Managing Update Conflicts in Bayou

Lucy Youxuan Jiang, Hiwot Tadesse Kassa
Outline

● Background + Motivation

● Bayou Model
  ○ Dependency checking for conflict detection
  ○ Merge procedures for conflict resolution
  ○ Replica consistency

● Storage System Implementation

● Experimental Results + Conclusion
Background (1995)

We have strong connection!

I can connect!

I can barely connect...

I lost connection!!!
Problem: Weak Connectivity

Device 1

Device 2  Can’t Reach  Device 3

What’s happening to other replicas?!
Bayou

- Previous Attempts
  - Focused on transparent replicated data support for storage systems

- Bayou Insights
  - Applications should be aware of the risks of weak connectivity and must help detect and resolve those conflicts

- Goal
  - Provide an infrastructure for collaborative applications to manage conflicts between concurrent activities relying on weak connectivity
Two Applications

Meeting Room Scheduler

- At most one person (or group) can reserve the room for any given period of time.

Bibliographic database

- Users cooperatively manage databases of bibliographic entries (e.g., append, update)

Tentative:

Waiting to be confirmed or reject due to possible conflicts
Bayou Model

- Bayou system is replicated in full at a number of servers
- Applications running as clients interact with the servers through the Bayou API
- Presents a weakly consistent replication model with a read-any/write-any style of access
Bayou Model

Bayou Model

- Bayou servers propagate Writes among themselves during pair-wise contacts, called **anti-entropy sessions**

- The session is used agree on the set of Bayou Writes they have seen and the order in which to perform them

- as long as the set of servers is not permanently partitioned each Write will eventually reach all servers
Bayou Model

Conflict Detection and Resolution

Meeting Room Scheduler

- At most one person (or group) can reserve the room for any given period of time.

- **Application-specific** conflict detection and resolution

Bibliographic database

- Each publication have distinctive key
Conflict Detection and Resolution

- Bayou writes include
  - Update function - the actual value to be written
  - Dependency check - used for conflict detection
  - Merge function - decided how the conflict should be resolved
Dependency checks

- Consists an application-supplied query and its expected result

```python
Bayou_Write(
    update = {insert_Meetings.12/18/95.1:30pm.60min."Budget Meeting"},
    dependency_check = {
        query = "SELECT key FROM Meetings WHERE day = 12/18/95
        AND start < 2:30pm AND end > 1:30pm",
        expected_result = EMPTY},
)```
Merge procedures

- **Resolves** conflicts detected by its dependency check

```c
mergeproc = {
    alternates = {{12/18/95, 3:00pm}, {12/19/95, 9:30am}};
    newupdate = {};
    FOREACH a IN alternates {
        # check if there would be a conflict
        IF (NOT EMPTY ( 
            SELECT key FROM Meetings WHERE day = a.date
            AND start < a.time + 60min AND end > a.time))
            CONTINUE;
        # no conflict, can schedule meeting at that time
        newupdate = {insert, Meetings, a.date, a.time, 60min, "Budget Meeting"};
        BREAK;
    }
}
```
Replica consistency

- All servers move towards eventual consistency
- When a Write is accepted by a Bayou server from a client, it is initially deemed **tentative**
- Eventually, each Write is **committed**
- Servers must be able to undo the effects of some previous tentative execution of a Write operation and reapply
Replica consistency


Meeting MAR 19,3:00PM
Replica consistency

How is consistency achieved?

1. Include timestamp of the update requests
2. When a replica receive a new request re-execute based on timestamp

A write is stable when it is executed for the last time
Replica consistency

What is the problem?

Storage System Implementation

Read Access

1. Authenticate your identity…
2. Is your certificate legitimate?
   OK, access granted!

Client

Show me your data!

Request
public/private key
access control certificates

Data

Request

Data

Server Replica
Write Access

Update your data!

1. Authenticate identity
2. Verify certificate
OK, update

Data Authenticating Info

Client

Accepting Replica

Primary

Data Authenticating Info
Evaluation and Performance

- Bibliographic database with **five different configurations** of the database characterized by the number of Writes that are tentative.
- measured storage requirements and the execution times for
  - undoing and redoing the effect of all tentative Writes
  - executing a client Read operation against the database
  - adding a new Write to the database
## Evaluation and Performance

<table>
<thead>
<tr>
<th>Number of Tentative Writes</th>
<th>0 (none)</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1550 (all)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Log</td>
<td>9</td>
<td>129</td>
<td>259</td>
<td>1302</td>
<td>4028</td>
</tr>
<tr>
<td>Tuple Store Ckpt</td>
<td>396</td>
<td>384</td>
<td>371</td>
<td>269</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>405</strong></td>
<td><strong>513</strong></td>
<td><strong>630</strong></td>
<td><strong>1571</strong></td>
<td><strong>4029</strong></td>
</tr>
<tr>
<td>Factor to 368K bibtex source</td>
<td>1.1</td>
<td>1.39</td>
<td>1.71</td>
<td>4.27</td>
<td>10.95</td>
</tr>
</tbody>
</table>
## Evaluation and Performance

<table>
<thead>
<tr>
<th>Tentative Writes</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1550</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undo all (avg. per Write)</td>
<td>0</td>
<td>31 (6)</td>
<td>70 (20)</td>
<td>330 (155)</td>
<td>866 (195)</td>
</tr>
<tr>
<td>Redo all (avg. per Write)</td>
<td>0</td>
<td>237 (85)</td>
<td>611 (302)</td>
<td>2796 (830)</td>
<td>7838 (1094)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tentative Writes</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1550</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undo all (avg. per Write)</td>
<td>0</td>
<td>47 (3)</td>
<td>104 (7)</td>
<td>482 (15)</td>
<td>1288 (62)</td>
</tr>
<tr>
<td>Redo all (avg. per Write)</td>
<td>0</td>
<td>302 (91)</td>
<td>705 (134)</td>
<td>3504 (264)</td>
<td>9920 (294)</td>
</tr>
</tbody>
</table>
## Evaluation and Performance

### Table 3: Performance of the Bayou Client Operations
(times in milliseconds with standard deviations in parentheses)

<table>
<thead>
<tr>
<th>Server Client</th>
<th>Sun SPARC/20 same as server</th>
<th>Gateway Liberty same as server</th>
<th>Sun SPARC/20 Gateway Liberty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read: 1 tuple</td>
<td>27 (19)</td>
<td>38 (5)</td>
<td>23 (4)</td>
</tr>
<tr>
<td>100 tuples</td>
<td>206 (20)</td>
<td>358 (28)</td>
<td>244 (10)</td>
</tr>
<tr>
<td>Write: no conflict</td>
<td>159 (32)</td>
<td>212 (29)</td>
<td>177 (22)</td>
</tr>
<tr>
<td>with conflict</td>
<td>207 (37)</td>
<td>372 (17)</td>
<td>223 (40)</td>
</tr>
</tbody>
</table>
Conclusion

- Bayou is a storage infrastructure for mobile applications
- The system is built around pair-wise client-server and server-server communications
- To provide high availability, Bayou employs weakly consistent replication
- Based on Application-specific conflict detection and resolution
- Guarantees eventual consistency