Outline

ACID Transaction

- Transaction has to show up everywhere at the same time or not at all.
  - E.g. when you withdraw cash from your ATM machine, the balance should reflect the actual money left. If it doesn’t, then you could go back to a store, use your ATM card and withdraw cash that you do not have.
  - E.g. when you make your Airline reservation and the system assigns you a seat, you expect the seat to be available to you (of course, Airlines overbook).
  - E.g. When students register at OASIS, it has to register enough students (upto the class limit).
Replication and Availability

- Replication is a powerful tool that allows us to trade off availability for consistency.

- Applications need different levels of consistency.

- Applications know best on how to deal with inconsistency.
Application 1: Meeting room scheduler

• Suppose we have two conference rooms of the same capacity. I want to schedule my meeting in one of the conference rooms. I don’t care which exact room it is.

  – If two people reserve the same room at the same time, there is a conflict, but if they reserve the same room at different times or reserve different rooms at the same time, there is no conflict.

<table>
<thead>
<tr>
<th>Rm1</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rm2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No conflict
Application 1: Meeting room scheduler

- Suppose we have two conference rooms of the same capacity. I want to schedule my meeting in one of the conference rooms. I don’t care which exact room it is.
  - If two people reserve the same room at the same time, there is a conflict, but if they reserve the same room at different times or reserve different rooms at the same time, there is no conflict.
Application 1: Meeting room scheduler

• Suppose we have two conference rooms of the same capacity. I want to schedule my meeting in one of the conference rooms. I don’t care which exact room it is.
  – If two people reserve the same room at the same time, there is a conflict, but if they reserve the same room at different times or reserve different rooms at the same time, there is no conflict.
Application 1: Meeting room scheduler

- We can lock the entire database
  - Not needed when there is no conflict
  - In the case of conflicts, there is an application specific way to deal with the conflict – we move one reservation to the other room.
Application 1: Meeting room scheduler

- We can lock the entire database
  - Not needed when there is no conflict
  - In the case of conflicts, there is an application specific way to deal with the conflict – we move on reservation to the other room
  - If the other room is reserved, we ask the user, they can easily move the reservation to another acceptable time
Application 2: Shared mailbox

- Shared mailbox folders - shared between, me, my TA, my secretary
- We all replicate the mailbox.
  - OP1: I see a mail from the class, respond to it and delete it
  - OP2: The TA sees the same mail and files it in UbiqComp
  - OP3: I see a mail from my wife and file it in important
Application 2: Shared mailbox

• All of us operate on the same mailbox
• You can lock the entire mailbox before we operate
  – Can’t work when disconnected
  – Clearly not necessary for operation OP1 or OP2 and OP3
  – For operation OP1 and OP2, it is no clear who should win, should the mail be deleted or should it be filed in UbiqComp?
Two approaches to building replicated services

• Transparent replication system:
  – Allow systems that were developed assuming a central file system or database to run unchanged on top of a strongly-consistent replicated storage system (e.g. Harp)

• Non-transparent replication system:
  – Relaxed consistency model – access-update-anywhere
  – Applications involved in conflict detection and resolution. Hence applications need to be modified (e.g. Bayou, Coda file system etc)
Hypothesis

• Applications know best on how to resolve conflicts
• Challenge is providing the right interface to support cooperation between applications and their data managers
  – Programmers do not want to deal with propagating updates, ensuring eventual consistency
    • Anyone who has synchronized the project files in school, work, and home can feel the pain..
  – Programmers want to set replication schedules and control how conflicts or detected and resolved
    • Record level conflict detection rather than file level
Bayou

• Update-anywhere replication model:
  – Bayou manages databases that can be fully replicated at any number of sites
  – Applications can read and write to any single replica of the database (lazy group update)
  – Once a replica accepts a write operation, this write is performed locally and propagated to all other replicas via pair-wise reconciliation protocol
Bayou Write

3-tuple: \(<\text{update}> <\text{dependency check}> <\text{mergeproc}>\)

For example,

Update: \(<\text{insert}, \text{Boyd527}, 2/17/2002, 11:15\text{am}, 50\text{min}, \text{“Ubiquitous Computing”}>\)

Dependency check: \(<\text{Is there an entry in Boyd527 database on 2/17/2002 at 11:15am for 50 min? expected answer is empty}>\)

Mergeproc: \(<\text{Try 2/17/2002 at 12:15pm for 50min; if successful write that tuple}>\)

• sometimes users like conflicts
Pair-wise reconciliation

Eventual consistency

Global commit order assigned by Primary server
Replica management

- New replicas are cloned from existing replicas
- Information about new replicas propagate with normal writes ("creation write")
- Information about retired replicas also propagate with normal writes ("retirement write")
- Replicas are created by users or sys. Admins.
- Replication schedule is controlled by users
- Sometimes users want to choose specific replicas
- Tentative and committed data
  - Applications can choose to read tentative or committed data
Technology impact

- TrueSync - end-to-end synchronization software and infrastructure solutions for the wireless Internet
  - [http://www.starfish.com/](http://www.starfish.com/)

- SyncML - SyncML is the common language for synchronizing all devices and applications over any network.
  - Ericsson, IBM, Lotus, Motorola, Nokia, Palm Inc., Psion, Starfish Software etc. (614 companies)
  - [http://www.syncml.org/](http://www.syncml.org/)