Surfing the Web through OceanStore

Patrick R. Eaton
University of California, Berkeley
eaton@cs.berkeley.edu

January 16, 2002
Motivation

- We believe that it is important to study how OceanStore interacts with real applications.
- OceanStore must support legacy interfaces.
  - see the rise of web browsers and HTTP
- Legacy interfaces can be enhanced by OceanStore features.
  - archival storage
  - time travel
  - promiscuous caching
  - tolerance to machine failure
Goal

• Develop an OceanStore web caching architecture.

• The architecture should
  – support legacy HTTP interface and web clients
  – provide a migration path to an OceanStore-native web
  – use OceanStore features to enhance the web experience
Components of the OceanStore Web Caching Architecture

- **Client proxy.**
  - provide a user’s web browser with a connection to cached content

- **Passive caching agent.**
  - retrieve “hot” documents and cache them in OceanStore

- **Active caching agent.**
  - cache content pro-actively directly from content provider on all updates

- **Spidering caching agent.**
  - crawl the web in search of documents to add to the cache

- **Reverse proxy.**
  - allow “ancient” web clients to access data in the native OceanStore web
Client Proxy

- Serve as translator to convert HTTP messages into OceanStore messages.
- Check the OceanStore web cache for requested documents.
- Retrieve uncached documents via the legacy protocol.
- Provide the passive caching agent with hints on popular documents.
Passive Caching Agent

- Use hints from client proxies to determine popular content.
- Retrieve popular content using standard HTTP protocols.
- Store the content in the OceanStore cache.
Spidering Caching Agent

- Obvious extension to the passive caching agent.
- Crawl the web searching for content that can be added to the cache.
Active Caching Agent

- Used by content providers who want to ensure that their content is cached in OceanStore and available to their users.

- Providers proactively push content updates to the agent.

- Agent stores content in OceanStore web cache.

- Service could be based on contractual agreement.
  - Content can be signed to prevent spoofing
  - Contract could specify minimum number and location of replicas
Reverse Proxy

- Allow traditional web clients to access documents that are published natively to OceanStore.
- Sit at the IP address published for a server.
- Translate incoming HTTP request to OceanStore requests.
- Convert the OceanStore content to an HTTP response to serve to the user.
The Whole Web Cache Architecture
Success Metrics

• Improved browsing experience.
  – lower latency for end user

• Reduce server load.
  – reduce load on the web servers of content providers

• Technical metrics.
  – amount of cache sharing
  – effectiveness of caching content close to users
Issues - Document Freshness

- OceanStore stores all versions of a document that has ever been cached.

- **Issue:** How do we request a version that is still fresh enough (by HTTP cache requirements) without extra round-trips or reading unnecessary data.

- **Solution:** Include *version predicates* in read requests.
  - this parallels the predicates sent with updates
Issues - Negative Tapestry Results

• When Tapestry delivers a result, it can be checked because all data is verifiable.

• **Issue**: What does a negative result mean?
  – document does not exist

• **Answer**: Negative results are only hints.

• **Issue**: Are non-verifiable hints useful?

• **Answer**: Yes, negative hints simplify the application programming task and are vital for application performance.
Status

• Client proxy and passive caching agent prototypes are functioning.

• Client proxy
  – accepts HTTP requests from user’s browser
  – searches the OceanStore cache for a fresh copy of the document
  – converts the OceanStore result into a HTTP response
  – retrieves uncached documents from the origin server using HTTP
  – delivers HTTP response to user’s browser

• Passive caching agent
  – accepts hints from client proxies
  – retrieves documents from origin servers using HTTP
  – creates OceanStore objects for content being cached for first time
  – updates the cache with the current content of the document
Status (continued)

- What is missing?
  - strict adherence to HTTP caching directives
  - Tapestry timeouts and negative hints
  - multiple client tests
Preliminary Results

- I present these numbers only to emphasize that we have a working system. The system is not yet widely deployed or tuned for performance.

<table>
<thead>
<tr>
<th>Time to Retrieve a Document</th>
<th>1K</th>
<th>4K</th>
<th>16K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department Web Server</td>
<td>160 ms</td>
<td>982 ms</td>
<td>751 ms</td>
</tr>
<tr>
<td>OceanStore Web Cache (remote reads)</td>
<td>286 ms</td>
<td>647 ms</td>
<td>438 ms</td>
</tr>
<tr>
<td>OceanStore Web Cache (local reads)</td>
<td>4.3 ms</td>
<td>7 ms</td>
<td>7 ms</td>
</tr>
</tbody>
</table>
Future Work

- Implement all of HTTP’s caching policies.
- Perform larger scale usability tests using others in the department.
- Use an active caching agent to push content from the department’s web server into the OceanStore cache.
- Crazy Idea: Initial results indicate sub-10 ms cache hit times for content on the local node. Could the OceanStore web cache replace the local browser cache?
Conclusions

• We have presented an architecture for the OceanStore web cache.

• A prototype implementation of the OceanStore web cache has been developed.

• It is productive to examine the interaction of applications with OceanStore during the design of the system.
Issues - Key Management

• Every principal who stores data must have an identifying key. Any client that has a key to decode data can read an object.

• Issue: How are the keys organized in the web caching architecture?

• A passive caching agent needs a key to cache the content.
  – passive caching agent is billed for storing content
  – difficult to bill users
  – perhaps it is OK to avoid billing users since this matches the HTTP/web paradigm

• An active caching agent needs a key to cache the content.
  – active caching agent is billed for storing content
  – active cache can bill content providers who wish to push their content out - similar to paying for bandwidth
Pushing Content

- Scenario: A Super Bowl webcast of the plays and stats (not video) for each play. Content is updated every 25 seconds and needs to be pushed to several million viewers.

- Caching content in OceanStore has several advantages.

- Key feature is a number of mobile replicas arranged in a tree for rapid dissemination.
Pushing Content (continued)

- Advantages of caching fast-changing content in OceanStore:
  - no client starvation
  - replicas are created and moved to service hot spots
  - no load on server
  - rapid dissemination of updates via optimized dissemination tree
  - cache communication is implicit in the dissemination tree
  - no over-provisioning required (Victoria’s Secret fashion show)