Managing Versions of Web Documents in a Transaction-time Web Server

Curtis Dyreson, Hui-Ling Lin, and Yingxia Wang
School of E.E. and Computer Science
Washington State University
USA

WWW 2004 Conference

Outline

• Motivation
• Server extension
  • History of a site
  • Time travel queries
• Experiments
  • Turnaround time
  • Disk/memory usage

Motivation

• Sep. 25, 2000 Cathy Freeman won the 400m
• Sep. 26, 2000, article in The Australian
• URL of the on-line newspaper (sports section)
Queries

- URL of the on-line newspaper (sports section)
- URL understood by transaction-time server

Motivation

- Historical access
  - Entertainment, government, educational, financial
- Time travel queries
  - Timeslice
  - Next/previous versions
  - Document history
  - Show changes
    - HTMLdiff, XMLdiff (AT&T Bell Labs)
  - Statistics on frequency of page update
    - Notulas, Cho, and Olsotn (WWW 2004)
- Archives
  - A warehouse for old pages
  - No standard interface

Related Work

- Web Archiving
  - The Internet Archive (www.archive.org)
    - Wayback machine
  - iPROXY
    - Rao, Chen, and Chen (Middleware Symposium, 2000)
  - Author-requested archiving
    - Douglis (Workshop on Web-site Evolution, 1999)
- XML
  - Change detection
    - Marian, Abiteboul, Cobena, Mignet (VLDB 2001)
    - Butler, Rocco, and Liu (WWW 2004)
  - Transaction-time querying of XML with TTXPath
    - Dyreson (WISE, 2001)
  - Fabio Grandi’s temporal web bibliography

How long before resource was updated?

<table>
<thead>
<tr>
<th>Time</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>one year</td>
<td>4%</td>
</tr>
<tr>
<td>one month</td>
<td>5%</td>
</tr>
<tr>
<td>one week</td>
<td>10%</td>
</tr>
<tr>
<td>one day</td>
<td>14%</td>
</tr>
<tr>
<td>one hour</td>
<td>9%</td>
</tr>
<tr>
<td>one minute</td>
<td>32%</td>
</tr>
<tr>
<td>unchanged</td>
<td>26%</td>
</tr>
</tbody>
</table>
A Transaction-time Web Server

Transaction Time in Databases

- Interval representation - DBMS maintained

<table>
<thead>
<tr>
<th>Stock</th>
<th>Price</th>
<th>Transaction Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>34</td>
<td>Sep/8/2002 now</td>
</tr>
<tr>
<td>Microsoft</td>
<td>72</td>
<td>Sep/9/2002 now</td>
</tr>
<tr>
<td>TriGEO</td>
<td>105</td>
<td>Sep/9/2002 now</td>
</tr>
</tbody>
</table>

- On Sep/9/2002
  
  DELETE FROM Stocks WHERE Stock='TriGEO';
  INSERT INTO Stocks ('TriGEO', '105');

- Schema evolution (e.g., Roddick and Snodgrass, TSQL2, 1995)

Web vs. Database

- Must use DBMS to modify data

  ![Diagram](image)

- On the web, updates are independent of server

  ![Diagram](image)

An Observant System

- Each edit creates a version
- Read-only (read-mostly) access to the data
- Known vs. assumed

  ![Diagram](image)

Observant Systems

- Lots of Observant Systems
  - Web browsers, HTTP servers, portals, search engines
- HTML/XML
  - No temporal semantics
  - Few user-defined timestamps
- Goal: Implement transaction time in an Observant System
  - No transactions
  - No explicit timestamps in data
  - Implicit
    - System time of observer
    - File modification time

Constraints on Our Design

- Backwards compatible
  - Minimal changes to HTML, HTTP, servers, browsers
  - No changes to legacy pages
  - No changes to page maintenance culture
- Coexistence compatible
  - Partial migration
- Simplify problem
  - Ignore dynamic pages
    - ASP
    - CGI-bin
  - Ignore processing changes
    - Javascript bug fixes
**Lazy Transactions**

- Perform transaction during HTTP get

**Computing Resource Versions**

- Known vs. believed versions

**TTQueries**

- Syntax: `a.htm?q1, p2`  
  - A selection query(q1) and restructuring query(p2)

**Examples**

- Show current version, but links are to previous
  `a.htm?now, pre`
- Show all versions of seattle.html
  `a.htm?history`
- Grab version on 26 April, links are to current
  `a.htm?26-Apr-2002/02:15:04, now`
- Two versions ago
  `a.htm?pre.pre`

**Vacuuming**

- Remove versions
  - Vacuuming policies
    - time-window, version-window, periodic, percent-difference
      - e.g., Vacuum versions that differ by less than 5%
      - Never vacuum the current version
  - Remove history
  - Obliteration
  - Query repair

**Percent-difference Vacuuming Policy**

- Vacuum versions less than 30% difference

**Query Repair**

- What happens on query to vacuumed version?  
  - E.g., redirect to previous/next "good" version
Vacuuming Policy Tree

- Policy can be set for directory or document

```
  home directory
  |      |
  |      |
  +-----+-----+-----
  |   customers    |   merchants    |
  +-------------+-------------+
  |   info.html  |   order.html |
  +-------------+-------------+
  |   alert, yes, v-window(current-10) |
  +-------------+
  |   alert, no, earliest-diff(30) |
  +-------------+
  |   alert, no, no alert, no, earliest-diff(30) |
  +-------------+
  |   alert, no, periodic(2,5) |
  +-------------+
  |   alert, no, no alert, periodic(3,7) |
  +-------------+
  |   alert, yes, t-window(begin=6, now) |
  +-------------+
  |   alert, no, earliest-diff(30) |
  +-------------+
  |   past, yes, no alert, earliest-diff(30) |
  +-------------+
```

Experiments

Implementation

- Apache
  - Popular, open source server
  - Pre-forking model (child processes handle requests)
    - DB must have concurrency control
  - “inner-loop” – additional disk I/O on every request
- BerkeleyDB

```
  Apache Architecture
  Online user
  Resource owner
```

Cost

- Time
  - Overhead on I/O
    - Minimum – One DB read and write
    - Maximum - Many DB reads and writes, many file copies
    - Milliseconds are important (disk read approx. 11 msec)
- Space
  - Disks are cheap
  - Vacuuming
  - Store diffs
    - RCS

```
  Experiments
  - Factors
    - Page size (1KB vs. 60KB)
    - Update rate (0%, 1%, 2%, 5%, 7%, 10%, 15%, 50%, 100%)
    - Ratio of TTQueries (0%, 1%, 5%, 20%, 80%)
      - “pre”, “a timestamp”, “pre.pre.pre”, “history”
  - Design of the experiments
    - Step1. Start TTApache
    - Step2. Pre-fetch 3000 pages
    - Step3. Perform multiple runs
      - Measure turnaround times (3000 requests/run)
    - Step4. Shut down TTApache
```
Summary of Results

- Little overhead for low update rates
  - No significant difference for rates under 5%
  - EECS web site - .009%
- A large page size is a significant factor.
  - Increases memory use and disk I/O
    - EECS web site
      - 71% - less than 1K
      - 23% - 1K to 10K
      - 4% - above 10K
  - TTAPache "stress test"
    - 60KB page at 100% update => 20 requests per second
    - EECS web site
      - 0.68 requests per second (on average)
      - EECS - 73% of peaks > 20

Conclusions

- Local server extension
  - Archives documents
  - Supports
    - Time travel queries
    - Link rewriting
    - Vacuuming
    - Obliteration and forwarding of resource histories
  - Distinguishes known vs. assumed versions
- Compatible
  - HTTP, HTML, URLs
  - Page maintenance "culture"
- Cost
  - Modest overhead