Implementing Replication Protocols in the MADIS Architecture

J.E. Armendáriz¹, J.R. Juárez¹, F.D. Muñoz-Escobar², J.R. Garitagoitia¹, I. Unzueta¹, L. Irún-Briz²

¹ Dpto. Matemática e Informática
Universidad Pública de Navarra
Campus de Arrosadia s/n
31006 Pamplona (Spain)
{enrique.armendariz, jr.juarez, joserra, itziar.unzueta}@unavarra.es

² Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
Camino de Vera s/n
46022 Valencia (Spain)
{fmunyoz, lirun}@iti.upv.es
Contents

- Problem Description.
- System Model.
- The BULLY Replication Protocol Description.
- The TORPE Replication Protocol Description.
- The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
Problem Description

- Accomplishing database replication by way of database internals modification.
  - O2PL was the first protocol avoiding read-set propagation.
  - Group Communication-based database replication protocols.
    - Total order delivery guarantees.
    - They do not need the 2PC rule to terminate a transaction.
    - E.g.: 1SER, Optimistic Atomic Broadcast in Transaction Processing Systems, etc.

- A middleware architecture providing database replication.
  - We propose two eager update everywhere replication protocols:
    - GCS-based Protocol → TORPE: An adaptation of the 1SER proposed by Kemme et al.
    - No lock management needed at the middleware layer.
Contents

- Problem Description.
- System Model.
  - The BULLY Replication Protocol Description.
  - The TORPE Replication Protocol Description.
  - The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
System Model

- **Middleware Replication**
  - Replication Protocol
  - User Application Transaction t

- **DBMS**
  - {t' ∈ T: WS ∩ (RS(t') ∪ WS(t'))}
  - notify(t)

- **Group Communication System**
  - sendTOMulticast
  - sendRUnicast
  - receive

- **Transactions**
  - create(t)
  - begin_operation(t,op)
  - begin_commit(t)
  - abort(t)
  - end_operation(t,op)
  - end_commit(t)

- **Group Operations**
  - begin(t)
  - commit(t)
  - abort(t)
  - submit(t,op)

- **WS(t)**
  - getConflicts(WS)
System Model

- Our model is partially synchronous.
  - Different sites may run at different rates.
  - Message delay is unknown but under certain bounded limit.

- We consider a primary partition system.
  - Partial amnesia crash.
Contents

- Problem Description.
- System Model.
- The BULLY Replication Protocol Description.
- The TORPE Replication Protocol Description.
- The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
**BULLY Protocol Behavior**

- **Node i** Local Txn
- **Node j** Remote Txn
- **Node k** Remote Txn

**Read & Write Phase**
- `create(t)` : `{t.site} = i`

**Begin Operation**
- `begin_operation(t,op)`

**Send Updates**
- `{r(x), w(y), r(z), w(x), w(p)}`

**Wait Response**
- `{j, k}`
- `WS= {p, x, y}`

**Remote Transaction**
- **Apply Updates** `{p, x, y}`

**Termination Phase**
- **Commit**
- `<commit,t>`

**Node j** Remote Txn
- `<ready,t,j>`

**Node i** Local Txn
- `<ready,t,k>`
- `<commit,t>`

**Node k** Remote Txn
- `<ready,t>`
The “Priority” Concept

- A priority deadlock prevention function for conflicting transactions.
  - Dynamic. It depends on the status of the transaction and its associated priority (t.priority).
  - It does not allow remote transactions to wait.
    - Two different conflicting remote transactions will be rolled back at different nodes.
The "Priority" Concept

◊ function higher_priority(t, t') \equiv 
   node(t) = j \neq i \land (a \lor b \lor c)
\hspace{1cm} (a) \hspace{1cm} node(t') = i \land status_i(t') \in \{\text{active, blocked}\}
\hspace{1cm} (b) \hspace{1cm} node(t') = i \land status_i(t') = \text{pre_commit} \land 
   t.priority > t'.priority
\hspace{1cm} (c) \hspace{1cm} node(t') = k \land k \neq j \land k \neq i \land 
   status_i(t') = \text{blocked} \land 
   t.priority > t'.priority
Contents

- Problem Description.
- System Model.
- The BULLY Replication Protocol Description.
- The TORPE Replication Protocol Description.
- The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
TORPE Protocol Behavior

Node i
Local Txn

Node j
Remote Txn

Remote Transaction:
Termination Phase:
commit

<remote,t,WS>

<commit,t>

Node k
Remote Txn

Remote Transaction:
Apply Updates
{p, x, y}

Remote Transaction:
Apply Updates
{p, x, y}

Termination Phase:
commit

{r(x),w(y),r(x),w(p)}

Termination Phase:
commit

Termination Phase:
commit

<commit,t>
Contents

- Problem Description.
- System Model.
- The BULLY Replication Protocol Description.
- The TORPE Replication Protocol Description.
- The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
MADIS Architecture

User Application

Consistency Manager (JDBC Driver)

Conflict Detector

Replication and Recovery Protocol

DBMS

Communication Service
Membership Service

Group Comm. System

JDBC Interface

Original Schema

Extended Schema
Contents

- Problem Description.
- System Model.
- The BULLY Replication Protocol Description.
- The TORPE Replication Protocol Description.
- The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
Response time with 4 operations per transaction in a MADIS architecture varying the number of transactions per second and the number of clients.
Performance Analysis

Response time with 4 operations per transaction in a MADIS architecture varying the number of transactions per second and the number of clients.
Scalability Analysis

Response time with 1 operation per transaction in a MADIS architecture varying the number of nodes and the number of clients.
Scalability Analysis

Response time with 1 operation per transaction in a MADIS architecture varying the number of nodes and the number of clients.
Contents

- Problem Description.
- System Model.
- The BULLY Replication Protocol Description.
- The TORPE Replication Protocol Description.
- The MADIS Middleware Architecture.
- Performance Evaluation.
- Conclusions.
Conclusions

We have introduced two eager update everywhere replication protocols for the MADIS middleware architecture.

- They ensure 1-copy-serializability.
  - BULLY: It is based on O2PL.
  - TORPE: It is based on 1SER.
- No DBMS specific operations must be reimplemented at the middleware layer.

Preliminary results.
- PostgreSQL provides SI.
Implementing Replication Protocols in the MADIS Architecture

J.E. Armendáriz¹, J.R. Juárez¹, F.D. Muñoz-Escofí², J.R. Garitagoitia¹, I. Unzueta¹, L. Irún-Briz²

¹ Dpto. Matemática e Informática
Universidad Pública de Navarra
Campus de Arrosadía s/n
31006 Pamplona (Spain)
{enrique.armendariz, jr.juarez, joserra, itziar.unzueta}@unavarra.es

² Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
Camino de Vera s/n
46022 Valencia (Spain)
{fmunyoz, lirun}@iti.upv.es