InterPlanetary Wayback: Peer-To-Peer Permanence of Web Archives

Mat Kelly, Sawood Alam, Michael L. Nelson, and Michele C. Weigle

Old Dominion University, Department of Computer Science
Norfolk VA, 23529, USA
{mkelly,salam,mln,mweigle}@cs.odu.edu

Abstract. We have integrated Web ARChive (WARC) files with the peer-to-peer content addressable InterPlanetary File System (IPFS) to allow the payload content of web archives to be easily propagated. We also provide an archival replay system extended from pywb to fetch the WARC content from IPFS and re-assemble the originally archived HTTP responses for replay. From a 1.0 GB sample Archive-It collection of WARCs containing 21,994 mementos, we show that extracting and indexing the HTTP response content of WARCs containing IPFS lookup hashes takes 66.6 minutes inclusive of dissemination into IPFS.

Keywords: Web Archives, Memento, Peer-to-Peer, IPFS

1 Motivation

The recently created InterPlanetary File System (IPFS) [2] facilitates data persistence through peer-to-peer content-based address assignment and access. While web archives like Internet Archive (IA) provide a system and means of preserving the live web, the persistence of the archived web data over time is dependent on the resilience of the organization and the availability of the data [5]. In this paper we introduce a scheme and software prototype[1], InterPlanetary Wayback (ipwb), that partitions, indexes, and deploys the payloads of archival data records into the IPFS peer-to-peer "permanent web" for sharing and offsite massively redundant preservation and replay.

2 Background and Related Work

The Web ARChive (WARC) format is an ISO standard [4] to store live web archive content in a concatenated record-based file. IA's web crawler, Heritrix [7], generates WARC files to be read and the content re-experienced in an archival

[1] https://github.com/oduwsdl/ipwb

replay system. OpenWayback[2] (written in Java) and pywb[3] (written in Python) are two such replay systems. We leverage and extend on pywb in this work.

To access the representations stored by an archival crawler, a replay system must refer to an index that maps the records in a WARC file to the original URI (or, URI-R in Memento [9] terminology). The CDX format is one such index along with the extended CDXJ format [1], which allows for arbitrary JSON objects within each index record. InterPlanetary Wayback takes advantage of pywb's native support for CDXJ and uses the arbitrary JSON data to store metadata about WARC records within IPFS (i.e., the content digest needed for lookup in IPFS).

IPFS is a peer-to-peer distributed file system that uses a file's content as the address for lookup [2]. By extracting the HTTP message body [3] (henceforth "payload") from the records within a WARC file, IPFS allows our prototype to generate a signature uniquely representative of this content. This payload can then be pushed into the IPFS system, shared to the network, and then retrieved at a later date when the URI-M is queried. This unique signature of WARC content provides de-duplication of identical content in web archives where other efforts to prevent redundant information within a web archive at crawl time has mostly been addressed [8]. While these efforts focus on intra-archive de-deduplication, ipwb additionally provides inter-archive de-deduplication of archival content.

3 Methodology

We utilize the JSON object of a CDXJ record to store digests of the HTTP response payloads, HTTP response headers, original status code when the URI-R was crawled, and the MIME-type of the payload content. A CDXJ record also contains a Sort-friendly URL Reordering Transform (SURT) URI-R and a datetime in the fields before the JSON. The digests are encoded into a field called locator as a Uniform Resource Name (URN) [6] (Figure 1).

In designing ipwb, it was critical to consider the HTTP headers returned at crawl time separately from the payload. The HTTP response headers will change with every capture, as the datetime returned from a server includes the current time. Compare this to the payload, which often contains the same content on each access. Were the HTTP headers and payload combined then added to IPFS, every IPFS hash would be unique, nullifying the potential for de-duplication of identical content. Furthermore, ipwb only retains HTTP response records. The rationale for this design decision is that the state of the art of web archive replay systems do not consider the WARC request record upon replay. While including request records may be useful in the future (for instance, to take into account the

[2] `https://github.com/iipc/openwayback`
[3] `https://github.com/ikreymer/pywb`

```
SURT_URI DATETIME {
        "id": "WARC-Record-ID",
        "url": "ORIGINAL_URI",
        "status": "3-DIGIT_HTTP_STATUS",
        "mime": "Content-Type",
        "locator": "urn:ipfs/HEADER_DIGEST/PAYLOAD_DIGEST"
}
```

Fig. 1: A CDXJ index allows a memento to be resolved to a WARC record in a playback system. In the ipwb prototype we extract the relevant values from the HTTP response headers at time of index and include the IPFS hashes as the means for a replay system to obtain the HTTP headers and payload corresponding to the URI-M requested.

user agent originally used to view the live website), WARC content is currently able to be replayed without preserving the request records.

4 Implementation

Our prototype implements an indexer (shown as the red circles in Figure 2) that extracts HTTP headers and payload from a WARC record and stores them in the IPFS storage as separate objects. The two object references returned from the IPFS along with a URI-R, datetime, HTTP status code, content-type, etc. are then used to construct an index record, which is stored in CDXJ format (Figure 1), per Section 3.

InterPlanetary Wayback also implements an archival replay system (shown as the blue circles in Figure 2) using components from pywb. When a client requests a TimeMap, all corresponding records are fetched from the index and a TimeMap is constructed. When a client requests a memento, a single corresponding record is fetched from the index. This record will contain IPFS reference hashes for the HTTP headers and payloads corresponding to the URI requested. Using these two hashes, HTTP headers and payloads are fetched from the IPFS store and a response is constructed.

5 Evaluation

We tested our ipwb prototype on a data set from an Archive-It collection[4] about the 2011 Japan Earthquake consisting of 10 WARC files, each about 100 MB when compressed, totaling 1.0 GB on disk.

[4] `https://archive-it.org/collections/2438`

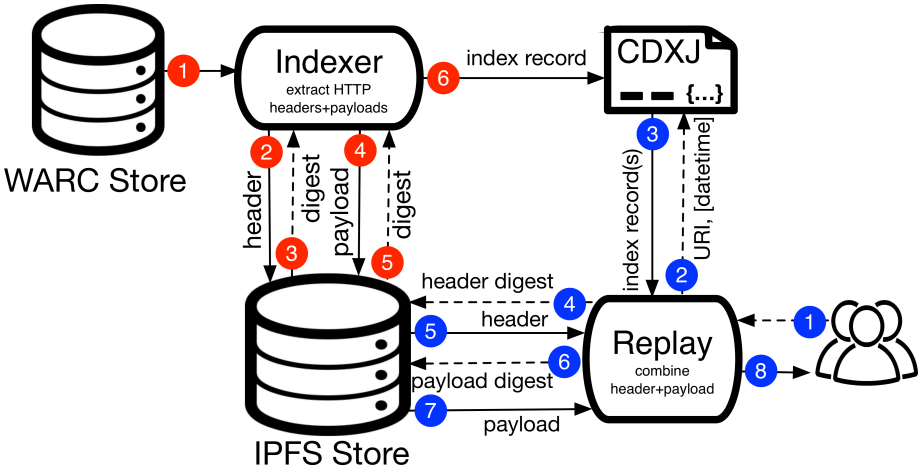


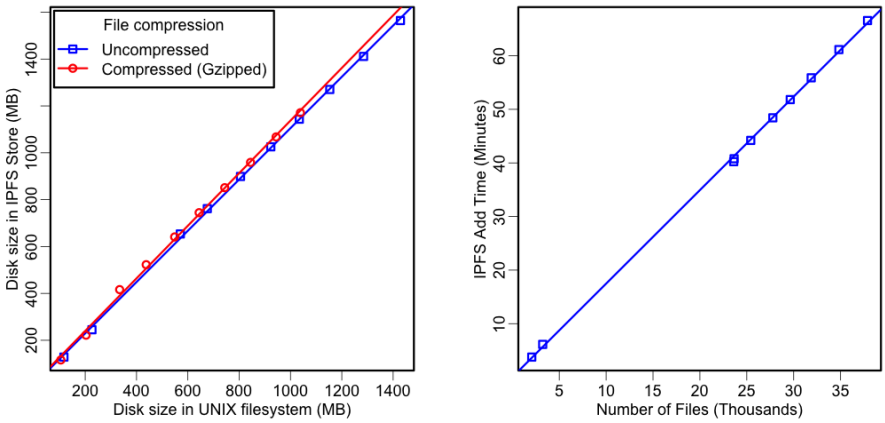
Fig. 2: Pushing WARC records to IPFS (red circles) requires the WARC response headers and payloads to be extracted (red 1), pushed to IPFS to obtain digest hashes (red 2-5), and hashes to be included in an index (red 6). The replay process (blue circles) has a user querying a replay system as usual (blue 1) that obtains a digest for the URI-datetime key from the index (blue 2 and 3), which is used as the basis for retrieving the content associated with the digests from IPFS (blue 4-7). The replay system can then process these payloads as if they were in local WARC files and return the content to the user (blue 8).

We indexed the WARCs using pywb's cdx-indexer and ipwb's prototype indexer to generate a standard CDXJ file and one containing the IPFS-hashes, respectively, as described in Section 3. The experiment was run on a late 2013 MacBook Pro running OS X version 10.11.3 with a 2.4 GHz Intel i5 processor, 8 GB of RAM, and a 250 GB SSD disk. Generating ipwb's CDXJ file for 21,981 mementos in the data set took 66.6 minutes including the time required to push the data into the IPFS network, producing the IPFS hashes to be included in the CDXJ. The average indexing rate inclusive of the data dissemination to IPFS was 9.48 files per second. Because IPFS is in the early stages of development, performance when adding files to the IPFS network[5] contributes a large part of the latency in the replay procedure.

To evaluate the replay time, we fetched 600 sample URI-Ms from each of pywb and ipwb independently, both using the same WARC basis for CDXJ generation, performed prior to the replay procedure. The total time required for pywb to access the sample URI-Rs using local WARC files for lookup was 5.26 seconds. The same URI-Rs replayed in ipwb with the same WARC records disseminated into the IPFS system took 222 seconds. The increased latency is

[5] `https://github.com/ipfs/go-ipfs/issues/1216`

because of how IPFS works, however, it provides caching that never expires, i.e., a change in content will cause a change in address. The latency is also because of our naive implementation where we fetch the header and payload sequentially rather than in parallel from the IPFS. Additionally, IPFS promises greater persistence (which is desired in archiving) with the cost of added latency. Figure 3a shows the amount of disk space required to convert and add compressed and uncompressed WARC content to IPFS. In the tested data set, with little duplication of HTTP response bodies because of URI-M uniqueness, the slope of the uncompressed additions was 1.10 while the slope of the compressed additions was 1.12. In practice, where duplication of payload content is much more prevalent in a collection of WARCs, the file representative of the payload of the duplicated content will not need to be added to the IPFS, requiring only the file representative of the unique HTTP header (Section 3) to be added. This would result in a significantly smaller slope were the experiment extended to a larger collection. Figure 3b shows that as more files (extracted from the WARCs) are added to the IPFS system, the time required to do so correlates linearly with the number of files (not necessarily the size of the files for small files) with a slope of 1.74 (on average, 570 files per minute).



(a) Disk space required using both compressed and uncompressed WARC headers and payloads as compared to the file system and the IPFS Store.

(b) As the number of files added to the IPFS network increases, the time required to do so linearly increases (on average, 570 files per minute).

Fig. 3: IPFS Storage Space and Time Cost Analysis

6 Future Work and Conclusions

Because of the novelty of IPFS, particularly relative to web archiving, there are numerous applications to expand this work. Collection builders can share their collections by just exchanging the index while keeping the data in the IPFS network and others can optionally replicate the data in their storage for redundancy. Further considerations of access control can also be addressed to encrypt and restrict content based on privacy and security mechanisms. Another model of IPFS-based archiving system can be built entirely using IPFS and IPNS technologies without the need of external indexes.

In this work we developed a prototype to partition, disseminate, and replay WARC file records in the InterPlanetary File System (IPFS). Through experimentation on a 1.0 GB data set containing 21,994 URI-Ms, we found that extracting and indexing records from WARC files took 66.6 minutes inclusive of dissemination into the IPFS system. The average indexing rate inclusive of the data dissemination to IPFS was 570 files per minute on average.

Acknowledgements. We would like to thank Ilya Kreymer for his feedback during the development of the ipwb prototype and guidance in interfacing with the pywb replay system. This work was supported in part by NSF award 1624067 via the Archives Unleashed Hackathon[6], where we developed the prototype.

References

1. S. Alam. CDXJ: An Object Resource Stream Serialization Format. `http://ws-dl.blogspot.com/2015/09/2015-09-10-cdxj-object-resource-stream.html`, September 2015.
2. J. Benet. IPFS - Content Addressed, Version, P2P File System. Technical Report arXiv:1407.3561, July 2014.
3. R. Fielding and R. J. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. IETF RFC 7230, June 2014.
4. ISO 28500. WARC (Web ARChive) file format. `http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml`, August 2009.
5. P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker. The LOCKSS Peer-to-Peer Digital Preservation System. *ACM Transactions on Computer Systems*, 23(1):2–50, February 2005.
6. R. Moats. URN Syntax. IETF RFC 2141, May 1997.
7. G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to Heritrix, an Archival Quality Web Crawler. In *Proceedings of the 4th International Web Archiving Workshop (IWAW'04)*, September 2004.
8. K. Sigurðsson. Managing Duplicates Across Sequential Crawls. In *Proceedings of the 6th International Web Archiving Workshop (IWAW'06)*, September 2006.
9. H. Van de Sompel, M. Nelson, and R. Sanderson. HTTP Framework for Time-Based Access to Resource States – Memento. IETF RFC 7089, December 2013.

[6] `http://archivesunleashed.ca`