Experimental Performability Evaluation of Middleware for Large-Scale Distributed Systems

L. Soares
losoares@di.uminho.pt

J. Pereira
jop@di.uminho.pt

Distributed Systems Group
Informatics Department
University of Minho
PORTUGAL

Context

- Database Replication providing high availabitiy and high performance.
- Replication protocols based on group communication services, for example: DBSM (Database State Machine).
  - Optimistic Execution.
  - Atomic Broadcast.
  - Deterministic Certification Process.
- We study both wide area and cluster replication.

Goals

- Incremental development and testing repeatability.
- Keep control of testing variables (specially hard in WAN networks).

Problem Statement

- How to perform realistic tests under self-contained and highly controllable environments?
- How to setup and support a cost-effective test environment with few resources?
- Existing testing and validation approaches do not meet all the requirements.

Existing Approaches

DART and PlanetLab

- A collection of machines distributed over the globe, in which applications run across all (or some) of the machines.
- Full implementations required.

FAUMachine

- High realistic virtualization software, which allows fault injection.
- Small scale tests and full implementations required.

Existing Approaches

Unit Testing

- Designed to evaluate the correctness of a particular unit of software.
- Limited when analyzing performance and dependability of large scale distributed middleware.

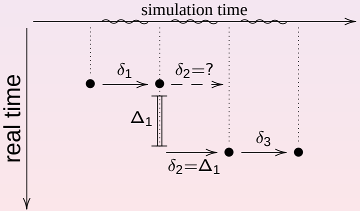
CESIUM

- Presents the notion of centralized simulation, in which distributed processes execute under the same address space. It is nearly want we wanted.

Discrete Event Simulation and Real Execution

- Provides deterministic execution and repeatability which is most valuable for debugging and tuning.
- Reuse validated simulation models mimicking the real components.
- Embed real implementations into the simulation environment.
- Freedom to choose testing and evaluation scenarios.
- Reduced probe effect in observations.
- Demands less resources than a real system.

- A simulation keeps virtual timelines that are explicitly advanced only by scheduling events
- When running real code, real time must modify the simulation clock



An example:

- A request message is transmitted over a simulated network with delay δ_1.
- Some real code is run at the server to handle the message and is measured as taking $\triangle_1$.
- A reply message is transmitted back with a delay δ_3.

The timing problem gets solved if the simulation kernel was capable of:

- Accounting time spent in real execution (precision issues).
- Scheduling events from the real execution into the simulation run-time, and making them show up at the correct instant in virtual time.
- Embedding real components into simulation framework without changes to the component source code.

Simulation Kernel With Real-Time Concerns - Contribution

- Augmented a popular open-standard simulation kernel, Scalable Simulation Framework (SSF), with real-time extensions:
  - Interface enabling transparent accounting of time spent on execution of real code.
  - Proxies enabling the interaction between unmodified implementations and simulation models.
- Extending SSF kernel enables the usage of libraries of simulation models already available for the original one.

Simulation Kernel With Real-Time Concerns - Contribution

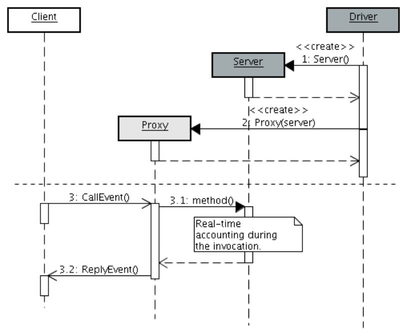


Figure: Simulation Calls Real

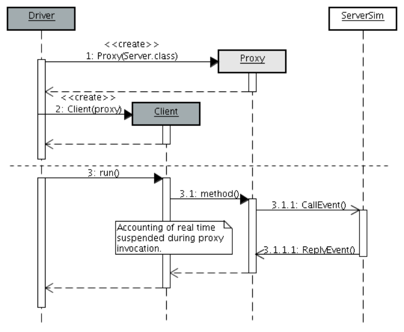


Figure: Real Calls Simulation

Database Replication Evaluation

Fault Tolerante Scalable Distributed Databases (Cluster Approach)

- Three sites connected through a LAN.
- Simulation Models: Database engine and network.
- Real Code: Group Communication and DBSM Protocol.
- Results published in the DSN-PDS’05.

Open Replication of Databases (Large Scale Approach)

- Nine sites connected through a LAN/WAN.
- Simulation Models: Database engine and network.
- Real Code: Group Communication, Middle-R, Postgres-R and DBSM protocol.
- Results published in the LADC’05.

Database Replication Evaluation

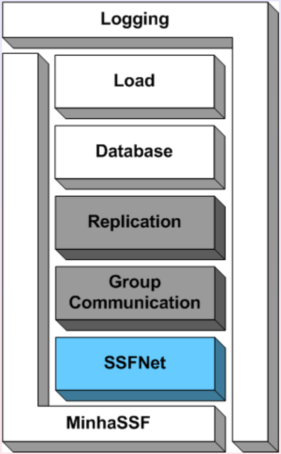
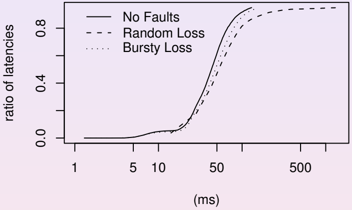


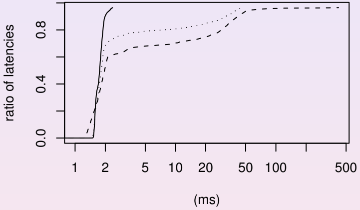
Figure: Database Site Model

- The kernel, we named it MinhaSSF, embraces all components which also make use of logging facilities.
- Load is generated according the TPC-C benchmark.
- The database box is a simulation model.
- Grey shaded boxes, are real components embedded in the simulation.
- Blue shaded box is a library of fully featured and fully tested network simulation models.

Database Replication Evaluation - Fault Injection



(a) Transaction latency.



(b) Certification latency.

Figure: Performance results with fault injection (750 Clients, LAN).

Database Replication Evaluation - Fault Injection (II)

No Faults	Random Loss (5%)	Bursty Loss (5%)
6.72	11.94	7.96

(a) Abort rates with 3 sites and 750 clients (%).

Run	**Usage**
No Faults	1.22
Random Loss	1.90
Bursty Loss	1.89

(b) Protocol CPU usage (%).

Simulation Performance

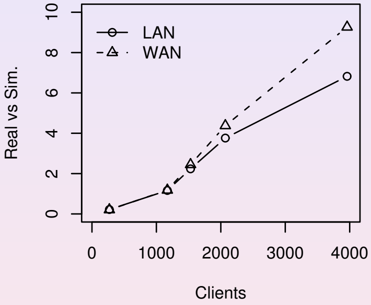


Figure: Ratio between real and simulation time.

- Simulation was performed in a Dual Opteron at 2.4GHz.
- Each database connection emulates a TPC-C client.
- Simulating 1 minute takes from 6 to 10 in real time when there are 4000 clients.
- The 10 minutes regard the centralized execution of 9 distributed sites.

Conclusions and Future Work

- Extended the SSF kernel to enable centralized execution.
- Use recognized simulation models, SSFNet, or even develop our own.
- The framework enables incremental development.
- It has been used to evaluate database replication using real implemented replication protocols.
- We measure performance, availability and reliability metrics with neglectible probe effect.
- Issues and future work:
  - Accounting of single-threaded run-times only.
  - Enhance the calibration of simulation models.

Contact Information

People

L. Soares - losoares@di.uminho.pt

J. Pereira - jop@di.uminho.pt

Projects

GORDA - http://gorda.di.uminho.pt [/community]

ESCADA - http://escada.lsd.di.uminho.pt

StrongRep - http://strongrep.lsd.di.uminho.pt

Research Team

GSD - http://gsd.di.uminho.pt