‘Wayback’ for Accessing Web Archives

Brad Tofel
Internet Archive
4 Funston Ave
San Francisco, CA 94129, USA
brad@archive.org

ABSTRACT

'Wayback' is an open-source, Java software package for browser-based access of archived web material, offering a variety of operation modes and opportunities for extension. In its basic, usual configuration it can both list available URL captures by date and offer recursive archive browsing starting from any capture. Advanced configurations offer better performance for challenging archived material and improved navigation.

'Wayback' is implemented as a collection of loosely coupled alternate implementations of core modules, for which an overview of each is provided. The functionality and implementation is also contrasted with its inspiration and predecessor, the Internet Archive's classic public Wayback Machine software, and other ways of accessing archived web material. Finally, future directions for improvement are outlined.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services

General Terms

Management, Documentation, Design, Human Factors.

Keywords

Web archives, Wayback Machine, access tool, web browsing.

1. INTRODUCTION

Archiving of digital materials has grown rapidly over the past decade, and as the number of organizations using common tools and formats grows, the opportunities for creating and sharing additional software specialized for those formats grows as well. Many organizations have adopted the ARC file format [1] for preserving digital documents, which consists of many individual URL captures concatenated into aggregate files, each capture prepended with a small amount of metadata. While this format has demonstrated significant advantages in managing and processing very large-scale archives, it does not lend itself to human access.

The Wayback project’s goal is to provide a common tool for delivering this access, leveraging common code bases and techniques, while allowing sufficient flexibility to be useful in the varied application contexts and infrastructures found in the digital archiving community. Modularity of the system has been a fundamental design goal, enabling organizations to implement custom modules as needed to adapt the Wayback system to their specific infrastructure and access needs.

IWAW’07, June 23, 2007, Vancouver, British Colombia, Canada.

2. MAJOR COMPONENTS

Wayback consists of four primary components, each with several current implementations that can be combined to customize installations of varying scales and capabilities.

The four components are:

- **Query UI**: responsible for parsing user queries, executing them against the Resource Index, and rendering the results in lists or tables for end user consumption. Hyperlinks are typically included in the results, allowing users to navigate into the Replay UI.
- **Resource Store**: responsible for retrieving archived content, abstracting the medium and format used to store the web content.
- **Resource Index**: responsible for satisfying queries against the documents located in the Resource Store, either through URL based queries, full-text search, or other novel search mechanisms.
- **Replay UI**: responsible for altering the context of documents being viewed by users, so hyperlink references refer back into the Wayback system, allowing users to interactively browse the web “as it was”.

2.1 Component Implementations

2.1.1 Query UI Implementations

There are currently three implementations of this component for rendering the results of user queries in a specialized way. Specific fields parsed from user queries are forwarded transparently to the Resource Index, decoupling this module from various Resource Index implementations.

The three implementations are:

- **Classic Query UI.** This mode mimics the “calendar” query results view found in the Internet Archive’s classic public Wayback Machine [2], partitioning results by Year, Month, Day, or other time slices, and places all results within each partition in a dense column of links. It is effective at displaying a large number of results in a format that is comprehensible and easy to navigate.
- **Search Engine Query UI.** This mode provides search results in the familiar ranked, paginated method commonly used by search engines.
- **XML Query UI.** This mode provides results of user queries in a format easily read by other programs. This implementation is useful in providing a web service for end users, and is critical for hosts providing remote access to a local Resource Index in distributed Wayback installations (described later).

2.1.2 Resource Store Implementations

Presently there are two Resource Store implementations, both of which assume documents are stored in ARC file format. Providing a new Resource Store implementation can support other storage formats. Access to archived content is wrapped in a Wayback specific format for use by other components in the system.

The two current implementations are:

- **Local ARC Resource Store.** This provides simple access to content stored in ARC files local to the Wayback service. Using NFS, or any other network file system, this implementation can also provide access to ARC files distributed across multiple networked storage devices. This implementation also has the capability to use a background thread to poll the local file system and notice the appearance of new ARC files, automatically index their content, and forward records to the Resource Index for incorporation in the live Wayback index.
- **HTTP 1.1 Remote ARC Resource Store.** This implementation accesses individual documents via HTTP 1.1's range request feature from a single remote HTTP directory. Wayback comes with an application called the ArcProxy that maintains a database mapping ARC files to HTTP URLs. The ArcProxy forwards incoming HTTP range requests to the appropriate storage server, enabling Wayback to access records stored in millions of ARC files.

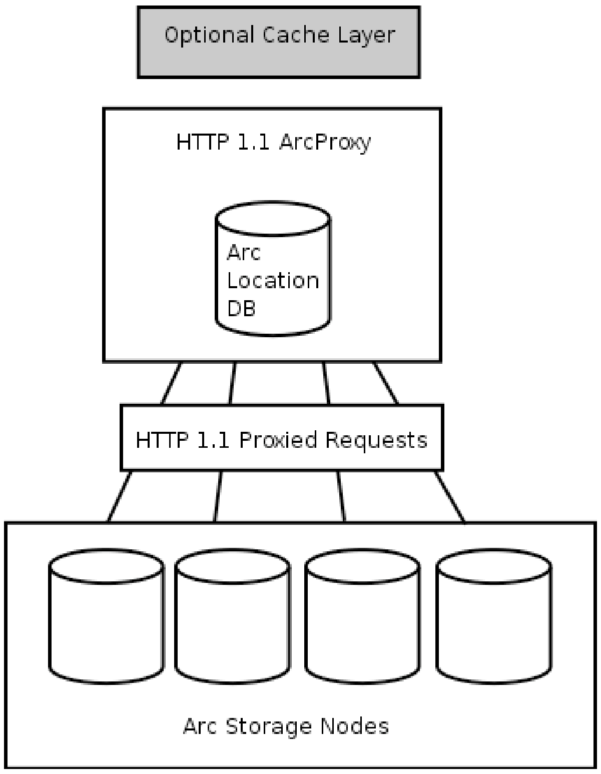

Figure 1: ARC Proxy Application

2.1.3 Resource Index Implementations

There are several Resource Index implementations, each with varying capabilities. Implementations are free to offer whatever search parameters they wish, but to satisfy the minimum requirements for the Replay UI, they must be able to map an URL and a capture date to a specific document in the Resource Store. Current implementations allow queries based on either an URL or on an URL prefix, and enforce primitive access control mechanisms.

The five current implementations are:

- **Local BerkeleyDB (BDB) Resource Index.** This implementation stores records in a Java-native database (BerkeleyDB Java Edition [3]), allowing records to be quickly inserted into the index, and immediately available for subsequent queries. This Resource Index supports only URL or URL prefixed based searching, and includes optimizations for URL-date queries for Replay UI access.
- **Local CDX Resource Index.** This implementation performs binary searches against a sorted, plain text file (called a 'CDX' [4] when indexing ARC files) to satisfy URL and capture date queries, just like the BDB Resource Index. These text files are maintained by a system outside the scope of the Wayback software. The primary advantage over the BDB implementation is size on disk, increased scalability, and explicit management and updating of the index.
- **Remote Resource Index.** This implementation accesses a remote BDB or CDX Resource Index by assembling an HTTP query against the remote index, and extracting the query results from the returned XML data. This allows separation of processing and storage of a

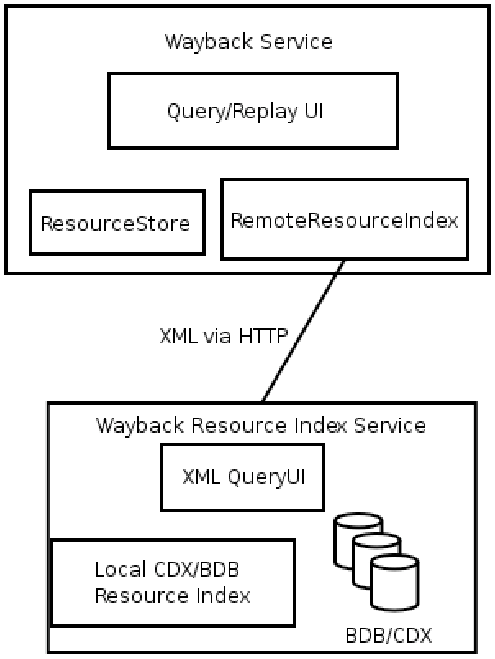

Figure 2: Remote Resource Index

Resource Index from the rest of the system.

- **Remote NutchWAX Resource Index.** This implementation allows an externally built NutchWAX [5] full text index to act as a Wayback Resource Index. This implementation is primarily responsible for marshaling Wayback queries into HTTP requests meaningful to a NutchWAX index, and then translating the results from their XML format into the abstracted result format that the other Wayback components use.
- **Alphabetic Distributed Resource Index.** This implementation uses a configuration file to map contiguous alphabetic partitions of the URL space to individual hosts responsible for each partition. This Resource Index then forwards each incoming request to the appropriate host, allowing a single logical index to span many hosts. This implementation allows responsibility for specific ranges to be assigned to multiple hosts, providing load balancing and fault tolerance to the distributed index.

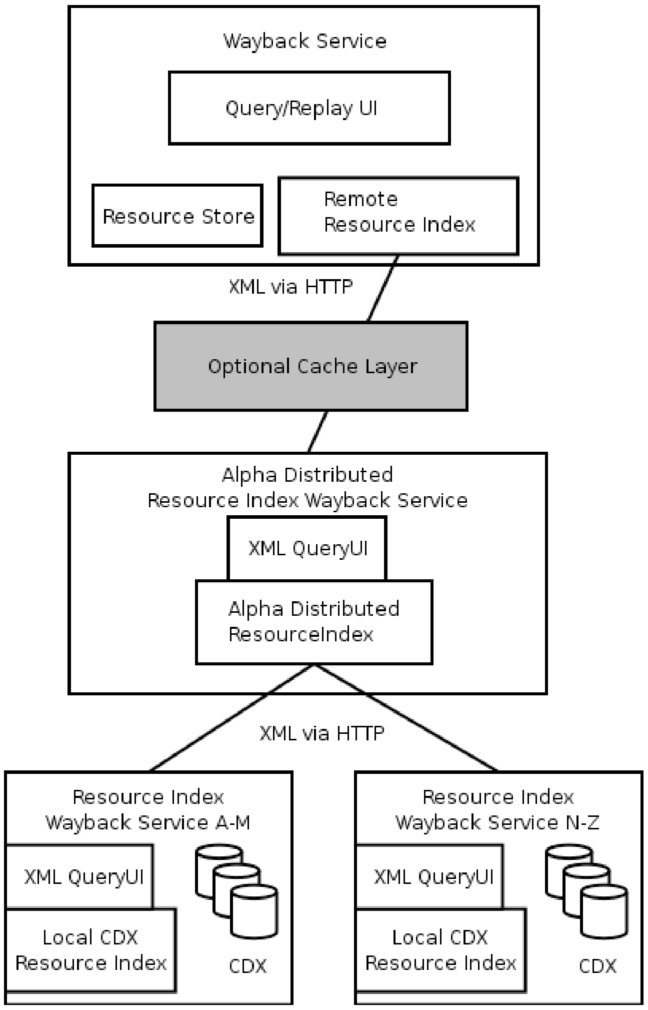

Figure 3. Alpha Distributed Resource Index

2.1.4 Replay UI Implementations

There are presently three implementations of the Replay UI component, each using a different strategy for altering the context of replayed pages hyperlink references to allow archived documents to be viewed correctly in web browsers.

The three implementations are:

- **Archival URL Replay UI.** This implementation mimics the current Internet Archive classic public Wayback Machine's mechanism for altering archived documents as they are served. Specifically, this includes altering HTML documents by rewriting FRAME:SRC attributes, updating or inserting BASE:HREF attributes, and inserting Javascript at the end of the document. This inserted Javascript executes in the client browser and is responsible for rewriting hyperlink anchors and embedded objects found in the page so they refer back into the Wayback Replay UI service. This mechanism suffers from several shortcomings, which are described later in this document.
- **Timeline Replay UI.** This implementation mimics the WERA [6] (see Related Work, below) user interface by providing a timeline navigation element at the top of each replayed document. This timeline allows users to navigate between individual archived versions of the current page without returning to the Query UI. The hyperlink context alteration methods used are the same as in Archival URL Replay mode.
- **Proxy Replay UI.** Using this implementation, the Wayback Replay UI acts as an HTTP proxy server. Users can configure their web browsers to proxy all requests through the Wayback service, which has significant advantages over other replay modes, in that no hyperlink context alteration is needed. Any hyperlink references found in replayed documents will automatically be requested though the Wayback service, including dynamic content, such as embedded Flash or Java objects, and hyperlink references constructed by Javascript found in the original page. However, casual web users may find enabling/disabling use of a remote web proxy for their archive access more challenging than the other mechanisms.

3. EXAMPLE DEPLOYMENTS

The Internet Archive has deployed the Wayback in several contexts, providing access to over 70 archived collections. There have been 3 primary deployment patterns used thus far.

3.1 Simple Standalone

The first deployment pattern is a simple, standalone application, using a Local ARC Resource Store and a Local BDB Resource Index to automatically make accessible new content discovered in a local ARC directory. This pattern has proved very effective for small-scale collections, where the Heritrix web crawler is run on the same server as the Wayback, and the Wayback is configured to automatically index ARC files, as they are crawled.

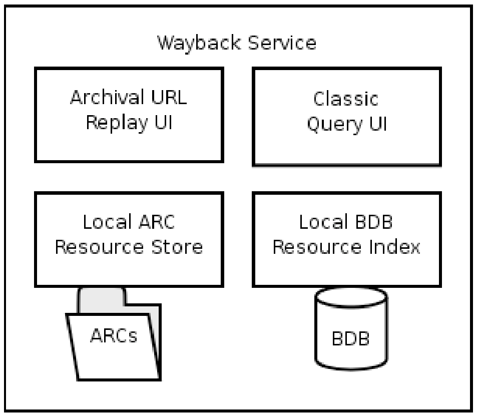

Figure 4. Simple Standalone Deployment

3.2 Multiple Replay Modes Sharing Resource Index and Resource Store

The second deployment pattern involves hosting many small collections on a single host, where each collection provides access to a collection of ARC files via all three current Replay UI implementations. In these contexts, a single Local CDX Resource Index is exported with an XML Query UI. All three Replay modes share the common remotely accessed Resource Index and Resource Store, allowing users to choose their preferred Replay mode while minimizing the server resources required.

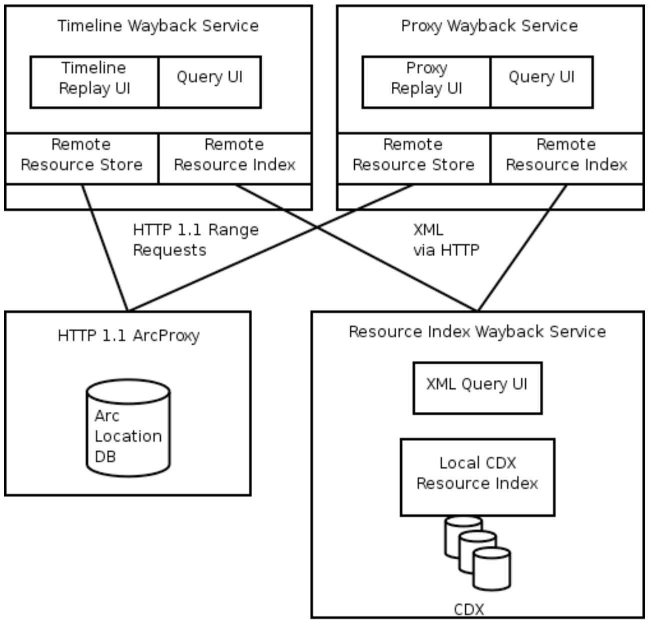

Figure 5. Alternate UIs Sharing Resource Index/Store

3.3 Large Scale Distributed Resource Index

The final deployment pattern used at the Internet Archive demonstrates a large-scale, distributed collection of over 140 million captured documents. In this configuration, ARC files are highly distributed across hundreds of storage machines, and are accessed with an HTTP 1.1 Remote ARC Resource Store. The Resource Index is broken into multiple, alphabetically partitioned CDX shards, and each of those shards is stored on a host that is only responsible for satisfying queries against that local index shard, responding with XML data. There is a single Alphabetic Distributed Resource Index server that accepts all queries, and forwards the queries on to the appropriate shard server.

This configuration allows the number of shards to grow with the number of documents being indexed, and also provides a single point for caching Resource Index query results, decreasing latency as well as load on the shard servers. This configuration also allows a varying number of servers to be responsible for each shard segment, which provides fault tolerance, as well as flexibility in increasing the replication level of each shard based on traffic patterns. In this deployment pattern, the same Resource Index and Resource Store instances are again shared by multiple Replay UI deployments, so all three major Replay modes are supported using the same remote server resources.

3.4 Performance Analysis

Study of the performance characteristics of the Wayback code is ongoing, with some initial data available from experience so far and simulated volume testing. All statistics provided are based on software running on servers with a 1Ghz Via C3 processor, 512MB RAM, and 4 720RPM IDE hard drives.

Generating a CDX index for a single ARC file using Wayback software takes 60-120 seconds, but this rate varies dramatically depending on the consistency of each ARC file. Combining these per-ARC CDX indexes is then performed with a generic tool, such as the Gnu 'sort' command included with most Linux distributions.

We have not collected detailed timings on the creation of BDB indexes, but insertion rates of 7K records-per-second have been observed when building indexes of 10 million records or more. BDB insertion rates are known to be sensitive to the amount of BDB-specific cache memory allocated, and to the initial ordering of records. When inserting randomly ordered records, the insertion rate plummets after the total amount of data exceeds cache size; when inserting pre-sorted records, the insertion rate stays close to the initial rate.

Query and retrieval rates were tested against a sample Wayback installation containing an index of approximately 1.5 billion URL captures. This installation consists of a single front-end server hosting the ReplayUI, QueryUI, ArcProxy, and Alpha Distributed Resource Index components. The distributed Resource Index forwards requests to two index servers, each returning index data in XML format from queries against a pair of local CDX files, each on a different disk drive. The total size of the four CDX files being searched across the 2 index servers is 254GB. ARCs themselves are spread across hundreds of other storage servers, and the ArcProxy application forwards retrieval requests to the appropriate ARC storage server. The front-end server, two index servers, and storage servers all match the hardware configuration described above, but the storage servers are also regularly contributing to other active services, even during our performance testing.

For testing query performance, a set of 2,000 URLs and another set of 4,000 URLs were randomly chosen from the combined four CDX files. For each set of URls, the front-end Wayback services and index server Wayback processes were restarted, and the URLs queried in series via a scriptable HTTP client. The time for each query to return with results was records. For the 2,000 queries, the average elapsed time per query was 0.47 seconds; the median was 0.29 seconds. For the 4,000 queries, the average elapsed time was 0.48 seconds; the median was 0.28 seconds. Thus with one rapid-fire client, this Wayback configuration could serve 2-3 queries per second.

A similar procedure was used for testing retrieval performance. Sets of 2,000 URL-at-timestamp records and 4,000 URL-at-timestamp records were chosen at random from the combined CDX. For each set, the front-end and index server Wayback processes were restarted, and the URLs-at-timestamps were requested from the front-end server in series. For the 2,000 retrievals, the average elapsed time was 0.53 seconds; the median was 0.40 seconds. For the 4,000 retrievals, the average elapsed time was 0.64 seconds; the median was 0.40 seconds. Facing the one rapid-fire client, this Wayback configuration could serve 1.6-2.5 retrievals per second.

In a smaller informal test, up to 10 concurrent querying or retrieving clients in parallel had little effect on response elapsed time, but throughput began declining with greater traffic. Further investigation of current Wayback performance and potential optimizations is planned.

4. OPEN SOURCE PROJECT

The Internet Archive maintains the Wayback project on sourceforge.net, under the 'archive-access' project. The project website includes source code, documentation, a user manual, bug and feature tracking, project statistics. There is also an open mailing list for Wayback discussion by developers and software users, and mutual technical assistance. These can all be found at:

http://archive-access.sourceforge.net/projects/wayback/

The Wayback project is distributed under the GNU "Lesser" (or "Library") Public License (LGPL) [7], enabling it and its source code to be used standalone or as a component of larger projects under other licenses.

5. FUTURE DIRECTIONS

Plans for the Wayback in the future include:

- Improved Replay systems, which will involve novel methods for altering the context of replayed content, additional inserted in-page content providing simpler access to metadata about documents being viewed, and cooperating web browser plug-ins that will allow greater control of replayed content.
- Improved internationalization support, which will add additional languages to the Wayback UI elements.
- Improved functionality in access-control methods, which will allow content to be blocked or visible based on access location, capture time, user authentication, and various robots.txt policies.
- Deeper integration with the Heritrix web crawler, allowing additional content to be scheduled in the web crawler from the Wayback user interface, and linking from the Heritrix logs and user interface into the Wayback UI.
- Leveraging Wayback Proxy Replay mode for large-scale QA of collections. Using pools of automated web browsers, content can be loaded through the Wayback in the same method that end users will view the content, to identify web content that was missed in the initial capture process.
- Large-scale deployments, including replacing the software used to operate the 100 billion URL Internet Archive classic public Wayback Machine.
- Integration with the Hadoop data processing system to allow automated large-scale indexing operations.
- In-depth performance analysis comparing different index strategies (BDB, CDX, and full-text).

6. RELATED WORK

6.1 Classic public 'Wayback Machine'

In 2001, Alexa Internet and the Internet Archive collaborated in the development of an access tool to provide public access to the full historical collection maintained by the Internet Archive. The project was called the Wayback Machine, referring to a device used by professor Peabody in *The Rocky and Bullwinkle Show* to travel through time.

The initial deployment of the Wayback Machine held an index of approximately 100 TB of data, nearly 10 billion archived documents. The architecture of the service used 4 tiers:

1. Load balancer: distributing requests to the CGI farm
2. CGI Farm: Tens of Apache servers, running a mod_perl CGI, which roughly correlates to the Replay and Query UI.
3. Index Farm: Tens of custom C HTTP servers that performed binary searches through sorted text index files, and retrieved documents from the Document Server Farm, returning them to the CGI Farm nodes. This component roughly correlates to the Resource Index.
4. Document Server Farm: Hundreds of C custom TCP-protocol servers that extracted documents from ARC files, and returned them to the Index Farm nodes. This same functionality is implemented in the Wayback via HTTP 1.1 range requests, and the interface to this component roughly correlates to the Resource Store.

This architecture is still used to serve over 80 billion URL captures, but over time has shown problems. Some of this system uses Alexa proprietary code, so cannot be shared as the basis of an open source project. Additionally, this architecture has proved inflexible for experimentation with new features, and difficult to maintain, due to program logic that spans programs, machines, and programming languages.

One technology developed for the Wayback Machine that is significantly leveraged by the Wayback system is the method for rewriting HTML documents so hyperlinks refer back into the Wayback Replay UI. The technology uses minimal server-side document rewriting, which reduces the need for extensive and complex server libraries to handle malformed archived documents. Documents are modified as they are returned to clients by:

1. Adding or replacing a <BASE href=""> tag so relative links are resolved correctly against the original document URL.
2. Modifying some tags, including FRAME src's, and document background images, which cannot be modified by Javascript after the document has loaded within the client browser.
3. Inserting Javascript which executes within the client browser, after the page has loaded, and is responsible for changing all URLs in the document to point back at the Wayback Machine.

This context alteration technology is extremely simple, and heavily leverages client browser DOM and Javascript capabilities, but has several significant shortcomings, which are also present in the current Wayback implementations, since the same strategy is used:

1. Web browsers begin rendering the pages before the Wayback link-alteration Javascript has executed. Since web browsers are optimized to begin downloading embedded content while the original page is still loading, some requests for embedded content may first be made to the original content on the live web, before being subsequently patched to load the archived content in the Wayback Machine.
2. URLs present in embedded content, such as Java applets and Flash documents, are not rewritten, and often resolve to their original locations on the live web.
3. Javascript present in the original page is not updated by the Wayback software, so URLs constructed by this Javascript often resolve against the live web. In addition, Javascript in the original document occasionally will redirect the browser to a page on the live web, in effect hijacking the Wayback replay session.

The Proxy Replay mode present in the new Wayback software addresses most, if not all of these problems, but there is added complexity in requiring users to change their web browser's configuration to use the Wayback as a proxy server, and the Proxy mode is sometimes confusion to users who may not be aware of their browser's proxy configuration.

6.2 WERA: Access using full-text search and the Timeline

The Nordic Web Archive [8], a collaboration of Nordic National Libraries, in 2005 released WERA [6], another access tool for viewing archived web content in ARC files. WERA leveraged the same hyperlink patching technology found in the Wayback Machine, but also added two new powerful features. The first is a timeline banner user interface element, allowing users to navigate between archived versions of a page without returning to a search result list. The second major feature is the utilization of the NutchWAX full-text indexing system. This allows users to locate content within the archive using text search, whereas the original Wayback Machine only allowed access to content based on URL queries. WERA's features have directly inspired the Timeline Replay UI and NutchWAX Resource Index components/modes of the current Wayback project.

6.3 Filesystem Directory Tree Access

One relatively simple approach to solving the problems of archiving and providing access to archived content is to store web content directly on a file system. This approach is used by popular web-content collection tools including wget [9] and HTTrack [10].

If content is stored within the file system in a directory structure that mimics the original location of the content, then both the storage and (URL-based) indexing problems are solved. There are portability issues, since different operating systems use different directory separators, and may not allow filenames with special URL characters that can be used to describe content on the Internet. Additionally, most operating systems impose maximum filename lengths that limit the ability to correctly store some content.

In addition to the problems associated with accurately mapping the global URL space into file system paths, much of the original content needs to be altered when saving the content to correct hyperlink references. Hyperlinks expressed relative to a particular document may function correctly, but server-relative and absolute URLs will not correctly resolve to archived content unless they are modified when documents are saved.

One approach used by some of these web capturing tools, involves rewriting the content itself, so server-relative and absolute hyperlinks refer correctly to locally archived content. This approach has significant simplicity benefits in that once content is saved with this specialized software, it requires no specialized software to replay it. However, the process of altering the content cannot be reversed deterministically. Correcting problems discovered after documents are captured requires saving an original, unmodified version of the content, in addition to the version used for replay browsing. The same problems non-Proxy Wayback modes have with Flash and Java applets, and URLs generated by Javascript content, are also present in these filesystem-based tools today.

Current file systems are not usually adept at handling millions or billions of files, so this technique also has scaling limits that prohibit it from being used in many contexts. These considerations helped motivate the original design and adoption of the ARC file format by the Internet Archive and its storage and access software.

7. ACKNOWLEDGMENTS

The work of Dave Sherfesee, as primary creator of the original Wayback Machine, has been a seminal influence on all later extensions and new implementations of ARC-based web archive access tools. The International Internet Preservation Consortium and its individual member institutions have helped support the

Internet Archive's targeted web archiving and associated technology development, of which the open source Wayback project is an important part. The Library of Congress, especially, has helped drive the development of Wayback with their ongoing collection activities – and willingness to be early testers of new software. Thanks also go to the rest of the Internet Archive web team for helping develop, deploy, maintain, support, and explain Wayback for the world, and Gordon Mohr for assistance editing this paper.

8. REFERENCES

[1] Burner, Mike. and Brewster Kahle, "The ARC File Format," September 1996. <http://www.archive.org/web/researcher/ArcFileFormat.php>

[2] Koman, Richard. "How the Wayback Machine Works," January 21, 2002. <http://webservices.xml.com/pub/a/ws/2002/01/18/brewster.html>

[3] BerkeleyDB Java Edition, Oracle Corporation. <http://www.oracle.com/database/berkeley-db/je/index.html>

[4] "CDX File Format," <http://www.archive.org/web/researcher/cdx_file_format.php>

[5] NutchWAX project. <http://archive-access.sourceforge.net/projects/nutch/>

[6] WERA project. < http://archive-access.sourceforge.net/projects/wera/>

[7] GNU Lesser General Public License (LGPL). <http://www.gnu.org/licenses/lgpl.html>

[8] Nordic Web Archive home page. <http://nwa.nb.no/>

[9] Wget. <.http://wget.sunsite.dk/>

[10] HTTrack project. <http://www.httrack.com/>