Implementing O2PL Protocols in a Middleware Architecture for Database Replication

J.R. Juárez¹, J.E. Armendáriz¹, J.R. Garitagoitia¹, J.R. González de Mendívil¹, F.D. Muñoz-Escoí²

Presents: José Enrique Armendáriz Iñigo
enrique.armendariz@unavarra.es

¹ Grupo de Sistemas Distribuidos
Dpto. Matemática e Informática
Universidad Pública de Navarra

² Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
Outline

- Introduction
- Basic Replication Protocol
- Enhanced Replication Protocol
- MADIS Architecture
- Experimental Results
Introduction

- Database replication is a way to
  - Increase system performance
  - Increase fault-tolerance of a given system
- The price to pay
  - Effort needed to guarantee data consistency
- O2PL was one of the first ROWAA protocols
  - It is a 2PC protocol for a DBMS-core modified architecture
  - Guarantees a lower distributed deadlock rate
    - Copy lock
    - Snoop process
Introduction

- A middleware-based data replication architecture
  - No lock management needed at the middleware layer
  - We rely for concurrency control on the DBMS
  - DBMS internals remain unaccessible
  - The consistency is maintained at the middleware layer
  - Client applications “see” a standard interface (e.g. JDBC)
Introduction

- We propose two protocols for a middleware-based data replication architecture:
  - Basic Replication Protocol (BRP)
    - Adaptation of the O2PL proposed by M.J. Carey and M. Livny
    - Distributed deadlock is avoided by a dynamic deadlock prevention schema
      - Transactions have a unique global priority
      - It also depends on the state of the transaction in the system.
  - Enhanced Replication Protocol (ERP)
    - BRP Response time:
      - $\theta_r(t) \approx \theta_{DB_i}(t) + \theta_{MC}(t) + \max_j(\theta_{DB_j}(t)) + \max_j(\theta_{UC_j}(t))$
    - If unilateral aborts are not considered
      - Deadlock prevention function guarantees a total order execution of transactions
        - $\theta_r(t) \approx \theta_{DB_i}(t) + \theta_{MC}(t)$
Basic Replication Protocol

Remote Txn

create(t)  
begin_operation(t,op)  
end_operation(t,op)  
{r(x),w(y),r(z),w(x)}

Active

begin_commit(t)  
WS = {x,y}  
Send Updates  
{r(x),w(y),r(z),w(x)}

Wait response  
{r(x),w(y),r(z),w(x)}

receive_commit(t,m)  
Commit  
committed

receive_remote(t,m)  
Apply Updates  
{r(x),w(y),r(z),w(x)}

receive_commit(t,m)  
Commit  
committed

receive_commit(t,m)  
Commit  
committed

receive_remote(t,m)  
Apply Updates  
{r(x),w(y),r(z),w(x)}

receive_remote(t,m)  
Apply Updates  
{r(x),w(y),r(z),w(x)}

Receive remote

j

i

k
Enhanced Replication Protocol

Remote Txn

1. `create(t)`
2. `begin_operation(t, op)`
3. `end_operation(t, op)`
   - `WS = {x, y}`
   - `Send Updates`

Active

Remote Txn

1. `receive_remote(t, m)`
2. `apply_updates(WS)`

Pre_commit

Remote Txn

1. `receive_remote(t, m)`
2. `apply_updates(WS)`

Commit

Remote Txn

1. `receive_commit(t, m)`
2. `commit()`

Committed
MADIS Architecture

Java Client Program
Servlet Container
EJB Container

JDBC Interface

Consistency Manager (JDBC Driver)
Conflict Detector
Consistency and Recovery Protocol

MADIS

Communication Service
Membership Service

Group Comm. System

DBMS
Extended Schema
Original Schema
<table>
<thead>
<tr>
<th>Experiments</th>
<th>WL1</th>
<th>WL2</th>
<th>WL3</th>
<th>Conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Size</td>
<td>30 tables of 1000 tuples each</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuple Size</td>
<td>Appr.100 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Servers</td>
<td>5</td>
<td>2-8</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Number of Update Operations</td>
<td>5</td>
<td>10</td>
<td>5-25</td>
<td>5</td>
</tr>
<tr>
<td>Number of Clients</td>
<td>1-20</td>
<td>2-8</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Submission Rate in TPS</td>
<td>10-35</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Hot Spot size</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>Varying</td>
</tr>
</tbody>
</table>
Performance Analysis

The graphs show the response time (in ms) for different TPS (Transactions Per Second) values and varying numbers of clients for ERP and BRP systems.

Key:
- 1 client
- 2 clients
- 5 clients
- 10 clients
- 15 clients
- 20 clients
Performance Analysis

![Graph showing response time for ERP, BRP, and UPDATE_TIME against the number of servers. The response time increases as the number of servers increases.]
Performance Analysis

![Graph showing response time versus updates per transaction. The graph compares three systems: ERP, BRP, and UPDATE_TIME. The x-axis represents updates per transaction ranging from 5 to 25, and the y-axis represents response time in milliseconds ranging from 0 to 400. ERP shows a consistently higher response time compared to BRP and UPDATE_TIME.](image-url)
Performance Analysis

![Graph showing Performance Analysis](image-url)
Implementing O2PL Protocols in a Middleware Architecture for Database Replication

J.R. Juárez¹, J.E. Armendáriz¹, J.R. Garitagoitia¹, J.R. González de Mendívil¹, F.D. Muñoz-Escoí²

Presents: José Enrique Armendáriz Iñigo
enrique.armendariz@unavarra.es

¹ Grupo de Sistemas Distribuidos
Dpto. Matemática e Informática
Universidad Pública de Navarra

² Instituto Tecnológico de Informática
Universidad Politécnica de Valencia