Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System

Authors: Terry et al.
Presented by Yitong Wang
Introduction

Bayou assumes
• A weak connectivity network model

Bayou supports
• Weakly consistent, replicated data
• Eventual data consistency
• Read-any/Write-any access for clients
• Application-specific conflict resolution
System Model

• Data Collection is replicated in full in servers.

• Clients access the service through Bayou API to Read/Write.

• Pair-wise anti-entropy session to synchronize operations.
Example Application

Meeting Room Scheduler

Server 1

User 1

Server 2

User 2
Example Application

Meeting Room Scheduler

- Users’ view may be outdated
- Reservation should be at first tentative, which may be accepted or rejected later.
- Records will eventually be synchronized
Conflicts cannot be concluded by simply observing read/write operations from application
Conflict Detection and Resolution

Conflicts cannot be concluded by simply observing read/write operations from application.

Solution: Application-specific dependency check and merge procedures
Conflict Detection and Resolution

Bayou_Write (update, dependency_check, mergeproc) {
    IF (DB_Eval (dependency_check.query) <> dependency_check.expected_result)
        resolved_update = Interpret (mergeproc);
    ELSE
        resolved_update = update;
    DB_Apply (resolved_update);
}

Dependency check:
• Compares results of queries with expected results
• Works as a precondition for update
• Detects not only write-write but also read-write conflicts
Concrete Example

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 = EMPT},
    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 EMPT (  
                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;  
        }
        IF (newupdate = {})  
            # no alternate is acceptable  
            newupdate = {insert, ErrorLog, 12/18/95, 1:30pm, 60min, “Budget Meeting”};  
            RETURN newupdate;  
    }
Write Stability and Commitment

Definition:
A write is *stable or committed* if it’s executed for the last time.

Bayou allows accessing both stable and complete data
(use <timestamp, server ID> to identify)
Write Stability and Commitment

Bayou uses primary commit scheme:

- A primary server determines commit and propagates relevant knowledge.
- Bayou inherently accommodate temporary unavailability of primary
- Writes may not be committed in the order of when they are received
(Eventual) Replica Consistency

To support this, Bayou ensures

1. Writes are performed in a globally well-defined order

2. Conflict detection and merge procedures are deterministic
(Eventual) Replica Consistency

To support this, Bayou ensures

1. Writes are performed in a well-defined order

- Tentative writes ordered by timestamp
- Committed writes ordered by time and before tentative ones
- Need ability to undo write
To support this, Bayou ensures

2. Conflict detection and merge procedures are deterministic

- Procedures cannot access time-dependent or machine-specific info
- Computation resources such as CPU and memory are bounded identically
the timestamp of the latest Write from the given server that has been discarded
Storage System Implementation

- Timestamp Vectors
- Tuple Store
- Undo Log
- Write Log
- Committed
- Tentative
- Tuple Store (checkpoint)

In Memory

On Stable Storage
Storage System Implementation

2 bits to indicate committed or full
Evaluation

Table 1: Size of Bayou Storage System for the Bibliographic Database with 1550 Entries
(sizes in Kilobytes)

<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>Total</td>
<td>405</td>
<td>513</td>
<td>630</td>
<td>1571</td>
<td>4029</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

**Table 2: Performance of the Bayou Storage System for Operations on Tentative Writes in the Write Log**  
(times in milliseconds with standard deviations in parentheses)

<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><strong>Server running on a Sun SPARC/20 with Sunos</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Undo</strong> all</td>
<td>0</td>
<td>31</td>
<td>70</td>
<td>330</td>
<td>866</td>
</tr>
<tr>
<td>(avg. per Write)</td>
<td>.62</td>
<td>.7</td>
<td>.66</td>
<td>.56</td>
<td></td>
</tr>
<tr>
<td><strong>Redo</strong> all</td>
<td>0</td>
<td>237</td>
<td>611</td>
<td>2796</td>
<td>7838</td>
</tr>
<tr>
<td>(avg. per Write)</td>
<td>4.74</td>
<td>6.11</td>
<td>5.59</td>
<td>5.05</td>
<td></td>
</tr>
<tr>
<td><strong>Server running on a Gateway Liberty Laptop with Linux</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Undo</strong> all</td>
<td>0</td>
<td>.47</td>
<td>104</td>
<td>482</td>
<td>1288</td>
</tr>
<tr>
<td>(avg. per Write)</td>
<td>.94</td>
<td>1.04</td>
<td>.96</td>
<td>.83</td>
<td></td>
</tr>
<tr>
<td><strong>Redo</strong> all</td>
<td>0</td>
<td>302</td>
<td>705</td>
<td>3504</td>
<td>9920</td>
</tr>
<tr>
<td>(avg. per Write)</td>
<td>6.04</td>
<td>7.05</td>
<td>7.01</td>
<td>6.4</td>
<td></td>
</tr>
</tbody>
</table>
## Evaluation

### 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 100 tuples</td>
<td>27 (19)</td>
<td>38 (5)</td>
<td>23 (4)</td>
</tr>
<tr>
<td></td>
<td>206 (20)</td>
<td>358 (28)</td>
<td>244 (10)</td>
</tr>
<tr>
<td>Write: no conflict with conflict</td>
<td>159 (32)</td>
<td>212 (29)</td>
<td>177 (22)</td>
</tr>
<tr>
<td></td>
<td>207 (37)</td>
<td>372 (17)</td>
<td>223 (40)</td>
</tr>
</tbody>
</table>
Conclusion

In a weakly connected network, Bayou

• Can achieve eventual consistency
• Uses tentative and stable writes
• Supports application-specific confliction detection