“Managing Update Conflicts in Bayou, a Weekly Connected Replicated Storage System”
Presented by – SANKALP KOHLI
OVERVIEW

- INTRODUCTION
  - GOALS
  - LIMITATIONS
  - SYSTEM PROPERTIES
  - WHY AND HOW?

- COLLABORATIVE APPLICATIONS

- DESIGN
  - ALGORITHMS USED (Epidemic Algorithms)
  - BAYOU BASIC SYSTEM MODEL
  - IMPLEMENTATION
    - CONFLICT DETECTION AND RESOLUTION
    - MECHANISM
    - DESIGN OVERVIEW

- EVALUATION
  - SETUP
  - EVALUATION RESULTS

- ADVANTAGES AND DISADVANTAGES

- DISCUSSION (Questioners)
Bayou Storage System provides an infrastructure for collaborative applications that manages the conflicts introduced by concurrent activity.
GOALS

- Supporting Disconnected work group:
  - Architecture does not include the notion of “disconnected” mode of operation.

- System Should be Highly Available.
LIMITATIONS

- Bayou was designed to support few real-time collaborative applications.

- Bayou targets machines with
  - Expensive Connection Time
  - Mobile Handsets,
  - PDA’s

- Frequent or occasional disconnections.
  - Cellular telephony
PROPERTIES

- Replicated, Weakly Consistent Storage system.
- Designed for Mobile Computing Environment.
- High availability
- Novel Methods for Conflict Detection
- Defines protocol that stabilizes Update Conflicts
- In fracture for collaborative applications
- Weakly Connected
CHALLENGES

- Concurrent and conflicting updates
  - Consequence of high availability
- Dependency between operations
- Merge Procedures
  - need to be defined flexible merge procedures to support wide range of applications
- Write Propagation delays
  - Asynchronous – no bounds on time
  - cannot be enforced strict bounds as it depends on Network Connectivity.
Bayou replicated storage system was designed to support variety of Real Time Collaborative applications such as:

- Meeting Room Scheduler
- Mail and Bibliographic Data Bases
- Shared Calendars
- Program Development
- Document Editing
Collaborative Applications: Meeting Room Scheduler

- Allows Users to reserve a room.
- At most one person can reserve the room at any given point of time.
- Users interact with a Graphical Interface.
- Scheduler periodically re-reads room schedule and refreshes the users display.

**Problem:**
- Users reservation might be out of date wrt. confirmed reservation.
Collaborative Applications: Meeting Room Scheduler

- User can select several acceptable meetings.
- Only one requested time will eventually be reserved.
- Users reservation will not be confirmed immediately.
  - Initially it will be Tentative (Grayed out)
- Users though disconnected from the rest can immediately see the others tentative room reservation.
Collaborative Applications: Bibliographic Data Base

- Allows users to add entries to a Data Base.
- Can read and write any copy of the Data Base.

**Approach:**
- Each Entry has a unique key.
- Entry key is tentatively assigned when entry is added.
- Users must be aware of the concurrent updates and should wait till the key is confirmed.

**Problem:**
- Same Bibliographic entry may be added by the Different user with different Key.

**Solution:**
- System Detects Duplicates and merges into a single entry with a single key.
EPIDEMIC ALGORITHMS

- Randomized algorithms for distributing updates and driving the replicas towards consistency.

- Ensures that the effect of every update is eventually reflected in all replicas.

- Efficient and Robust and Scale gracefully.

- Factors considered in designing an efficient algorithm:
  - Time
  - Network traffic.

- Bayou Design Uses Epidemic Algorithms for Conflict detection and Resolution.
Different Strategies can be Used for spreading updates:

- **Direct Mail:**
  - Each New Update is immediately mailed from entry site to all other sites.
  - Timely and reasonably efficient.

- **Anti Entropy:**
  - Every Site regularly chooses another site at random.
  - Extremely reliable:
  - Propagation of Updates are Slow
Each site executes the Anti Entropy periodically.

Anti Entropy Algorithm is very **expensive**, 
- It involves comparison of 2 complete copies of the data base.

Alternate Approach: Maintain the **checksum** of its database.

Checksums at different sites likely to disagree if:
- time required for an update to sent to all sites > Expected time between new Updates.
BAYOU BASIC SYSTEM MODEL

CLIENT
- Application
- Bayou API’s
- Client Stub

SERVER
- Storage System
- Server State

CLIENT
- Application
- Bayou API’s
- Client Stub

Anti Entropy

Machine Boundaries

Client

Server
Each Database is replicated in Full at a number of servers.

Client Application interact with the servers through the Bayou’s API’s.

API’s and Underlying Client Server RPC protocol supports 2 basic Operations:

1. Read
2. Write (Insert/ Modify/ Delete)

Access to one server is sufficient for the client.

Read Any/ Write Any.

Bayou provides Session Guarantees.

Bayou write carries information that lets each server receiving the write decide if there is a conflict and if so how to fix it.
Global Unique Write ID

Storage system consists of Ordered log of writes + Data

Each Server performs
- Conflict Detection and Resolution.
- write locally.

Bayou server propagates writes among themselves during a pairwise contacts called **Anti-entropy session**.

**Theory of Epidemic Algorithm**: As long as the set of servers are not permanently portioned, each write will eventually reach all servers.
In the absence of new writes from the client, all Servers will hold the same data.

The rate at which servers reach convergence depends on several factors:

- Network Connectivity
- Frequency of Anti Entropy
- Policies by which servers select Anti-Entropy partners.
Basic Implementation

- Bayou System includes two mechanisms for automatic Conflict detection and Resolution.

- Mechanisms will support Arbitrary applications.

- Dependency check and Merge Procedures are Application specific.

- There are 2 States of an Update:
  - Committed
  - Tentative
- **What is a conflict?**
  - Its Application Specific.
  - Different Application has different notion for what it means for two applications to conflict.

- **How to Identify Conflicts?**
  - It cannot be identified by simply observing conventional reads and writes.
  - Application should specify its notion of conflict.

- **How to Resolve Conflicts?**
  - Application should specify the policy for resolving the conflicts.
  - Depending on the specified Conflict policy storage system will specify the mechanism for reliably detecting conflicts and automatically resolve
Conflict detection and Resolution E.g. Application Specific Conflict

- **Meeting Room scheduling conflict**
  - Two users have concurrently updated two replicas of the meeting room calendar and scheduling for the same room.

- **Bibliographic Data Base:**
  - Two users of Application describe different publications but have been assigned the same key by their submissions.
  - Two users describe the same publication but have been assigned the different keys.
Conflict Detection varies according to schematics of the application.

An application must specify its notion of conflict along with its policy for resolving the conflicts.

Depending on the specified conflict policy, storage systems specify the mechanisms for reliably detecting conflicts and automatically resolving conflicts.
The Bayou system indicates two mechanisms for automatic conflict detection and Resolution.

- **DEPENDENCY CHECK**
- **MERGE PROCEDURES**

Mechanism permits clients to indicate the following for each individual write operation:

- How the system detects the conflicts involving the write
- What steps should be taken to resolve the conflicts based on the schematics of the application.
Dependency check is the pre-condition for performing the update that is included in the write operation.

Application Specific conflict detection is accomplished.

Each write operation includes a dependency check on Application supplied query and expected result.

If the dependency check fails then the requested update is not performed.
Bayou Write (update, dependency Check, merge-procedure)
{
    if (DB_Eval(dependencyCheck.query) ==
        (dependencyCheck.result)  )
    {
        resolved_Update = Interpret and Execute merge-procedure;
    }
    else
    {
        resolved_Update = Update the Data Base;
    }
    DB_Apply (resolved_update);
}
Bayou_write(
    update = {Insert, TableName, ScheduleTime, "Comment"},
    dependency check = 
    { query = "Select key from Meetings Where DAY = 12/18/09 AND START <2:30 AND END > 1:30 PM;
        Expected Result = EMPTY },
    Merge procedure = 
    { Alternatives = {{12/18/2009,3:00pm},{12/19/2009, 9:00pm}
        newUpdate = { },
        FOREACH a in alternatives
        {# check for conflict
            if(NOTEMPTY (Select key from Meetings Where
                DAY = 12/18/09 AND start < a.time +60 AND END > a.time))
                CONTINUE;
            # NO CONFLICT, CAN SCHEDULE MEETING AT THAT TIME SO INSERT IT TO newUpdate
        new Update = {insert, Meetings, a.data, a.time, 60 mins, "Budget Meeting"}
        }
        IF(newUpdate == EMPTY)
            newUpdate = {INSERT ERROR LOG};
        RETURN newUpdate;
    }
}
Merge Procedures:

- If a Conflict is detected then Merge procedure is run by Bayou Server to resolve it.

- Written in High Level Interpreted Language by application programmer in the form of template.

- Can read Current State of Servers Replica.

- Produces a revised Update.
What if Automatic Conflict resolution is not possible?

- Server will log the conflict, which will be used later by the user to resolve the conflict.

- Using Interactive Merge tool the conflicting Updates will be presented to the User

- Bayou will not lock the file or file volume.

Replica Consistency:

- Bayou Guarantees that all servers eventually receive all writes via pair anti-entropy process.

- Writes are performed in the same well defined order at all servers.

- Conflict Detection and Merge Procedures are deterministic.
1. **Tentative State:**
   - Initially when the write is accepted by the Bayou Server.
   - Tentative Writes are ordered according to the time stamps assigned to them by accepting system.

2. **Committed State:**
   - Eventually Each write is committed.
   - Committed writes are ordered according to the times at which they commit and before tentative writes.
   - Timestamps are monotonically increased at each server.
   - Pair of<Time Stamp, ID of the server that assigned it> produce a total order of write operation.
Figure 4. Bayou Database Organization
O Vector
Each server maintains this timestamp vector to indicate in a compact way the omitted prefix of the committed writes.

C Vector
Characterizes the state if the Tuple Store after executing the last committed Write in the Write Log

F Vector
Characterizes the state after executing the last tentative Write in the Write Log

Enable server pairs to identify the sets of writes that need to be executed

These timestamp vectors are not used for conflict detection
Data Base should support:

1. Efficient Write Logging.
2. Efficient Undo/Redo Write Operations.
5. The Undo log should facilitate the rolling back of tentative writes that has been applied to the store.
6. Two distinct views of Bayous data base (Committed and Full (C+T))

Server can discard a write from write log once it becomes stable.

Each server maintains a time stamp vector.

The running state of each server includes two time stamp vectors that represent committed and full view.
A user may be granted:
- R/W privileges to data collection.
- Server privileges to maintain a replica.

Mutual authentication and access control is based on public key cryptography.

Every user possesses a public/private key pair and a set of digitally signed access control certificates.

Types of certificates to grant, delegate and revoke access:
- \(AC[PU,P,D]\) certificate that grants privilege P on data D to user whose public key is PU
- \(D[PU,C,PY]\) certificate signed by users whose public key is PY to delegate his privileges encoded in certificate C to another user whose public key is PU
- \(R[C,PY]\) certificate signed by users whose public key is PY to revoke some user’s privileges encoded in certificate C
EVALUATION - SETUP

- Server and Client for a bibliographic database.
- Data Base contains a single table of 1550 tuples.
- Each tuple was inserted into DB with a single Write operation.
- Tested on 2 different Servers:
  1. Running on Sun SPARC/20
  2. Gateway Liberty Laptop with Linux.
- Language independent RPC package developed at Xerox PARC is used for communication between Bayou clients and servers.
- Results were presented for 5 different configurations of the DB characterized by the number of tentative writes.
### 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><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>

### 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>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>
<tr>
<td></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>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>
<tr>
<td></td>
<td>6.04</td>
<td>7.05</td>
<td>7.01</td>
<td>6.4</td>
<td></td>
</tr>
</tbody>
</table>

### 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>Write: no conflict with conflict</td>
<td>159 (32)</td>
<td>212 (29)</td>
<td>177 (22)</td>
</tr>
<tr>
<td>Write: no conflict with conflict</td>
<td>207 (37)</td>
<td>372 (17)</td>
<td>223 (40)</td>
</tr>
</tbody>
</table>
Advantages

- Client can read or write to any replica without explicit coordination with other replicas.

- **Highly available:** Bayou will not mark conflict Data or System Unavailable.

- Bayou also provides support for clients that may choose to access only stable data.

- Dependency and Merge Procedures are more general than previous techniques.
Disadvantages

- No transparent, replicated data support for existing file systems and Data Base applications.

- Applications should:
  - Be aware that they may read weekly consistent data.
  - Be aware that their write operations may conflict with other applications.
  - Involve in detection of resolution of conflicts.
  - Should exploit domain specific knowledge

- Expensive Merge Procedure
Discussions

- Too Abstract!!
- Why only pair-wise communication?
- Global Strategies not possible during merging.
- Eventual Consistency!!
- When should we commit?
Thank You 😊