



Project no. 004758

GORDA

Open Replication of Databases

Specific Targeted Research Project

Software and Services

Deployment Plan

GORDA Deliverable D7.1

Due date of deliverable: 2008/03/31

Actual submission date: 2008/04/21

Resubmission date: 2008/05/28

Start date of project: 1 October 2004

Duration: 42 Months

UMINHO

Revision 1.1

Project co-funded by the European Commission within the Sixth Framework		
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

José Pereira, U. Minho
Luís Rodrigues, U. Lisbon
Nuno Carvalho, U. Lisbon
Rui Oliveira, U. Minho
Fernando Pedone, USI
Emmanuel Cecchet, Continuent
Sara Bouchenak, INRIA



(C) 2007 GORDA Consortium. Some rights reserved.

This work is licensed under the Attribution-NonCommercial-NoDerivs 2.5 Creative Commons License.

See <http://creativecommons.org/licenses/by-nc-nd/2.5/legalcode> for details.

Abstract

This document describes the prototype setup, experimental setting, and evaluation procedure used to demonstrate GORDA deliverables as an integrated prototype in a realistic setting. This is done using the workload of the industry standard TPC-C benchmark and three different concrete scenarios.

Contents

1	Introduction	3
1.1	Objectives	3
1.2	Relationship With Other Deliverables	3
2	Additional Components	4
2.1	Workload Generator	4
2.2	Console Configuration	6
3	Plan	10
3.1	Scenario 0: Basic System	10
3.1.1	Goals	10
3.1.2	Pre-requisites	10
3.1.3	Deployment	10
3.1.4	Usage	11
3.2	Scenario 1a: Complete System and Fault-Tolerance	12
3.2.1	Goals	12
3.2.2	Pre-requisites	13
3.2.3	Deployment	13
3.2.4	Usage	13
3.2.5	Failure and recovery	14
3.3	Scenario 1b: Complete System and Adaptation	15
3.3.1	Goals	15
3.3.2	Pre-requisites	15
3.3.3	Usage	15
3.4	Scenario 2: Realistic Workload	16

3.4.1	Goals	16
3.4.2	Pre-requisites	16
3.4.3	Usage	17
3.4.4	Evaluation	17
3.5	Scenario 3: Legacy DBMS	18
3.5.1	Goals	18
3.5.2	Pre-Requisites	18
3.5.3	Deployment	19
3.5.4	Usage	19

Chapter 1

Introduction

The GORDA Prototype of the Integrated System provides an extensive framework on which to build replication database management solutions. It encompasses multiple configurations for different application and environment scenarios. In Chapter 2 we describe a realistic workload and measurement tools to demonstrate such prototype; and Chapter 3 provides detailed deployment and usage plan to achieve such demonstration.

1.1 Objectives

The goal of this document is to describe in detail deployments of the GORDA Prototype of the Integrated System used for demonstration sessions as well as dissemination activities occurring as described in Table 1.1.

Event	Date	Location	Scenarios
SafeComp Ws. on Software Reliability	2007/09/18	Nurnberg, Germany	S1a
Project Review Meeting 3	2007/10/05	Brussels, Belgium	S0,S1a,S2
Project Review Meeting 4	2008/05/30	Braga, Portugal	S1b, S3

Table 1.1: Demonstration and dissemination activities using the proposed scenarios.

1.2 Relationship With Other Deliverables

This deliverable describes the process of demonstrating GORDA technology and thus is based on D5.2 (Prototype of the Integrated System). The demonstration itself is D7.2 (Demonstration).

Chapter 2

Additional Components

This chapter discusses the workload used in the GORDA demonstration and how the console is configured to obtain performance and resource usage measurements.

2.1 Workload Generator

The workload generator is based on the TPC-C benchmark, that proposes a wholesale supplier with a number of geographically distributed sales districts and associated warehouses as an application. This environment simulates an OLTP workload with a mixture of read-only and update intensive transactions listed in Table 2.1. Namely, *delivery* transactions are CPU bound. *payment* transactions are prone to conflicts by updating a small number of data items in the *Warehouse* table. Finally, *payment* and *orderstatus* execute some code conditionally. *neworder* and *payment* transactions account each for 44% of submitted transactions.

TPC-C mandates that the database size is configured for each simulation run according to the number of clients as each warehouse supports 10 emulated clients [1]. This ensures a realistic amount of conflicting updates and I/O requirements and is used in Section 3.4. The workload generator can also be used in a mode that does not enforce standard TPC-C scaling guidelines, as explained in Section 3.2. The sizes of database tables in each of these scenarios is shown in Table 2.2.

The workload generator component spawns multiple threads using a pool of connections spread evenly across active database replicas. This is done by assigning each five emulated clients associated with the same TPC-C warehouse to a replica. When a replica is disabled, connections are reallocated to remaining active replicas using the same procedure. Upon start of

Transaction	Probability	Description	Read-only?
New Order	44%	Adds a new order into the system.	No
Payment	44%	Updates the customer's balance, district and warehouse statistics.	No
Order Status	4%	Returns a given customer's latest order.	Yes
Delivery	4%	Records the delivery of orders.	No
Stock Level	4%	Determines the number of recently sold items that have a stock level below a specified threshold.	Yes

Table 2.1: Transaction types in TPC-C.

Relations	Number of items		Tuple size
	Standard (150 cli.)	Light	
Warehouse	15	4	89 bytes
District	150	20	95 bytes
Customer	450000	2000	655 bytes
History	450000	3562	46 bytes
Order	450000	3455	24 bytes
New Order	135000	662	8 bytes
Order Line	3824013	31395	54 bytes
Stock	1500000	40	306 bytes
Item	100000	10	82 bytes
Total	2.3 GByte	40 MByte	

Table 2.2: Size of tables in TPC-C and in the “light” scenario.

a replica, load is re-balanced accordingly. The workload generator is monitored and controlled using a JMX interface.

2.2 Console Configuration

The web-based management console configured for the demonstration scenarios offers the following capabilities:

- A global view of the cluster (Figure 2.1) that provides:
 - performance measurements obtained by the workload generator presented as charts;
 - buttons to remove and generate the initial database;
 - buttons to initiate, pause, and stop the workload;
 - a button to check replica consistency, by checksumming the content of each replica;
 - a button for each replica, showing status (green for operational, red for inactive), and opening the respective replica view and control window.
- A replica control panel (Figure 2.2), containing:
 - buttons to start and stop the replica;
 - buttons to initialize the group and to query group membership status (Figure 2.3).
- A replica information panel, containing four different tabs:
 - information about the replication server (Figure 2.4), including size of internal queues, transaction rates, and number of locally connected clients;
 - information about the group communication protocol (Figure 2.5), showing message and byte throughput;
 - information about the system resources (Figure 2.6), showing CPU, memory, I/O bandwidth, and network bandwidth.
 - the message log.

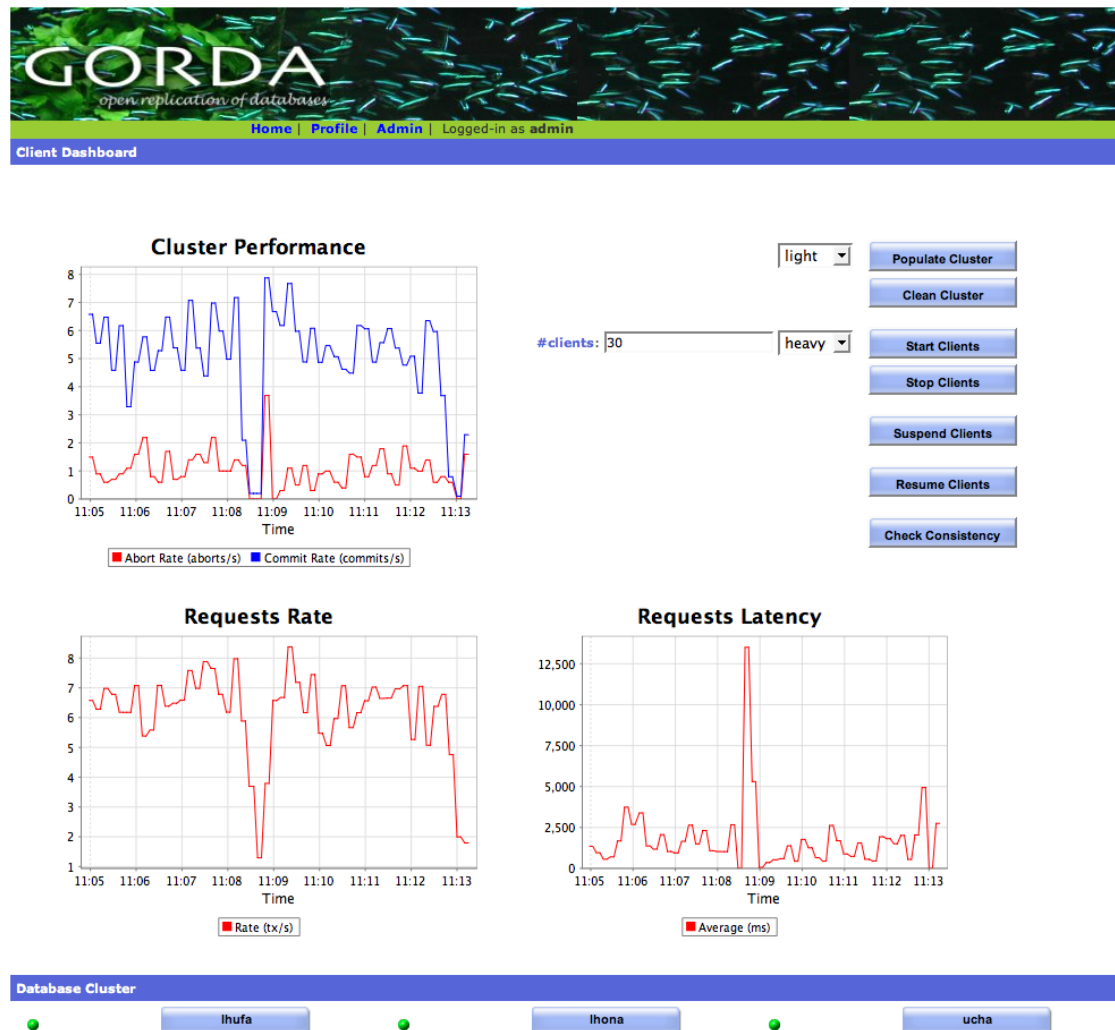


Figure 2.1: Cluster view and workload generator control panel.

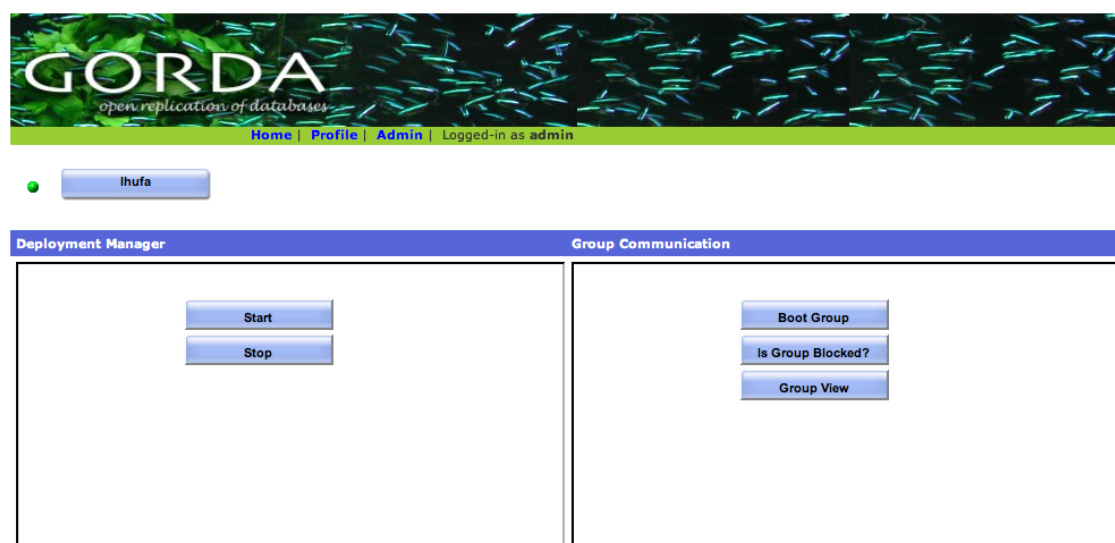


Figure 2.2: Replica view: Control panel.

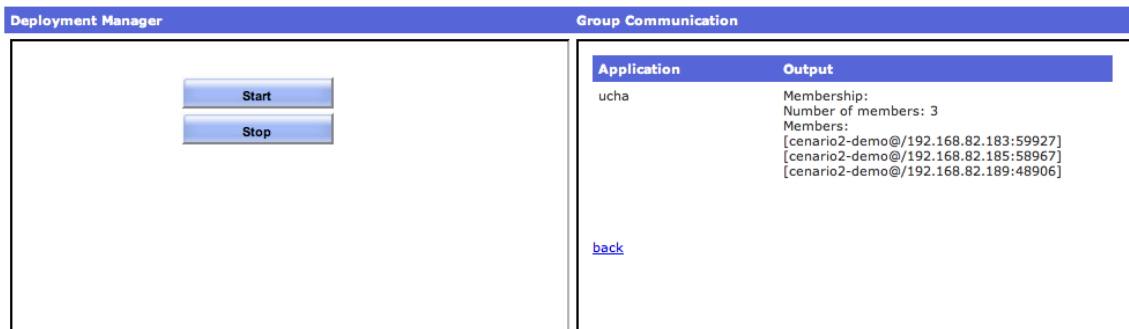


Figure 2.3: Replica view: Group membership.

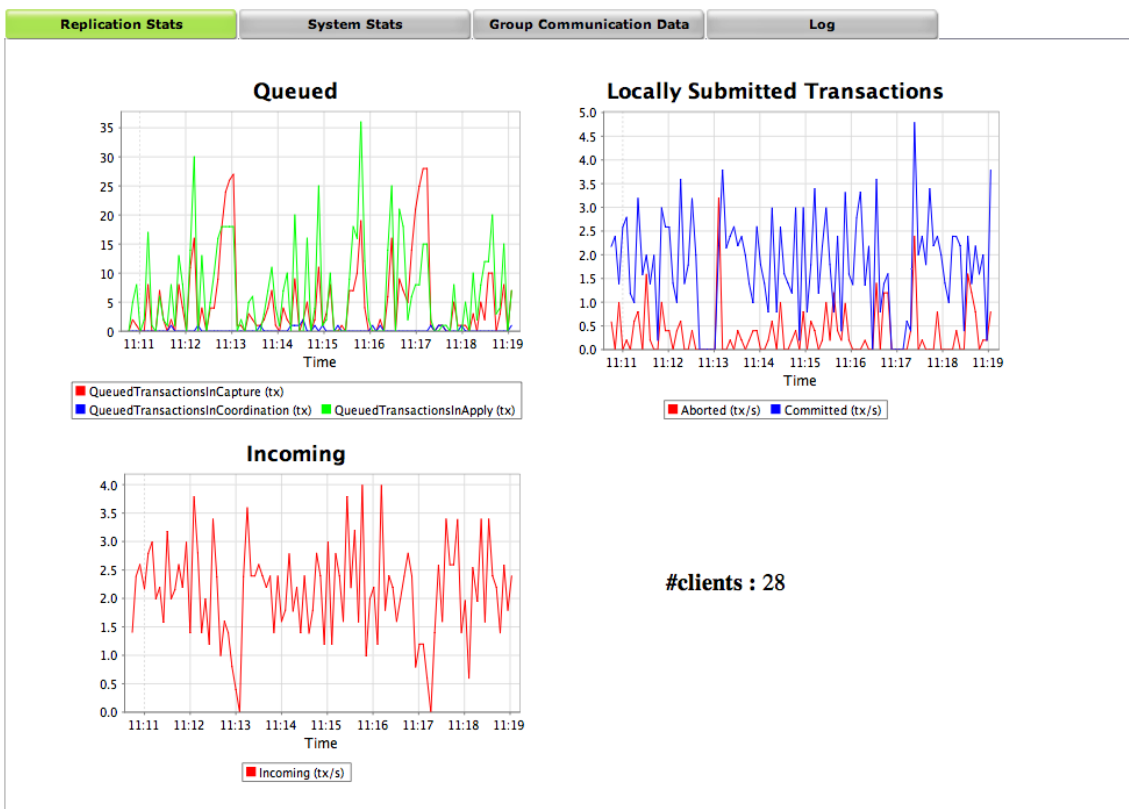


Figure 2.4: Replica view: Replication.

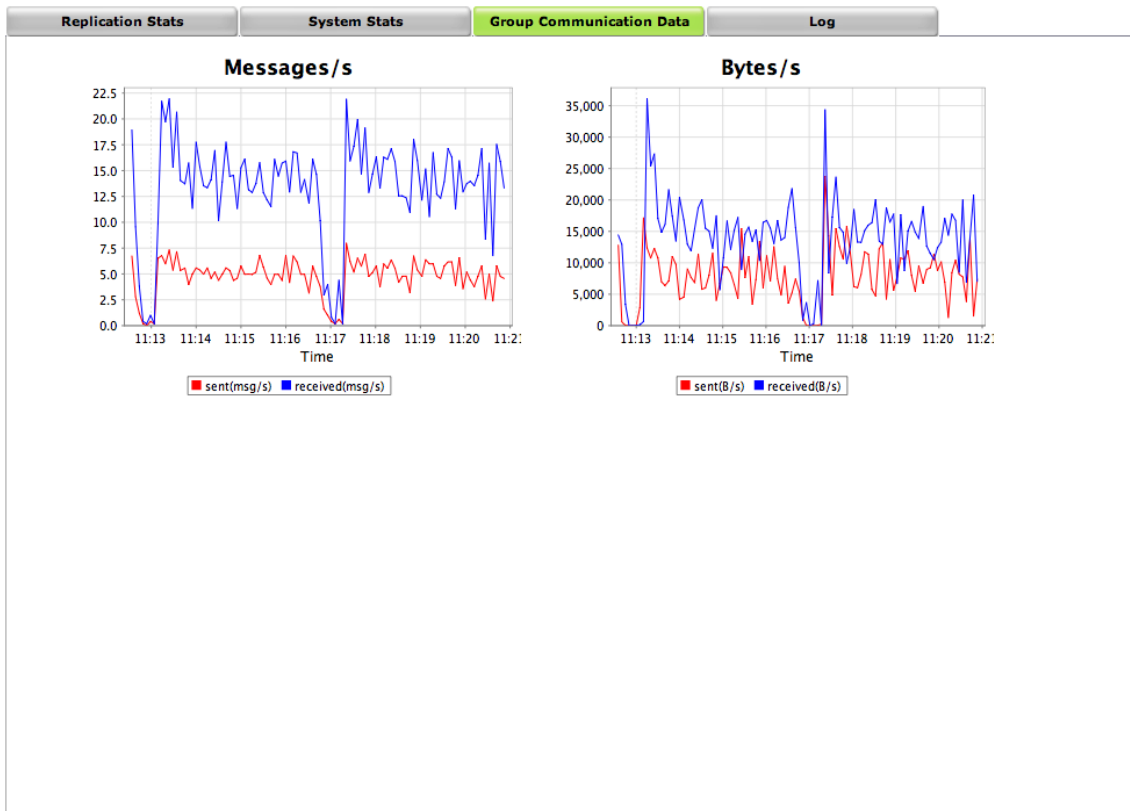


Figure 2.5: Replica view: Group communication.

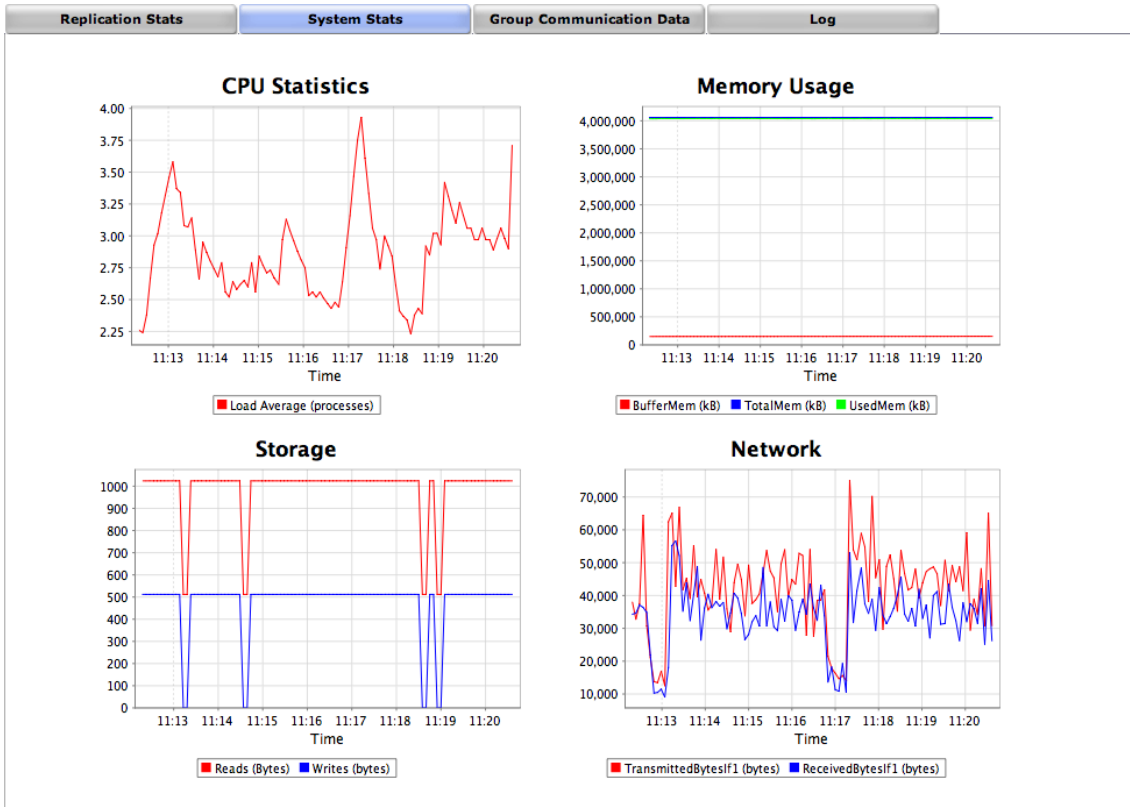


Figure 2.6: Replica view: System information.

Chapter 3

Plan

This chapter provides a detailed guide to the deployment and testing of the GORDA Integrated Prototype in three distinct scenarios.

3.1 Scenario 0: Basic System

3.1.1 Goals

This scenario aims at demonstrating a GORDA system by setting it up from publicly distributed software packages, as would be done by an end-user or system integrator while evaluating GORDA technology. In detail, this scenario shows a simple primary-backup replication setup using Apache Derby, ESCADA, and Appia.

3.1.2 Pre-requisites

- At least one workstation.
- Java Runtime Environment (JRE) version 1.5 installed.

3.1.3 Deployment

- Obtain the “quickstart” package from <http://escada.sf.net>.
- Unpack the file.
- Run the included `create-sample-databases` to create sample databases.
- Observe Appia configuration.

- Observe ESCADA configuration.

3.1.4 Usage

- Connect the standard Apache Derby client console, `i j`, to each of the replicas.
- Issue an update to the primary replica.
- Observe the update propagating to a backup replica.
- Issue an update to a backup replica and observe it failing.
- Launch `jconsole` and connect to one of the replicas.
- Observe system configuration and status parameters.

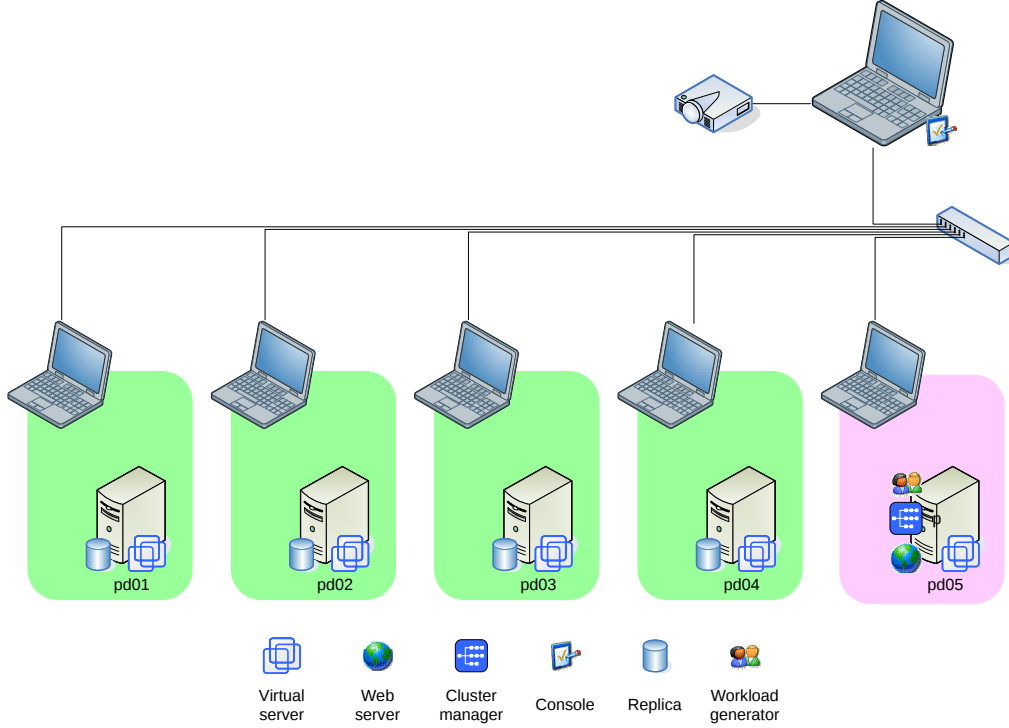


Figure 3.1: Environment of the complete system scenario.

3.2 Scenario 1a: Complete System and Fault-Tolerance

3.2.1 Goals

This scenario demonstrates the complete GORDA Integrated Prototype in a cluster, including a multi-master/update everywhere replication protocol, automated deployment, recovery, and a complex workload. This scenario is prepared to be run in a set of laptop computers in order to be performed wherever needed. In detail, it shows PostgreSQL, ESCADA, Jade, and Appia. The outline of the hardware and software architecture is presented in Figure 3.1.

The workload used is based on the industry standard benchmark TPC-C. The difference is that in this Scenario we don't respect the scaling requirements because the size of the database would mean a very large amount of time for initializing the system and during recovery, incompatible with a public demonstration. On the other hand, the usage of a smaller database has also the effect of increasing the likelihood of update conflicts. The Scenario in Section 3.4 is performed with a properly scaled database.

3.2.2 Pre-requisites

- Cluster nodes running the remote deployment agent.
- A web server with files for deployment.
- The cluster manager running.
- The console server running.
- A web browser to interact with the console.
- A video projector for public display of the console.

3.2.3 Deployment

- Start all replicas. This deploys all software and starts it.
- Observe group status.
- Initialize the replica group. This step is required when the cluster boots from scratch, not when recovering failed replicas.
- Observe group status.
- Start the database populate client process. This connects to one replica and inserts data, that is propagated to the entire group by replication.
- Check consistency.

3.2.4 Usage

- Start client processes on a single replica.
- Observe system status, system performance and resource usage measurements.
- Start client processes in all other replicas.
- Observe system status, system performance and resource usage measurements.

3.2.5 Failure and recovery

- Physically remove one replica.
- Observe group status.
- Observe system performance and resource usage during failover.
- Reboot replica and restart it.
- Observe group status.
- Observe system performance and resource usage during recovery.
- Observe group status after recovery.
- Check consistency.

3.3 Scenario 1b: Complete System and Adaptation

3.3.1 Goals

This scenario complements Scenario 1a of Section 3.2 by adding self-tuning. By using Jade it adapts the number of deployed replicas based on the workload. The hardware, software, and workload pre-requisites and configurations is the same as the one described in the previous section.

3.3.2 Pre-requisites

- Cluster nodes pre-deployed with four replicas, populated but without active clients, as available after running Scenario 1a in Section 3.2.

3.3.3 Usage

- Start 10 clients in the workload generator.
- Observe group status.
- Enable autonomic management in the cluster manager.
- Wait until the group size adapts to 10 clients and observe group status.
- Add 5 additional clients.
- Observe group status.
- Disable autonomic management in the cluster manager.
- Check consistency.

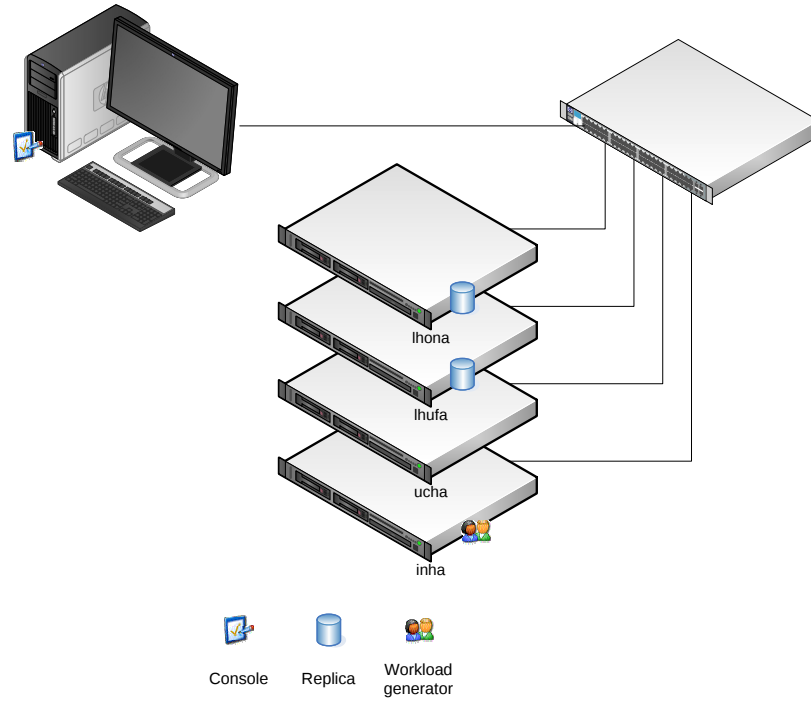


Figure 3.2: Environment of realistic workload scenario.

3.4 Scenario 2: Realistic Workload

3.4.1 Goals

This scenario demonstrates the performance of a GORDA configuration identical to the one described in Section 3.2 running the workload of the industry standard TPC-C benchmark with proper scaling. Since this benchmark requires a large amount of resources to run properly, it is run in dedicated servers and recorded for public playback. Extensive logs are also collected to allow a more precise evaluation that can be obtained by visual inspection of the console. The outline of the hardware and software architecture is presented in Figure 3.2.

3.4.2 Pre-requisites

- Cluster nodes pre-deployed with the software, all replicas running and the database already populated.
- The console server running.
- A web browser to interact with the console.

- Facility to capture screen to video active.

3.4.3 Usage

- Start multiple client processes.
- Observe system status, system performance and resource usage measurements.

3.4.4 Evaluation

- Logs are processed to extract detailed performance and resource usage information.

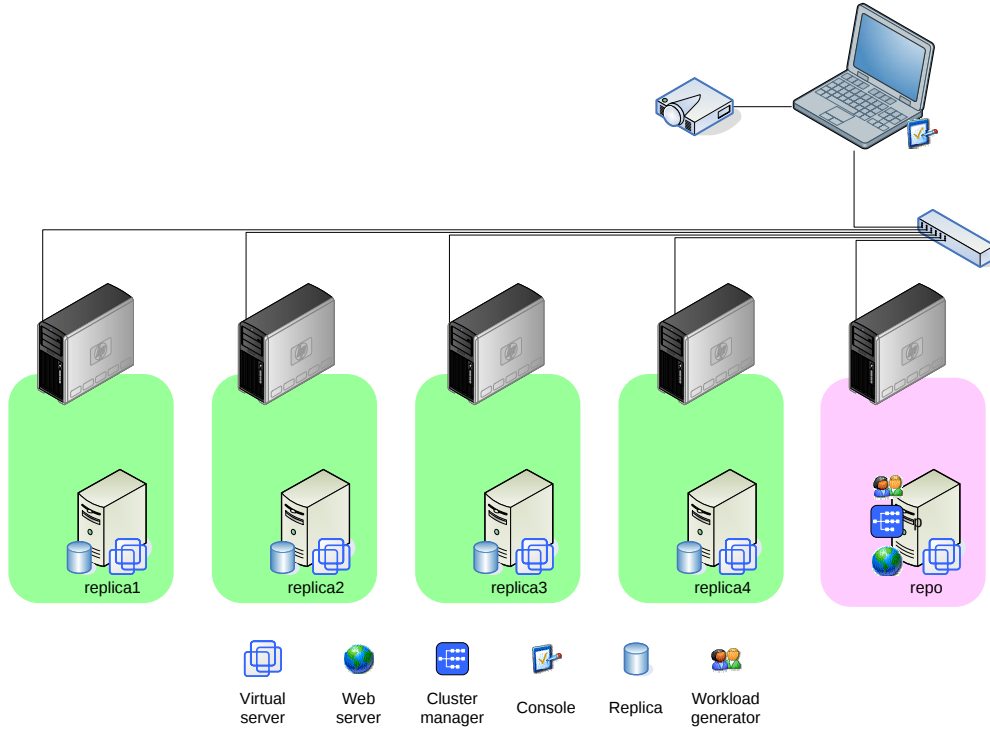


Figure 3.3: Environment of the legacy DBMS scenario.

3.5 Scenario 3: Legacy DBMS

3.5.1 Goals

This scenario demonstrates a cluster with an off-the-shelf MySQL setup using Sequoia 3.0, the GORDA Middleware proof-of-concept. This scenario includes a primary-backup replication protocol, automated deployment and a complex workload. This scenario is prepared to be run in a set of virtual machines in order to be performed wherever needed. In detail, it shows Sequoia 3.0, ES-CADA Replication Server, Appia, and Jade. The outline of the hardware and software architecture is presented in Figure 3.3. The workload used is the same detailed in Section 3.2.

3.5.2 Pre-Requisites

- Cluster nodes running the remote deployment agent.
- A web server with files for deployment.
- The console server running.
- A web browser to interact with the console.

- A video projector for public display of the console.

3.5.3 Deployment

- Start all replicas. This deploys all the software and starts it.
- Observe group status.
- Initialize the replica group. This step is required when the cluster boots from scratch.
- Observe group status.
- Start the database populate client process. This connects to the master replicas and inserts data, that is propagated to the entire group by replication.
- Check consistency.

3.5.4 Usage

- Start client processes connected to the master replica.
- Observe system status, system performance and resource usage measurements.
- Check consistency.

Bibliography

- [1] Transaction Processing Performance Council (TPC). TPC BenchmarkTM C Standard Specification Revision 5.0, February 2001.