on them. Finally, we present the complete set of requirements for GORDA. These requirements have been referred to, refined, and updated during the life of the project.

## 1.1 Objectives

The GORDA User case Studies and requirements report, has the following goals:

- identify relevant case studies embodying the key issues in database replication in current information systems;
- define a set of usage scenarios for GORDA compliant databases;
- establish the base for a set of strategic research guidelines.

## 1.2 Relationship With Other Deliverables

By identifying the usage scenarios for GORDA compliant databases, this document is of major relevance for Deliverable D1.3 - Strategic Research Directions Report.

Furthermore, this document guidelines will be taken into account by Deliverable D5.2 - Prototype of the Integrated System, on the choices shaping the resulting prototype.

# Chapter 2

# Case Studies and Use Cases

## 2.1 Basic Availability Case Study and Use Cases

### Overview of Availability Problem

Continuent's experience with commercialization of Sequoia clustering has demonstrated the need for basic availability for a wide range of customers. Open source databases have low barriers entry due to the fact that users can download them for free and develop basic applications with a minimum of difficulty. Many of these applications quickly prove their worth and become invaluable to the organizations that use them. At this point, database availability suddenly becomes a significant concern and a focus of management attention.

In a study of recent customer engagements and sales prospects undertaken in Q4 2007, Continuent found that database availability was a strong or predominating need for almost 100% of the cases studied (in fact, all but one from a sample of approximately 30). By contrast, scalability of databases was a concern for between one third to one half of the cases. There is an overwhelming need for a transparent, easy-to-deploy solution that users can deploy in local as well as hosted environments without requiring specialized hardware or introducing complex administration procedures.

Maintaining copies of databases using data replication technologies is an ideal solution to basic availability. Many organizations apply database replication in an ad-hoc manner (for example using scripts and native master/slave replication) to implement basic availability. However, this solution results in extra management overhead and complexity that many organizations can ill afford. A properly managed cluster is a much better answer. The following diagram shows a typical deployment scenario.

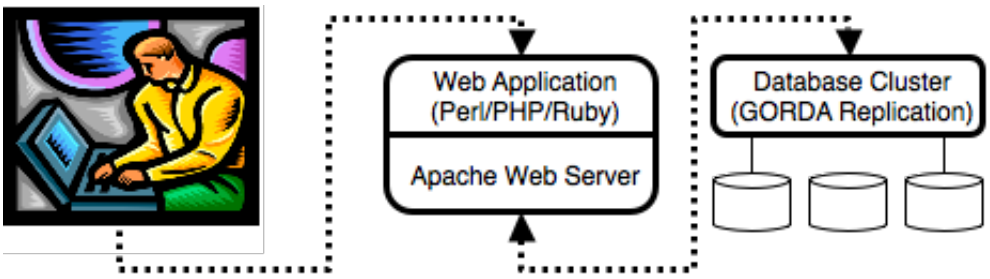


Figure 2.1: Basic Availability Scenario

While the availability deployment scenario superficially resembles use cases shown later in this requirements document it is very important to stress the simple nature of such setups. The following table summarizes typical characteristics of basic usability deployments.

| Characteristic | Description | Typical Values |
|---|---|---|
| Update rate | Number of updates per unit time | 1 to 100 per second |
| Read rate | Number of read operations per unit time | 10 to 1000 per second |
| Data Volume | Size of database | Less than 50 GB |
| Transactions | Use of SQL transactions | Commonly omitted, especially for Perl or PHP |

It is clear that from a performance perspective these requirements can be met using a single database server running on rather modest hardware. The need for clustering and hence replication is purely driven by availability considerations.

### 2.1.1 Use Cases

In the following we describe specific use cases comprising this use profile.

***CS-AVAIL-A-1: Upgrade from Single Database to Cluster*** The user upgrades an existing standalone database to a clustered database. Data from the single database are copied to each replica comprising the cluster. The cluster is then started. In the event of a cluster initialization failure, user applications can use the standalone database.

***CS-AVAIL-A-2: Connect Applications to Cluster*** The user halts applications, configures application connection strings to connect to the clustered database, and restarts applications. The applications come up using the cluster without any other changes.

***CS-AVAIL-A-3: Disable Database for Maintenance*** The user disables one of the databases in the cluster in order to perform scheduled maintenance. On disabling, SQL requests are transparently shifted to other databases in the cluster. The user completes maintenance and re-enables the database in the cluster. The database is synchronized with other cluster databases. Once this is complete, SQL requests are directed to the newly enabled database.

***CS-AVAIL-A-5: Failover after Database Crash*** A database host encounters a CPU panic during normal operations and crashes. The cluster software detects that the database is no longer available and transparently shifts traffic away from the cluster. After the host returns to service an administrator resynchronizes the database and re-enables it in the cluster.

***CS-AVAIL-A-4: Failover to a second site*** All hosts on a single site fail. The site operators initiate failover of applications and database processing to a second site, using replica(s) of databases that have been created from the primary site. This failover can be very fast since the replica databases are already up and do not require recovery.

### 2.1.2 Applicability in Commercial Markets

This use case is the single most important case for commercialization. As mentioned above, availability is a requirement for virtually all customers. In addition, solutions that might otherwise help with this problem tend to have significant draw-backs:

- Shared disks allow rapid transfer of disk resources when databases fail but are expensive and not available in many low to medium size IT environments. They also introduce a single point of failure.

- Disk block replication removes the single point of failure but has problems with recovery and also has a number of practical management and performance drawbacks.
- Many existing replication database replication mechanisms have weak support for avaiability.

The GORDA/H design is particularly applicable for availability as it allows configuration of master/slave replication. GORDA APIs will provide mechanisms to enable database replication to work more effectively to offer increased availability and durability of data (for example by enabling semi-synchronous replication to slaves).

### 2.1.3 Scoping of Solution Sizes

3 GORDA/H nodes is a standard availability solution. Additional nodes may be used to provide off-site replication as discussed in the final use case above.

## 2.2 LAN Replication Case Studies and Use Cases

This section describes a set of use cases that will be used to guide the enumeration of system requirements for GORDA as well as to validate the GORDA architecture. The selected use cases fall into three broad categories. The first category consists of use cases that are defined by a set of industry standard benchmarks. The second category focuses on use cases that have been captured from current and prospective commercial customers for Continuent's uni/cluster, a middleware LAN-based database replication solution. The last category consists of hypothetical use cases for database replication over WAN and are derived from multiple case studies of Continuent's customers and prospective customers.

### 2.2.1 E-Commerce Enterprise, Web Shopping Application

This first case study is founded on an analysis of the TPC-W benchmark from the Transaction Processing Council and represents an E-Commerce enterprise web shopping application. Applications of this type are widely deployed and well understood. An additional advantage of using this case study as part of the GORDA requirements is that we will also be able to run the benchmark at key iteration phases of development and thus can catch any issues that may arise prior to the first demonstration deployments.

The TPC-W benchmark is summarized as follows, on www.tpc.org/tpcw The summary has been directly quoted:

"TPC Benchmark™ W (TPC-W) is a transactional web benchmark. The workload is performed in a controlled internet commerce environment that simulates the activities of a business oriented transactional web server. The workload exercises a breadth of system components associated with such environments, which are characterized by:

- Multiple on-line browser sessions
- Dynamic page generation with database access and update
- Consistent web objects
- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line transaction execution modes
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Transaction integrity (ACID properties)

- Contention on data access and update”

Many Continuent’s customers have TPC-W type of processing needs and they currently have a uni/cluster installation, which manages the replication at middleware layer, in front of the database servers. The GORDA-M version can be utilized with the uni/cluster product yielding a powerful TPC-W workload processing DBMS cluster.

Figure2.2 shows a TPC-W setup with a set of emulated users through the Internet connecting to a cluster of application servers. Application logic connects to GORDA-M enabled uni/cluster middleware, whch manages the data replication in a consistent and scalable way. The middleware can be easily configured to support any number of database replicas attached. In real life, the typical TPC-W setups usually have 2 to 6 database replicas.

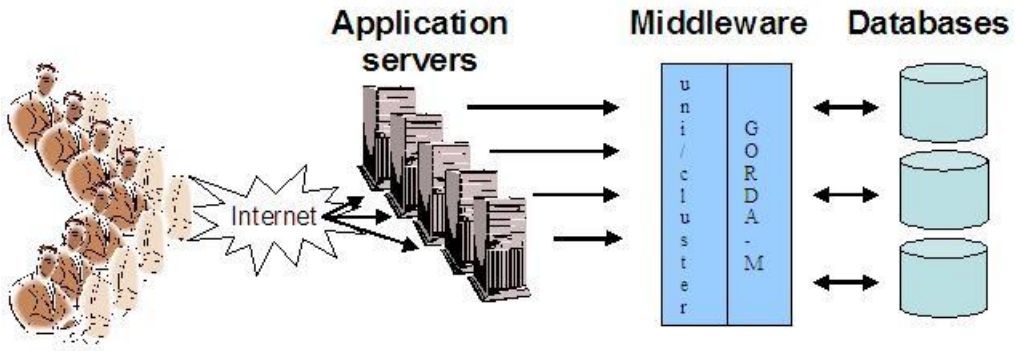


Figure 2.2: Typical TPC-W setup with GORDA-M

Moreover, inspired by the GORDA achievements, Continuent has started a project for a more general clustering framework (Tungsten) — Figure2.3, which enables the integration of GORDA-H directly with Continuent’s clustering solution. With GORDA replication it is possible to increase the performance of the replication significantly and reach customers with more strict requirements on response time, throughput and scalability. This type of clustering can reach the best performance figures both with read and write access.

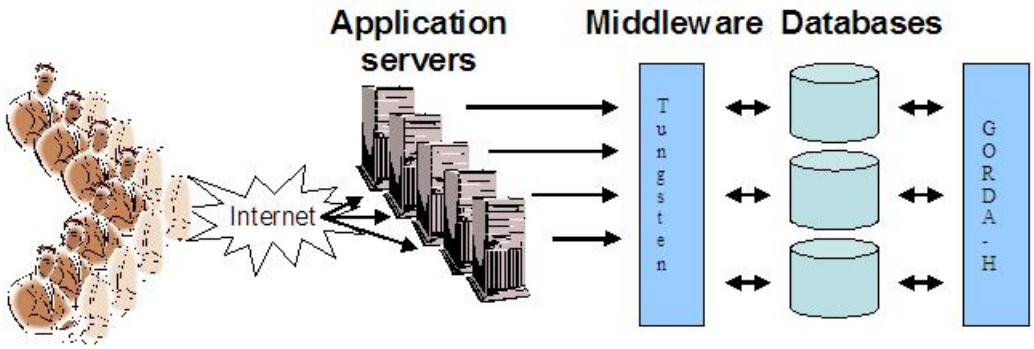


Figure 2.3: Typical TPC-W setup with GORDA-H replication working under Tungsten middleware

Both architectures offer a good platform for processing TPC-W type of workload.

“The performance metric reported by TPC-W is the number of web interactions processed per second. Multiple web interactions are used to simulate the activity of a retail store, and each interaction is subject to a response time constraint.”

”TPC-W simulates three different profiles by varying the ratio of browse to buy: primarily shopping (WIPS), browsing (WIPSb) and web-based ordering (WIPSo). The primary metrics are the WIPS rate, the associated price per WIPS ($/WIPS), and the availability date of the priced configuration.”

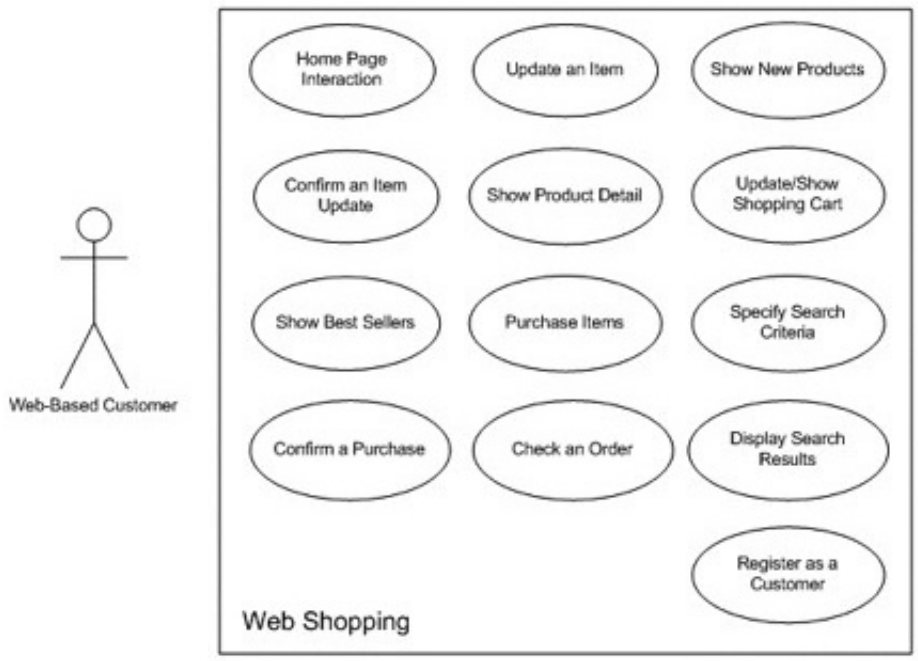


Figure 2.4: Web Shopping Use Cases

**Web Shopping DB Access Use Cases**

The set of use cases which appear in the above diagram and in the following sections have been adapted from the text of the TPC-W specification titled **TPC BENCHMARK ™ W (Web Commerce) Specification**, Version 1.8. Rather than refer the reader to this specification, we have chosen to adapt, summarize, and quote material which appears in this document.

A detailed description of TPC-W can be found by following this link:
www.tpc.org/tpcw/spec/TPCW_V1.8.doc

***CS-WSA-A-1: Home Page Interaction*** “This unprotected web interaction returns to the EB ( Emulated Browser) a web page which contains links to product lists for new products and for best sellers. This is the initial web interaction requested by all Users starting a new User Session. It is also a navigation option from most other web pages.”

***CS-WSA-A-2: Update an Item*** “This unprotected web interaction returns to the EB a web page which allows a User to request the update of an item.”

***CS-WSA-A-3: Confirm an Item Update*** “This unprotected web interaction updates an item and returns to the EB a web page which contains the details of the updated item.”

***CS-WSA-A-4: Specify Search Criteria*** “This unprotected web interaction returns to the EB a web page which allows a User to specify search criteria to find qualifying items.”

***CS-WSA-A-5: Display Search Results*** "This unprotected web interaction returns to the EB a web page which contains the list of items that match a given search criteria."

***CS-WSA-A-6: Show New Products*** "This unprotected web interaction returns to the EB a web page which contains the list of recently released items."

***CS-WSA-A-7: Show Product Detail*** "This unprotected web interaction returns to the EB a web page which contains detailed information on a selected item."

***CS-WSA-A-8: Update/Show Shopping Cart*** "This unprotected web interaction updates the associated CART (refreshing the CART's date and optionally adding a new item or updating existing items) and always returns to the EB a web page which displays the updated contents of the User's CART. If a Shopping Session is not identified at the start of this web interaction, a new Shopping Session is created."

***CS-WSA-A-9: Show Best Sellers*** "This unprotected web interaction returns to the EB a web page which contains the list of best seller items."

***CS-WSA-A-10: Purchase Items*** "This secure web interaction registers a new customer or identifies a returning customer and returns to the EB a web page which displays information about the customer, confirming either the registration or the identification, and displays a summary of the items in the associated CART. The page provides editable fields for entering credit card information and selecting shipping options."

***CS-WSA-A-11: Confirm a Purchase*** "This secure web interaction transfers the content of the associated CART into a newly created order for the registered customer and executes a full payment authorization. It then returns to the EB a web page containing the details of the newly created order."

***CS-WSA-A-12: Look up and Order*** "This secure web interaction returns to the EB a web page which allows a User to provide the information necessary to enter or confirm their identity as a returning customer. This is the first step in displaying information about the customer's last order."

***CS-WSA-A-13: Display an Order*** "This secure web interaction returns to the EB a web page which displays the status of the last order placed by the customer."

***CS-WSA-A-14: Register as a Customer*** "This unprotected web interaction returns to the EB a web page which allows a User to provide the information necessary to register as a known Customer or as a new Customer and to submit their registration. This is the first step in buying the contents of the CART."

**Web Shopping Cluster Operations Use Cases**

The following set of use cases illustrates some typical cluster operations tasks. Because these use cases are cluster specific, they do not appear in the original TPC-W document.

***CS-WSA-O-1: State Transition to Active*** This cluster operations interaction starts when a cluster administrator initiates a state transition for cluster node from the inactive to the active state and terminates when the node has reached the active state and is sharing the processing of read requests with the rest of the active nodes in the cluster.

***CS-WSA-O-2: State Transition to Inactive*** This cluster operations interaction begins when a cluster administrator removes a given cluster node from active operation, performs a memory upgrade, and terminates when the node resumes active operation.

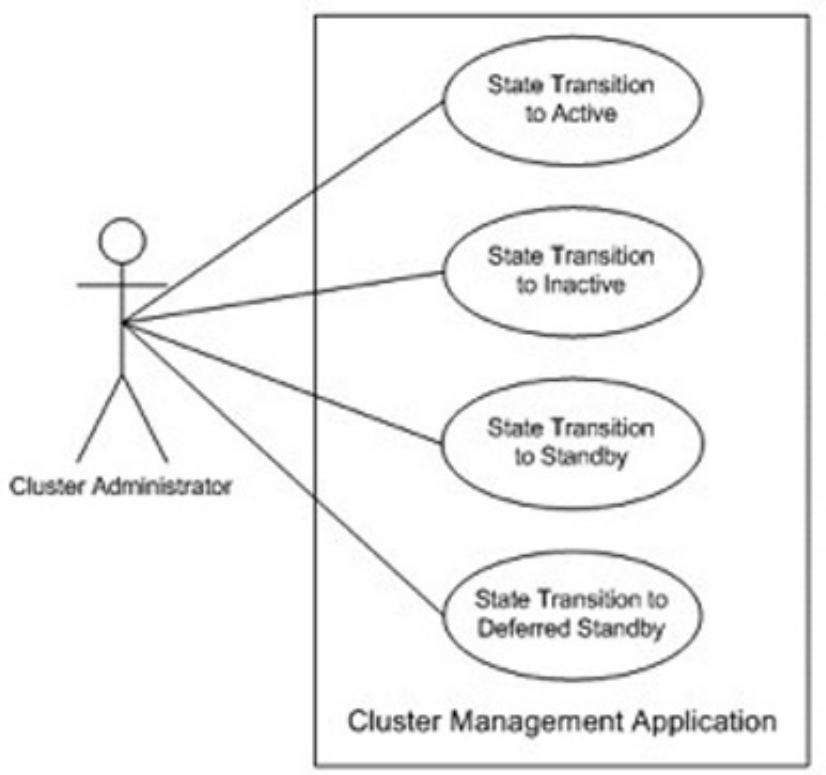


Figure 2.5: Cluster Operations Use Cases

***CS-WSA-O-3: State Transition to Standby*** This cluster operations interaction begins when a cluster administrator initiates an operation to perform a hot backup of all of the databases that are maintained by a given cluster and terminates when the backup has been completed.

***CS-WSA-O-4: State Transition to Deferred Standby*** This cluster operations interaction begins when a cluster administrator initiates a sequence of events that allow for the re-indexing of a given table without any downtime for the cluster and the propagation of the re-indexed table to the other nodes of the cluster.

### Applicability in Commercial Markets

The type of application modeled by the TPC-W benchmark continues to represent a very important usage profile within the overall database market according to Continuent's internal research. Approximately one third to one half of Continuent customers according to a survey in late 2007 were seeking performance increases on database operating within a single site.

Commercial database vendors tend to approach TPC-W application scenarios using large, vertically scaled systems or high-end clustering mechanisms like Oracle RAC. These solutions have the advantage of maximizing transparency for existing applications but are complex solutions with high up-front investment costs. Such solutions are unsuitable for many businesses, especially those experiencing high growth from a small base.

Replication-based database architectures are an important alternative to conventional DBMS architectures to implement TPC-W. Continuent expects business opportunities to continue to grow in this area, especially given that many such applications are relatively new and can accommodate the modest changes required to make replication work effectively.

Both GORDA/M and GORDA/H architectures are applicable for the TPC-W type of application, though there are trade-offs between approaches. For example, middleware solutions have the advantage that they support scaling of reads with minimal application changes. The downside is that write performance is slower and there are more limitations on SQL that can be handled. Master/slave replication requires application changes to use effectively (TPC-W) implements this but handles a wider variety of SQL operations, including changes generated by stored procedures and triggers. At this time we see more commercial interest in master/slave designs of the type that GORDA/H enables.

Over the longer term it is likely that certification-based architectures based on GORDA/H or similar designs will become viable solutions. We are continuing to look closely at this approach as a third way for solving problems involving single-site database scaling. However, there are a number of technical difficulties in handling contending updates as well as DDL that need to be investigated very carefully. It will likely be some time before we see certification-based systems fully deployed.

#### Scoping of Solution Sizes

TPC-W clusters typically require two or more database replicas. Continuent currently certifies its Sequoia-based solutions for up to four database nodes. We would expect to see four nodes continue as a base requirement. Likely configurations would look like the following:

**GORDA/M** 4 database replicas standard. Larger numbers of replicas would be appropriate only in low-write situations, as this approach has relatively weak update performance.

**GORDA/H** 3 or more database replicas scaling to dozens for very large installations with large numbers of reads. Applications with catalogs, for example, tend to be highly read intensive.

4- and 8-core hosts are now commonly available for database hosts, and databases like PostgreSQL use the additional cores quite effectively. This has had the effect of reducing the number of replicas in at least some cases.

### 2.2.2 Telecom Equipment Vendor, LSMS for LNP

#### LSMS Design Overview

Holy acronyms, Batman!! You have entered the telecom zone, but don't worry - The telecom world, in general, is widely known for building redundancy into their hardware and software platforms in order to provide the appropriate 'carrier grade' levels of availability. So database replication is a natural fit when it comes to providing this kind of redundancy on the database tier of any application.

A major telecom equipment vendor provides software and hardware to telephone service carriers that allows them to fully support Local Number Portability (**LNP**). They need a replicated database solution for their Local Service Management System (LSMS). A GORDA-compliant database with 2 or 3 replicas can be used to provide high availability properties to LSMS.

The Local Service Management System (**LSMS**) receives number portability data from any given Number Portability Administration Center (**NPAC**). There are eight regional NPAC in the United States and one in Canada. The LSMS then properly structures that data for use by the company's Service Control Point (SCP) hardware/software platform and transfers the data, as required, to an SCP. The SCP, in turn, uses the data provided to perform the actual telephone network-level operations necessary to provide the LNP service. A description of the full details of this service is beyond the scope of this paper. The following diagram shows the principal architecture of the LNP system.

The LNP databases can be clustered with Tungsten and GORDA-H type of replication system providing a synchronous and fault tolerant data access for LSMS and SCP systems. Having a synchronous

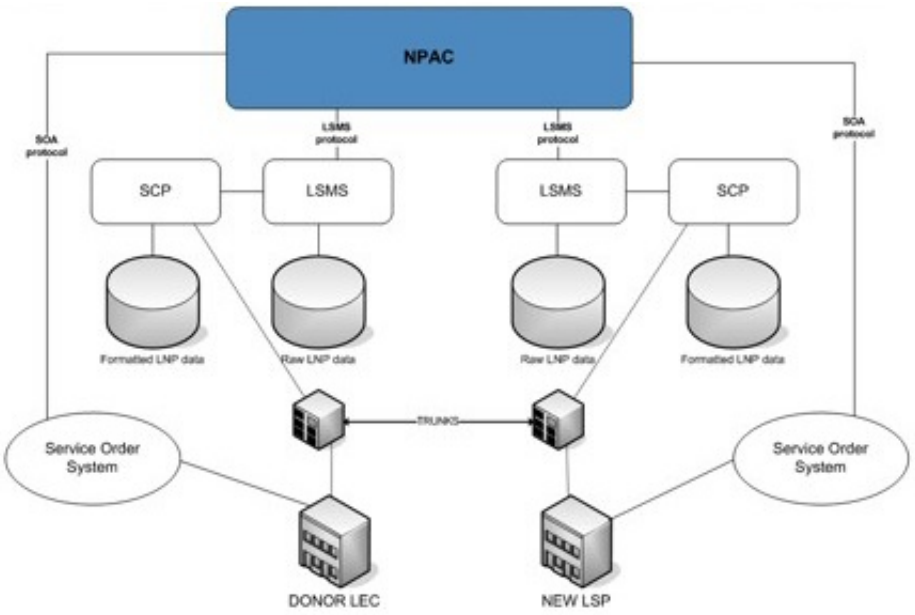


Figure 2.6: LNP Architecture

replication eliminates the need to have manual backup servers, which will automate and ease the management of the DBMS systems. The response time requirements in all the use cases are so strict, that the actual cluster sizes will have to be much bigger than, e.g., with TPC-W type of work load. We estimate that number of replica databases ranges between 4-16 depending on actual load and read/write ratio.

**LSMS DB Access Use Cases**

The following set of use cases all ultimately involve database access and, therefore, are the set of use cases which will drive requirements for database replication, high availability, performance and fault monitoring, etc.

***CS-LSMS-A-1: Get LNP Data*** LSMS gets 25 LNP records/second per NPAC and there is a single connection to each of 8 NPAC

1. each LNP record is a max of 500 bytes
2. it takes 2 database reads and 2 writes to process each record

***CS-LSMS-A-2: Audit LNP Data*** The SCP executes a request to the LSMS which, in turn, executes a query that scans a range of records in the LNP database and compares this data to the data that it has to verify that the local SCP data is correct.

***CS-LSMS-A-3: Distribute LNP Data*** The LSMS coordinates the distribution of a set of formatted LNP records to a given SCP.

**LSMS Cluster Operations Use Cases**

***CS-LSMS-O-1: Manual Failover*** The LSMS Administrator uses a management interface to command the LSMS to switch from the current active cluster node to the hot-standby cluster node. How-

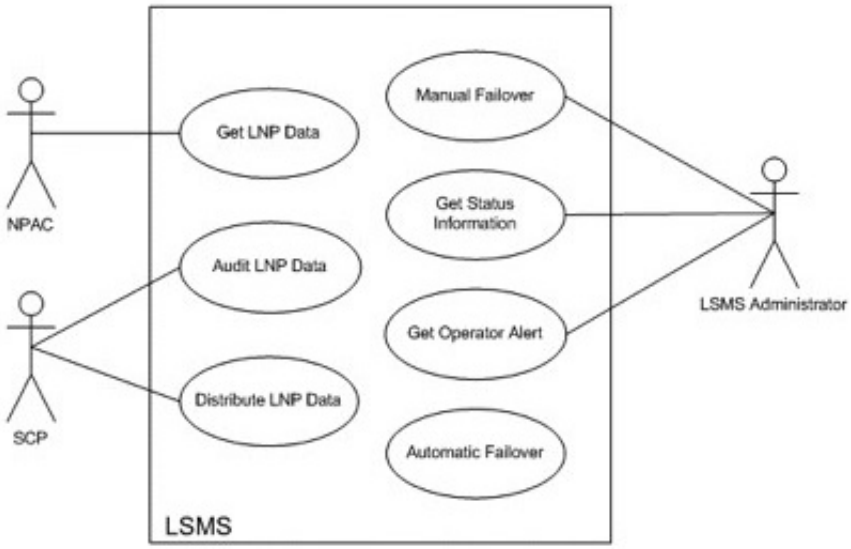


Figure 2.7: LSMS DB Use Cases

ever, with a GORDA compliant replicated database, such failover would be fully transparent and automated.

***CS-LSMS-O-2: Get Status Information*** The LSMS Administrator requests summary status information on the operation of the LSMS.

***CS-LSMS-O-3: Get Operator Alert: Database Server Exit*** A fault detection component of the cluster detects that the database server process has exited. The appropriate node state transition is initiated and an explicit alert is sent to the LSMS operator.

### Applicability in Commercial Markets

The LSMS use case originally seemed more interesting than is currently the case in early 2008. There are two main reasons for this.

First of all, the database landscape changed with the introduction of MySQL Cluster, which uses a proprietary clustering design to provide highly available databases based on multiple, in-memory copies of data partitioned across different hosts. MySQL Cluster was relatively immature at the time MySQL acquired the technology from Ericsson and had a number of technical limitations that made it difficult to use easily. MySQL has improved the product significantly and markets it as a solution for carrier greade applications like LSMS. The in-memory design is well suited for these applications, as it allows very rapid processing of transactions that consist of short, simple reads and writes. MySQL Cluster is therefore very well suited for Telco transactions of the sort described here.

Second, at Continuent we have found that fail-over for Telco applications is in fact a very challenging problem that clusters based on ordinary databases do not handle well. The most difficult problems occur when one of the databases loses an interface or crashes completely. It is difficult in this case to avoid system hangs of a minute or more before network timeouts take effect, which then violate application SLAs. Solving this problem requires specialized network configuration and support for back-channel pings built into the database itself.

We do not anticipate that this case will be a significant driver of commercial replication-based design in future. It represents a relatively small proportion of the market compared in particular to TPC-W, basic

availability, and WAN use cases.

### 2.2.3 Brick and Mortar Enterprise, OLTP Application

#### Overview

This case study is founded on an analysis of the TPC-C benchmark from the Transaction Processing Council (tpc.org) and represents an OLTP application that can typically be found in any large enterprise that sells products or services. Applications of this type have been widely deployed and well understood for several decades. As in the TPC-W based case study, an additional advantage of using this case study and set of use cases as part of the GORDA requirements gathering and validation activity is that we will also be able to run the benchmark at key iteration phases of development and thus can catch any issues that may arise prior to the first demonstration deployments.

A detailed description of TPC-C can be found by following this link: www.tpc.org/tpcc/detail.asp. Figure 2.8 shows a typical TPC-C setup where the number of database replicas can vary from 2 to 6 databases.

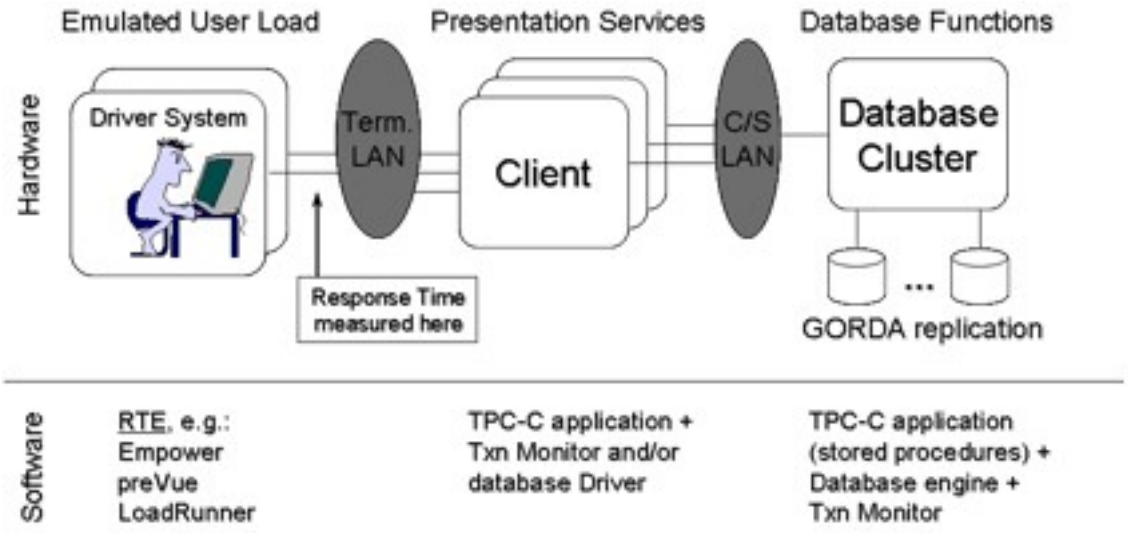


Figure 2.8: Typical TPC-C setup

"Approved in July of 1992, TPC Benchmark C is an on-line transaction processing (OLTP) benchmark. TPC-C is more complex than previous OLTP benchmarks such as TPC-A because of its multiple transaction types, more complex database and overall execution structure. TPC-C involves a mix of five concurrent transactions of different types and complexity either executed on-line or queued for deferred execution. The database is comprised of nine types of tables with a wide range of record and population sizes. TPC-C is measured in transactions per minute (tpmC)."

"TPC-C simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centered around the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service."

**OLTP DB Access Use Cases**

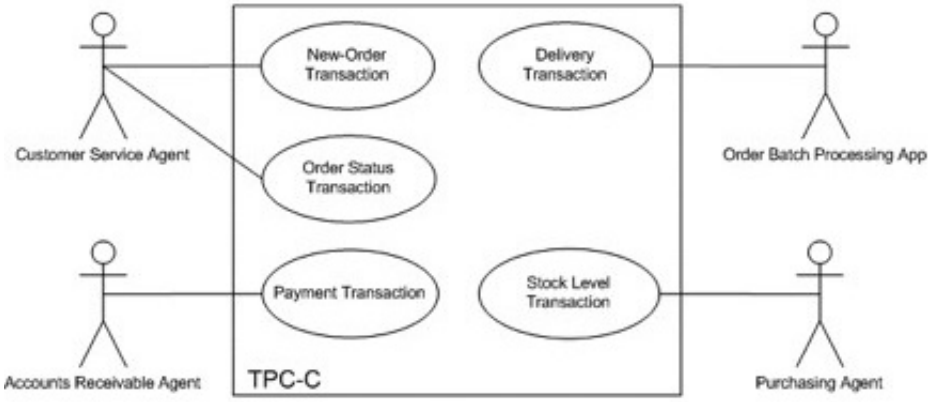


Figure 2.9: OLTP DB Access Use Cases

The set of use cases which appear in the above diagram and in the following sections have been adapted from the text of the TPC-C specification titled **TPC BENCHMARK™ C, Standard Specification**, Revision 5.4 Rather than refer the reader to this specification, we have chosen to adapt, summarize, and quote material which appears in this document.

***CS-OLTP-A-1: New-Order Transaction*** The New-Order business transaction consists of entering a complete order through a single database transaction. It represents a mid-weight, read-write transaction with a high frequency of execution and stringent response time requirements to satisfy on-line users. This transaction is the backbone of the workload. It is designed to place a variable load on the system to reflect on-line database activity as typically found in production environments.

***CS-OLTP-A-2: Order Status Transaction*** The Order-Status business transaction queries the status of a customer's last order. It represents a mid-weight read-only database transaction with a low frequency of execution and response time requirement to satisfy on-line users. In addition, this table includes non-primary key access to the CUSTOMER table

***CS-OLTP-A-3: Payment Transaction*** The Payment business transaction updates the customer's balance and reflects the payment on the district and warehouse sales statistics. It represents a light-weight, read-write transaction with a high frequency of execution and stringent response time requirements to satisfy on-line users. In addition, this transaction includes non-primary key access to the CUSTOMER table.

***CS-OLTP-A-4: Delivery Transaction*** The Delivery business transaction consists of processing a batch of 10 new (not yet delivered) orders. Each order is processed (delivered) in full within the scope of a read-write database transaction. The number of orders delivered as a group (or batched) within the same database transaction is implementation specific. The business transaction, comprised of one or more (up to 10) database transactions, has a low frequency of execution and must complete within a relaxed response time requirement.

The Delivery transaction is intended to be executed in deferred mode through a queuing mechanism, rather than interactively, with terminal response indicating transaction completion. The result of the deferred execution is recorded into a result file.

***CS-OLTP-A-5: Stock Level Transaction*** The Stock-Level business transaction determines the number of recently sold items that have a stock level below a specified threshold. It represents a heavy read-only database transaction with a low frequency of execution, a relaxed response time requirement, and relaxed consistency requirements.

**Applicability in Commercial Markets**

The TPC-C use case can be handled using replication designs as described in this section. However, TPC-C shares two important characteristics of OLTP applications that affect the applicability of particular replication approaches:

- TPC-C has a relatively high update rate
- TPC-C transactions tend to conflict, which means that they hold competing row-level locks that block transactions and may lead to deadlocks

These properties mean that state-machine and certification based approaches are unsuitable because of their lower update rates as well as vulnerability to conflicting transactions. GORDA/M and GORDA/H solutions need to use master/slave replication to get around these problems, which in turn means that application changes are necessary.

As a result, it appears that conventional DBMS solutions using a single vertically scaled DBMS server will continue to dominate in these types of applications. From a commercial point of view it is important to be able to support these cases, but they will probabl remain less important than both TPC-W and basic availability.

## 2.3 WAN Replication Case Studies and Use Cases

### 2.3.1 Continuent's WAN Replication

We currently have one reference case study in this section and, for the present, it is representative enough of the most prominent set of requirements in this area.

Many Continuent's customers have been asking for a WAN replication system, mostly for disaster recovery purposes. Based on these demands, Continuent has started a project for implementing a WAN replication system, clustering local GORDA clusters. The WAN replication will be asynchronous in nature, with a single master component.

Following picture (Figure 2.10)shows the high level architecture of the WAN replication system.

### 2.3.2 FSecure Subscription Authorization and Profile Application

The application under study here is used by FSecure, a company that provides a wide range of computer and network security products via subscriptions. In this application, a central or **master** database, hosted at a single network operations center (NOC) maintains the subscription authorization and profile information for all of the customers for a specific security product. Data from this primary database, as well as the subscription content, is then propagated to a set of **backup** subscription servers which are hosted at a number of different NOC in geographically diverse areas. When an FSecure customer starts the FSecure software that is resident on their computer, the software initiates a set of operations with one of these backup replicas. For example, the software checks to make sure that the license is valid and whether or not any new content (virus definitions, for example) is available.

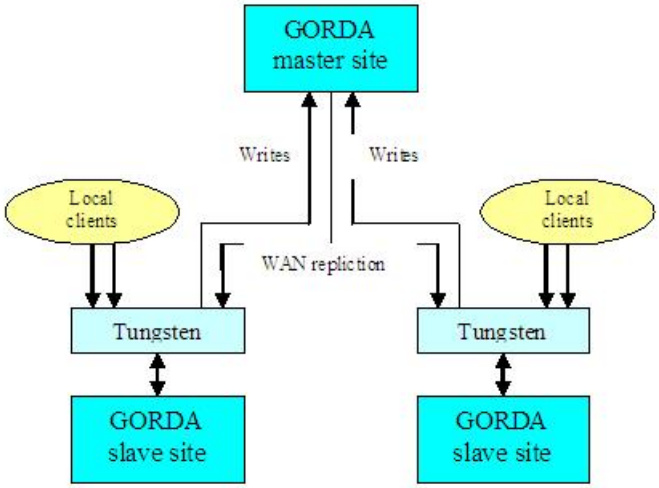


Figure 2.10: WAN Architecture

In the FSecure system architecture, accesses from the subscription clients is made via a BackWeb client application that resides on the customer's machine to a BackWeb host that is part of each subscription server. If database access is needed, the BackWeb (BW) host invokes a BW hook which is, essentially, a means for making access to a database via ODBC.

Current plans specify that the primary subscription database requires high availability and it's anticipated that that high availability will be provided by some sort of a LAN replication based cluster. A GORDA compliant replicated database would use 4 to 8 replicas. For the backups, there will generally be only a single database node unless the number of subscriptions managed by that node exceeds some threshold, in which case LAN replication clustering of 2 to 3 replicas may be used to provide a degree of read scalability.

### F-Secure Subscription Service DB Access Use Cases

The use cases described in the following sections involve direct access to the data stored in the primary database and may result in replication activity to the backup databases.

***CS-SUBADMIN-A-1: Add Subscription*** The subscription administrator uses a GUI to provide the input data for a new subscription. This adds a new row to the SUBSCRIPTION table.

When a new subscription is added, the subscription data must be immediately propagated to all of the backup servers.

***CS-SUBADMIN-A-2: Change Subscription*** The subscription administrator makes a change to the subscription data. For example, a subscription can be cancelled. This updates fields in the SUBSCRIPTION table and adds a new row to the SUBSCRIPTION_HIST table.

When a subscription is changed, the appropriate changes must be immediately propagated to all of the backup servers.

***CS-SUBADMIN-A-3: Reactivate Subscription*** The subscription administrator resets a subscription to its initial state. This updates fields in the SUBSCRIPTION table, adds a row to the SUBSCRIPTION_HIST table, removes one or more rows from the BWINSTALLED table (if there is one) and adds one or more associated rows to the BWINSTALLED_HIST table.

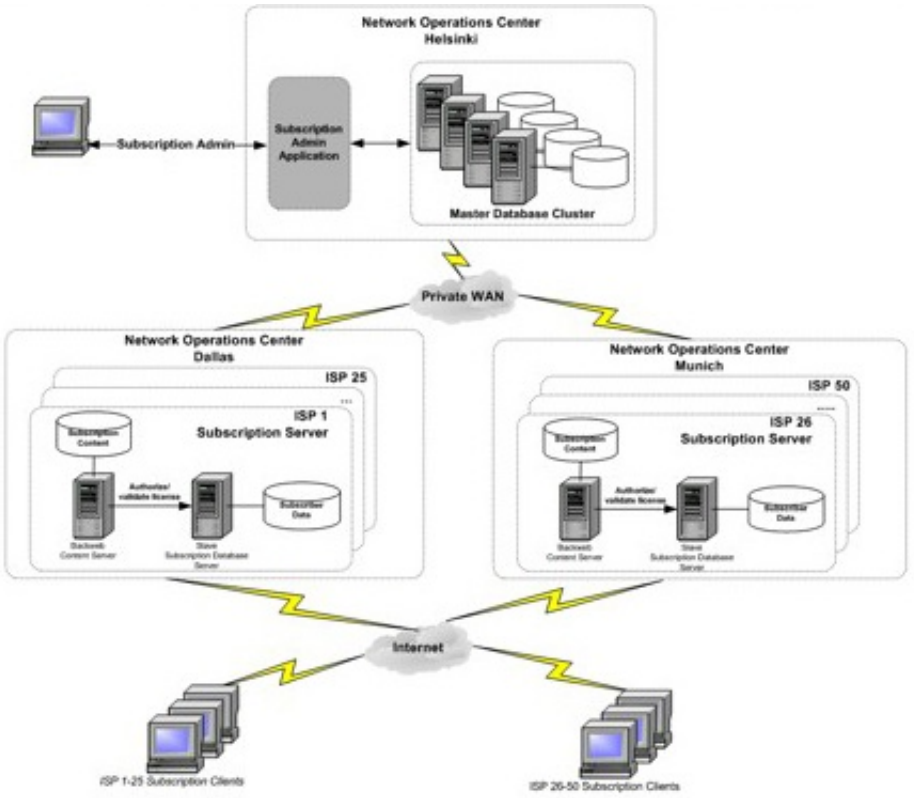


Figure 2.11: Subscription Services Architecture

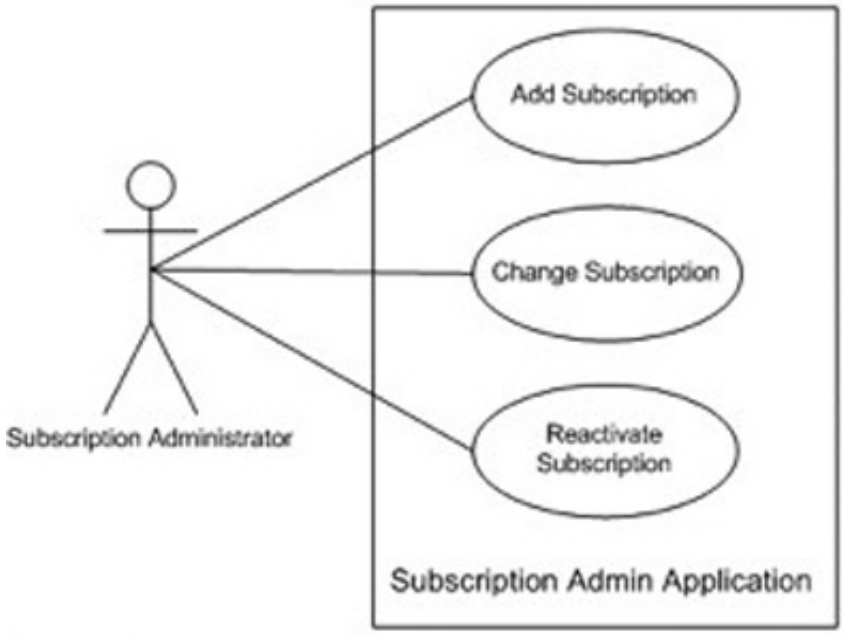


Figure 2.12: Subscription Administration Use Cases.

When a subscription is reactivated, the appropriate changes must be immediately propagated to all of the backup servers.

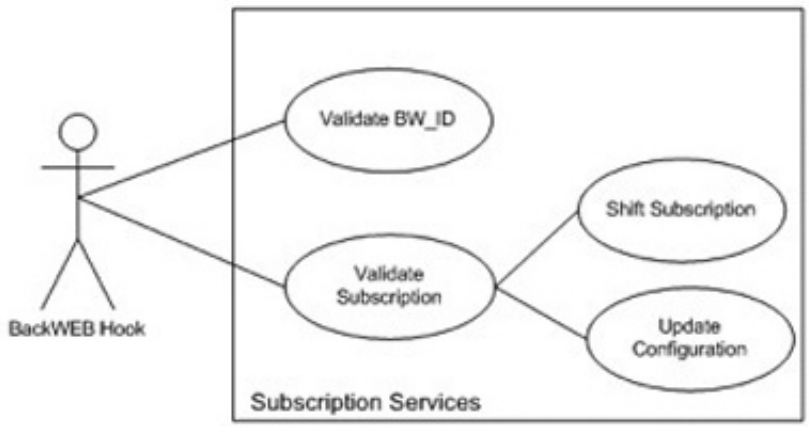


Figure 2.13: Subscription Use Cases.

**NOTE:** A BackWEB (BW) Hook is, essentially, an application that is called by the BW server and that uses ODBC to connect to a local database to perform a specific task. For the purposes of the following use cases, the word ***HOOK*** will be used to refer to BW Hook.

***CS-SUBSVC-A-1: Validate BW_ID*** Given a BW_ID, a BW Server uses the HOOK to look up the BW_ID in the BWINSTALLED table. The status is returned as appropriate i.e. NOT_FOUND or a valid SUBSCRIPTION_ID.

***CS-SUBSVC-A-2: Validate Subscription*** Given a SUBSCRIPTION_ID, a BW Server uses the HOOK to check whether the subscription exists and is up to date. If the subscription is found, some status fields are updated in the BWINSTALLED table including, if necessary, fields associated with a configuration change on the client. If there was a configuration change on the client, one or more events get added to the EVENTS table.

***CS-SUBSVC-A-3: Shift Subscription*** If the subscription ID has changed, the HOOK attempts to find the new subscription and the BWINSTALLED, BWINSTALLED_HIST, EVENT, EVENT_HIST and SUBSCRIPTION_HIST tables have rows updated, added or deleted.

***CS-SUBSVC-A-4: Propagate Configuration and Host Changes*** Based on some pre-defined interval, information about changes that have been made on subscriber machines as well as statistical information collected on each local, backup, server must be propagated to the primary. In this use case, it is assumed that the rule is for there never to be a case where data from the same subscription client is present on more than one subscription server (backup server). This fact, in turn, prevents complexity in logic to ensure that updates are never lost.

### 2.3.3 Applicability in Commercial Markets

As of early 2008 the WAN replication case is among the most significant use cases for replication-based clustering architectures. Database replication is one of the few technologies that solves the problem of spreading live data across multiple database systems. Other approachs, such as replication using application level messages, require costly and difficult changes to applications. They also create awkward recovery problems if improperly implemented.

In addition, there is tremendous commercial demand for such solutions as businesses expand from single site operation to multiple sites in order to solve problems with availability and to raise performance by spreading transactions across additional sites. Since the original use cases were published, we have analyzed and confirmed the applicability of the F-Secure architecture to the clusters used by the US National Weather Service, a current Continuent customer. We have also confirmed prospects for multi-site architectures in dozens of prospective customers.

GORDA/H nodes configured with master/slave replication are a key part of solving this problem. In summary, we see this as one of the key areas of growth commercially and (we hope) a fruitful area for future research.

# Chapter 3

# Requirements

The following sections provide the full set of requirements for a GORDA-based system that is capable, in turn, of successfully executing the use cases outlined in the previous sections.

In what follows, we define some acronyms used throughout our explanation.

**hybrid** used in the context of this paper, the word hybrid refers to a model for GORDA that includes both a generic, high-availability framework and replication protocols as well as a modified version of a target DBMS which, through cooperative execution, meets the GORDA requirements.

**GORDA-H** refers to the GORDA hybrid model.

**middleware** when used in conjunction with the term GORDA, middleware refers to a model for **GORDA** which includes a generic, high-availability framework and replication protocols but does NOT rely on any modifications, whatsoever, to the target DBMS.

**GORDA-M** refers to the GORDA middleware model.

## 3.1 Application Transparency

Application transparency can be defined, in practical terms, as follows:

- Any application that will run against a given DBMS shall also run, without alterations, against a GORDA compliant system.
- Because GORDA will, undoubtedly, involve clustering technology, the requirements outlined in this section step a little in the direction of design details.

### 3.1.1 SQL Transparency

There are essentially three broad categories into which SQL statements can be categorized: **DML**, **DDL**, and **DBMS ADMIN commands**. The transparency level required for commands from each of these categories may vary since not all statements have equal importance in the real world nor can all commands be supported transparently under all conditions.

- GORDA shall provide 100% transparency for all DML commands and DDL commands. The only exception to this is that stored procedures shall not be supported by the GORDA middleware implementation.

- GORDA shall provide as much transparent support for DBMS maintenance and administration commands as possible. A good example of this, in MySQL, would be to support the 'kill' command, which kills a thread, in such a way that threads are killed across all nodes in the cluster if appropriate. This may be necessary, for example, if a thread is found to be holding locks etc.
- GORDA shall provide for local execution of some DBMS DDL, maintenance and administration commands as an option to transparency. An example of this would be a command that rebuilds an index on a table: a user may not want to tie up the table on all of the nodes, but only on a single node, and propagate the changes to the other nodes when the re-indexing is completed.

### 3.1.2 Concurrency/Lock Granularity

- GORDA-H shall support the most fine-grained level of locking possible, and, ideally, will transparently show through the locking granularity present in the underlying DBMS.
- GORDA-M shall provide a configurable range of alternatives for providing the highest levels of concurrency for a given workload at the middleware level without requiring any changes to the target DBMS. Specifically:
  - one-write-operation-operation-at-a-time, which means that the replicas are updated per write-operation instead of waiting to commit statement.
  - table-level locking
  - predicate locking

### 3.1.3 User Accounts, Authorization and Permissions Transparency

- GORDA shall provide 100% transparency for all facets of user account administration, user account-based permissions etc. In practice, this means that whatever security constraints are enforced with a single instance of a given DBMS for a given set of applications must also be enforced, identically, by GORDA.
- GORDA shall provide transparent support for client application authorization protocols.
- GORDA shall provide for application connectivity in such a way that the application does not need to be aware that there is more than one DBMS in operation. For instance, in a cluster, a typical solution to this requirement is to implement a single ***virtual IP address*** for the entire cluster.

### 3.1.4 Error Status Transparency

- GORDA shall propagate DBMS specific error status information transparently to client applications i.e. error status semantics must be maintained.

### 3.1.5 Connection Status Transparency

- GORDA shall propagate any fatal errors which would result in the loss of client application connectivity to the application as normal. This requirement explicitly rules out automatic, application-transparent failover in the case of lost connectivity.

### 3.1.6 Client API Transparency

- GORDA shall provide client APIs that duplicate or are identical to native DBMS APIs for clients. In the ideal case, native client applications shall be able to connect without library or connection string alterations. Failing this, native clients shall connect using libraries that faithfully reproduce native client library behavior.

## 3.2 Database Consistency Criteria

- GORDA replication protocols should be able to provide 1-copy-equivalence with respect to the underlying DBMS consistency criterion. Complementary, weaker criteria, justifiably tolerated by the application and ensuring bounded divergence, should be provided whenever allowing to increased the system performance.
- GORDA shall ensure that ACID properties inherent in the underlying DBMS are fully supported. GORDA does not need to exhibit ACID properties for operations which do not already provide for them in the underlying DBMS.

## 3.3 Performance and Scalability

### 3.3.1 Read Scalability

- GORDA shall provide a means to achieve scalability for read operations.
- Given a steady, consistent workload composed of 100% read activity, GORDA shall provide for an increase in read throughput at the same average response time.

### 3.3.2 Connection Scalability

- GORDA shall provide a means to achieve scalability for the number of concurrent client connections that can be handled by a given GORDA system. The actual level of scalability for connections will depend on the read/write ratio of the workload being processed and, potentially, on whether or not the workload involves the execution of multi-statement transactions.

### 3.3.3 Load Balancing

- GORDA shall provide a means to distribute the total system workload across a plurality of processing elements.

## 3.4 WAN Support

### 3.4.1 Primary/Backup Asynchronous Replication

- GORDA shall support asynchronous replication, over a WAN, in a primary/backup configuration. In this configuration, all writes are applied against the primary GORDA system instance and writes are then propagated from the primary GORDA system instance to one or more backup GORDA system instances.
- GORDA primary/backup lazy replication shall be recoverable in the face of network outages, system faults etc.

### 3.4.2 Replication with Synchronous Semantics

- GORDA shall support eager replication over a WAN. In this configuration, write operations may be applied at any GORDA system instance and are propagated, by GORDA, to all other GORDA system instances.
- GORDA eager replication shall exhibit one-copy-equivalence for all operations.

### 3.4.3 SSL Support

- GORDA shall support SSL encrypted WAN links.

### 3.4.4 System Management

- GORDA system management shall support management of all GORDA instances, including those present at disparate geographical locations, from a single access point.

## 3.5 Security

### 3.5.1 SSL Support

- GORDA shall transparently support secure links for all data replication operations over SSL.

## 3.6 DBMS Support

- Both GORDA-H and GORDA-M shall support the PostgreSQL 8.x and Apache Derby 10.x.

## 3.7 Supportability and Testability

### 3.7.1 Fine-Grained Dynamic Diagnostics

- GORDA shall provide a means for dynamically enabling and disabling the collection, distribution and display of system diagnostic information on a very fine-grained level in two domains:
  - within the source code itself, down to the level of an specific method invocation
  - within the running system itself, down to the level of components running on a specific node or set of nodes.

### 3.7.2 Fine-Grained Dynamic Test Hooks

- GORDA shall provide a means of dynamically enabling and disabling a wide range of built-in testing facilities including but not limited to:
  - fault simulation traps
  - process synchronization barriers
  - inter-process synchronization primitives

### 3.7.3 System Diagnostic Dump Facility

- GORDA shall collect, at runtime, using the most efficient means possible, a sufficient quantity and quality of data to enable a trained engineer to get a high level view of an entire running system at the time of a system fault.
- GORDA shall dump the collected system fault data, under all conditions where it is possible, to a flat file or other non-volatile storage.

## 3.8 System Management

### 3.8.1 Centralized Management

- GORDA shall provide for the management and monitoring of all aspects of the cluster from a single reference point.
- GORDA shall provide system management facilities that can manage one or more GORDA system instances from a central point.

### 3.8.2 Text-Based Socket Interface

- GORDA system management shall provide a text based interface accessible via a standard protocol.

### 3.8.3 Java API

- GORDA system management shall provide a Java API that can be used by third party vendors and customers to create/integrate management tools etc. on top of GORDA system management facilities.

### 3.8.4 SNMP Traps

- GORDA system management shall provide SNMP version 1.0 traps where appropriate.

### 3.8.5 Extensive Statistics

- GORDA system management shall provide a high level of detail for statistical information for a running GORDA system.

### 3.8.6 Dynamic Configuration

- GORDA system management shall provide a means for changing system configuration parameters both dynamically and statically.

### 3.8.7 Integrated Database Configuration

- GORDA system management shall provide an integrated means for the configuration of both the GORDA system as well as of the underlying DBMS such that the underlying DBMS is always assured of being configured correctly in support of a given GORDA system configuration.

## 3.9 Failure Handling

### 3.9.1 Automatic Failover

- GORDA system software shall remove failed nodes and transparently route SQL requests away from them. (In the absence of external load-balancing, full transparency may only be supported in GORDA-M.)

## 3.10 GORDA System Software Upgrades

### 3.10.1 Zero Downtime Field Upgradeable

- GORDA system software shall be field upgradeable without causing any system downtime.

### 3.10.2 Rolling DBMS Upgrades

- GORDA system software shall support the rolling upgrade of the underlying DBMS version without causing system downtime.

## 3.11 Network Equipment Requirements/Compatibility

- GORDA shall be capable of operating in any standard internetworking environment with layer 2 or layer 3 switching.
- GORDA shall not require any special models of switches or routers in order to operate correctly.