Transactional Web Archives

Robert Sanderson
Lyudmila Balakireva
Harihar Shankar
Herbert Van de Sompel

Los Alamos National Laboratory Research Library

Web Archive Globalization
JCDL 2011, Ottawa
Jun 16-17, 2011

This research is funded by the Library of Congress
Transactional Web Archiving

- Transactional Archiving?

- Server Side Capture
  - Submission, Storage, Access

- Browser Side Capture
  - Submission, Storage, Access

- Memento for Access
Transactional Archiving?

- Current web archives actively crawl the web

- For example, Heritrix from the Internet Archive and the many archives that use it
Transactional Archiving?

- Transactional archives passively accept submitted HTTP transactions between browser and server

  ![Diagram of transactional archiving](image)

- For example, TTApache, PageVault and Everlast.
Why Transactional Archiving?

- Issues with crawler based archiving:
  - Can be rejected (robots.txt, by user-agent, by host IP)
  - Can be deceived (cloaking: geo-location, by user-agent)
  - Can be trapped (infinite auto-generated pages)
  - Don't necessarily capture well used resources
  - Require constant and massive bandwidth

- None of these are true for Transactional Archiving …

… but, it has its own different set of challenges
Transactional Archiving?

- Need to record transactions between browser and server
  - Server side: Servers to be archived must cooperate
  - Browser side: Many browsers must cooperate

- Need to transfer data to archive: either batch mode or real-time
- Archive must trust submission to be authentic

- Deduplication challenges as can't control what will be submitted:
  - Aliases: Different URL, same response
  - Negotiation: Same URL, different response
  - Determine "significant" change in response
  - Other factors for what to archive/throw away?
Server Side Capture

- Approach:
  - Willing server records the request and response headers and response body just before returning to the browser
  - Server sends to an archive for storage
Server Side Capture/Submission

- Developer: Luda Balakireva

- Capture Implementation
  - Apache connection filter module implemented in C to trap URL, headers and response body
  - Module POSTs to a configurable URL in real time

- Submission Implementation
  - Java/Grizzly+Jersey for handling submission interface
    - Can also be deployed under tomcat or glassfish
  - BerkeleyDB for storing metadata
  - Headers and response body data stored in file system
Server Side Capture

- Direct server to server upload, in real time:
  - Most configurations will have server/archive in close network proximity
  - Reduces wait time between observation and being discoverable in archive
Server Side Capture: Issues

- If archive is not local, network latency may be an issue
  - But could be amortized by batch upload

- Size of dataset could very large for dynamically generated pages
  - But could be reduced by better detection of high value changes compared to counters, timestamps, etc.

- Content Negotiation problematic!

- Capture of pages with “attack vector” query params
  - index.html?f=/etc/passwd
Browser Side Capture

- Approach:
  - Willing browser records the request and response headers and response body after receiving from server
  - Browser sends to an archive for storage
Browser Side Capture/Submission

- Developer: Rob Sanderson

- Capture Implementation
  - Firefox add-on captures headers and body and writes to temporary storage on local disk
  - After configurable amount of data stored, module compresses and moves to a shared Dropbox folder for batch upload
  - (Limited) Ability to detect and ignore private data

- Submission Implementation
  - Dropbox used as transfer, temporary storage mechanism
  - Python monitor system on top of Dropbox
  - Cassandra (NoSQL hash store) for storing metadata
  - Response body and headers stored in pair-tree file system
Browser Side Submission

- Reasons for Dropbox rather than direct upload:
  - Batch upload via existing infrastructure reduces bandwidth
  - Increases Firefox responsiveness
  - Batch processing can be scheduled as needed
Browser Side Capture/Submission

Memento wants to make it easy to . . .

Our Memento ideas have attracted quite some attention since we first shared them in November 2009. And a lot has happened since then. Stay up-to-date by checking in on Memento news here.

Preferences

Upload

Public/Private

Status Icon
Browser Side: Issues

- Privacy! Privacy! Privacy!
  - Difficult to determine if resource should be captured or not
  - Current approach:
    - No HTTPS
    - Check for “log out”, “sign out” etc in body
    - Check for usernames, personal name in body, headers
    - Blacklist for domains

- Bandwidth
  - Slow-down while uploading batch file noticeable on home connections
Memento in One Slide

Browser

Original Resource

TimeGate

Memento Archive

Do you have a preferred TimeGate?

Yes, G // No, use a default

Where is the archived copy for the time that I want?

It's at M // I don't know, please try another TimeGate

Please give me the archived copy

Here it is
Access via Memento

- Both archives provide Memento TimeGates for access

- TimeGates can be used with MementoFox:
  - Must be configured with Dropbox archive TimeGate
  - Processes every HTTP request, not just HTML page

- Distributed access is intentional design feature
  - Possible to construct views from multiple archives:
    Server side will have most authentic copy, but may embed image from another server, only in Dropbox archive
Server Side Archive: Access

- Access to archive via Memento TimeGate
  - Implemented in Grizzly server using Jersey library
- Original Server uses HTTP Link header to point to archive

- Export functionality also available to WARC format to extract data in batch mode
  - By datetime of last update
  - By URL
Browser Side Archive: Access

- Apache/Python Memento TimeGate for access
  - Archive provides combined, anonymous TimeGate
  - Also provides per-user TimeGates to see own archive
  - Per-User currently secure only through obscurity
  - Export functionality also yet to be implemented
Access via Memento

Experimental Transactional Archive

TimeGate Preferences

Memento: Transactional Archiving
Web Archive Globalization Workshop, Jun 16-17 2010
Summary

- Implemented and tested two types of Transactional Archive:
  - Server Side
  - Browser Side
- Transactional Archives lack many of the challenges of Crawler based Archives (but have their own)
- Implemented Memento TimeGates for Transactional Archives:
  - Does not require rewriting URIs for self-contained-ness
  - Works well with automated, distributed access patterns
- Access via Browser add-on is fast and seamless
- Server and Browser archiving code will be released
Memento wants to make Navigating the Web’s Past Easy

Learn: http://www.mementoweb.org/
Talk: http://groups.google.com/group/memento-dev