Designing a global EMail repository using OceanStore

Steven Czerwinski, Anthony Joseph, John Kubiatowicz

Summer Retreat
June 11, 2002
UC Berkeley
Goal

- Build an another interesting OceanStore app
  - Further tests OceanStore client APIs
  - Examine conflict resolution strategies
  - Explore data clustering/introspection techniques

- Email infrastructure using OceanStore
  - Fulfills the above criteria
  - Enables mail delivery, retrieval, and organization
  - Legacy access methods (IMAP, SMTP, POP)
System architecture

OceanStore Infrastructure

- Native client
- Client Object API
- Client Host
- IMAP OS Proxy
- IMAP Client
- Jon Smith’s mail object
  - Messages, flags, folders
- Jon Smith’s mail drop object
  - Incoming mail queue
- SMTP OS Proxy

Benefits of using OceanStore
- High availability of data
- Automatic archiving
- Objects migrate towards client
OceanStore API

• Stores objects
  – Composed of encrypted blocks

• Objects addressed through...
  – GUID (which object)
  – Key (which component of an object)
  – Byte range (which bytes of the component)

• Changing an object
  – Update, expand, truncate

• Predicates on changes
  – What must hold true for a change to succeed
  – Example: object version must equal 1.2

• Updates within an object are atomic
• Notified when predicate fails (conflicts)
Designing OceanStore applications

• Three important areas to consider
• Object layout
  – Defines OceanStore usage
  – Affects conflicts and migration
• Conflict detection and resolution
  – Minimize false conflicts
  – Roll back and retry to resolve
• Migration strategy
  – Enable clustering for pre-fetching
Object design

• Information to store
  – Messages, folder listings, flags, envelopes

• Object granularity choices
  – Per message
  – Per folder
  – Per user

• Per user granularity
  – Atomic actions only involve one object
  – Minimizes number of objects

• Separate object for user mail drop
  – Requires different security/permissions
Object layout

Selection

Btree Key  Byte Range

Hash("List")  Hash("Ostore")

Folder List

<table>
<thead>
<tr>
<th>FOLDER</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ostore</td>
<td>AAF</td>
</tr>
<tr>
<td>Spam</td>
<td>BFA</td>
</tr>
</tbody>
</table>

Folder: Ostore

<table>
<thead>
<tr>
<th>UID</th>
<th>FLAGS</th>
<th>MSG_KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Delete</td>
<td>AF4594A</td>
</tr>
<tr>
<td>46</td>
<td>Reply</td>
<td>BF564AA</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>GH453AB</td>
</tr>
</tbody>
</table>

Message

Envelope

Contents

Block Aligned

OS Mail Account Object
Conflict detection and resolution

• Sources of conflicts
  – Mail arrival while user browsing (more likely)
  – Tentative updates delayed too long
  – Clients updating same account (less likely)

• Possible conflicts
  – Updates on same message
    • Changing different flags, moving to different folders
  – Updates to same folder
    • Appending new messages, deleting a folder
  – Updates on folder list
    • Create new folders with same name
Conflict resolution design

- **General strategy**
  - Upon detection, application reads offending state
  - Compares state w/ tentative to classify conflict
  - For false conflicts, simply retry
  - For true conflicts, take most conservative action
  - Works for most, but there are special cases

![Diagram of IMAP Client and OS IMAP Proxy interaction]

- **State Comparison**
  - Tentative State:
    - Tentative State #34 -> \High
    - Committed State #50 -> \Low

- **OS IMAP Proxy**
  - Update INBOX,
    - Predicate: version =1.1
  - Fail, version is 1.2
  - Update INBOX,
    - Predicate: version =1.2

- **OceanStore**
Special case: Assigning UIDs

• Unique Identifiers (UIDs) in IMAP
  – Each UID maps to a message
  – Mappings are permanent
  – UID namespace is per folder
  – Invariant: UID numbers are strictly increasing, ordered by addition time

• Possible conflicts:
  – Same UID used for two different messages
  – Msg added with UID smaller than largest in folder
  – Both cause clients to miss messages
Conflict example: Assigning UID

1. **IMAP Client #1**
   - Fetch 5:* (new msgs)
   - Fetched 5, 6, 7
   - Delete 6, Fetch 8:*
   - State: No msg 6, new msgs after 7

2. **OS IMAP Proxy #1**
   - Tentative
   - Proxy 1: Retrieves msgs from drop, updates account object to assign UIDs 5-7.

3. **OS IMAP Proxy #2**
   - Tentative
   - Proxy 2: Assigns UIDs 5-6 to msgs moved from Ostore

4. **IMAP Client #2**
   - Move Ostore: 12, 13 to INBOX
   - OK

**OceanStore**

- Fails!
- Succeeds!

**Client #1’s state is incorrect:** No msg 6, wrong msg 5

**Move Ostore: 12, 13 to INBOX**

**OK**
UID assignment algorithm

- Algorithm replicated at all clients
  - Defer assigning UIDs
  - Rollback back conflict by "deleting" messages
  - Re-add messages with higher, non-conflicting UIDs

<table>
<thead>
<tr>
<th>UID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>AEF</td>
</tr>
<tr>
<td>6</td>
<td>BD4</td>
</tr>
<tr>
<td>7</td>
<td>C45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>AEF</td>
</tr>
<tr>
<td>9</td>
<td>BD4</td>
</tr>
<tr>
<td>10</td>
<td>C45</td>
</tr>
<tr>
<td>11</td>
<td>AA4</td>
</tr>
<tr>
<td>12</td>
<td>BB4</td>
</tr>
</tbody>
</table>

IMAP Client #1

OS IMAP Proxy#1

Committed State

Tentative Update
Block clustering

• Designed pre-fetching behavior
  – Pre-fetch message envelopes
  – Priority to envelopes in INBOX
• But, OceanStore cannot see these attributes
• Initial strategy
  – Provide hints through object’s metadata
  – Assign type ids to blocks (envelope, msg body, …)
  – Assign referrer ids to blocks (INBOX, Ostore, …)
• Should improve pre-fetching ability
Example block hints

Folder List

- **Ostore** | AAF
- **Spam** | BFA

Type: Folder List (0)
Referrer: ROOT (0)

Folder: Ostore

<table>
<thead>
<tr>
<th>UID</th>
<th>FLAGS</th>
<th>MSG_KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>\Delete</td>
<td>AF4594A</td>
</tr>
<tr>
<td>46</td>
<td>\Reply</td>
<td>BF564AA</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>GH453AB</td>
</tr>
</tbody>
</table>

Type: Folder (1)
Referrer: List (1)

Message

Envelope

Contents

Header

Body

Type (2)

Referrer: Ostore (50)

OS Mail Account Object
Future work

• Implementation this summer
• Re-design with new client API features
  – Finer grain conflict detection
  – Support for data clustering hints
• Experiments
  – System scalability
  – Bandwidth usage measurements
  – Clustering effectiveness
Some thought questions

- Should we have objects per message?
  - Implement sharing of messages (mailing lists)
- Should we only have one SMTP server?
  - Without active checks, open to spam
- How feasible is disconnected operation?
- Can we generalize our introspective solution?