InterPlanetary Wayback: The Permanent Web Archive

Sawood Alam, Mat Kelly, and Michael L. Nelson
Old Dominion University, Department of Computer Science, Norfolk VA, 23529, USA
{salam,mkelly,mln}@cs.odu.edu

ABSTRACT

To facilitate permanence and collaboration in web archives, we built InterPlanetary Wayback to disseminate the contents of WARC files into the IPFS network. IPFS is a peer-to-peer content-addressable file system that inherently allows deduplication and facilitates opt-in replication. We split the header and payload of WARC response records before disseminating into IPFS to leverage the deduplication, build a CDXJ index, and combine them at the time of replay. From a 1.0 GB sample Archive-It collection of WARCs containing 21,994 mementos, we found that on an average, 570 files can be indexed and disseminated into IPFS per minute. We also found that in our naive prototype implementation, replay took on an average 370 milliseconds per request.

1. INTRODUCTION

The recently created InterPlanetary File System (IPFS) [2] is showing the potential to facilitate data persistence through a peer-to-peer network for dissemination and discovery. 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 redundant preservation and replay.

The Web ARChive (WARC) format is an ISO standard[2] to store live web archive content in a concatenated record-based file. IA's web crawler, Heritrix [3], generates WARC files to be read and the content re-experienced in an archival replay system. OpenWayback[3] (written in Java) and pywb[4] (written in Python) are two such replay systems. We leverage and extend on the pywb codebase in this work.

To access the representations stored by an archival crawler, a replay system must refer to an index that maps the original URI (or, URI-R in Memento [4] terminology) and the time of capture (Memento-Datetime) to the record stored in a WARC file. CDX is one such indexing format along with the extended CDXJ format [1], with the latter allowing arbitrary JSON objects within each index record. In our initial prototype we take advantage of pywb's native support for CDXJ and use 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 content addressable peer-to-peer distributed file system [2]. By extracting the HTTP response body (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 and retrieved at a later date when the URI-M is queried. Content addressability allows for network-wide deduplication of the content. The digest of the content is used as the key to locate the content in the peer-to-peer network.

```
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"
}
```

Figure 1: A single-line CDXJ record template, shown on multiple lines for readability

2. IMPLEMENTATION

CDXJ is a text-based file format that we utilize to store indexes of the archived content. Each line in the CDXJ file holds one index record. The line begins with a SURTed URI[5] and datetime followed by a single-line JSON block that stores reference to the content and other arbitrary metadata (Figure 1). We utilize the last field in a CDXJ record (a JSON object) to store the HTTP response headers and payload digests, original status code when the URI-R was crawled, the MIME-type of the content, and a UUID to identify a memento. The two digests that are used to locate the contents from the IPFS system and build the response are encoded into a single field called "locator" using a URN scheme[6].

In designing ipwb, it was critical to consider the HTTP header returned at crawl time separately from the HTTP response body. The HTTP response header's content will change with every capture, as the datetime returned from a server is temporally dependent. Compare this to the response body, which very often contains the same content on each access, more often for static resources. Were the HTTP header and response body combined then added to IPFS, every IPFS hash would be unique, nullifying the potential for de-duplication of identical content. Further, ipwb only retains 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 user-agent originally used to view the live website), WARC content is currently fully replayable without preserving the request records.

[1]https://github.com/oduwsdl/ipwb
[2]http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml
[3]https://github.com/iipc/openwayback
[4]https://github.com/ikreymer/pywb

[5]http://crawler.archive.org/articles/user_manual/glossary.html#surt
[6]https://www.w3.org/TR/uri-clarification/

JCDL '16 June 19-23, 2016, Newark, NJ, USA

ACM ISBN 978-1-4503-4229-2/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2910896.2925467

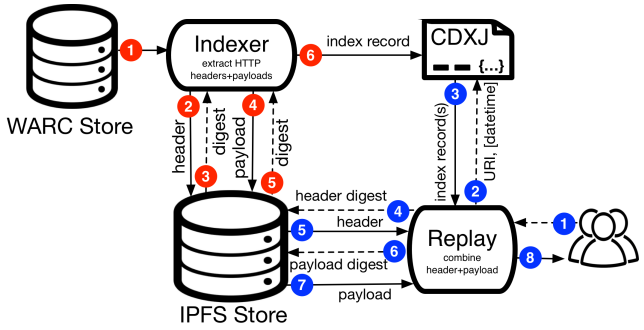
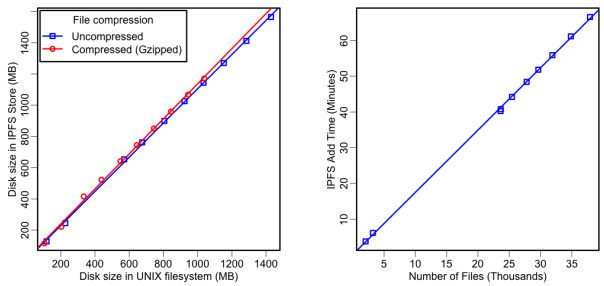


Figure 2: The ipwb indexer and replay workflow



(a) Space cost (b) Time cost

Figure 3: IPFS storage space and time cost analysis

Our prototype works in two phases, illustrated in Figure 2 with red (*Indexing*) and blue (*Replay*) annotations:

- *Indexing* – extracts records from the WARC store one record at a time, splits each record into HTTP header and payload, stores the two pieces into IPFS (compressing before storing, if necessary), and generates a CDXJ record using the returned references and some other metadata from the WARC record.
- *Replay* – receives request from users containing a lookup URI and optionally a datetime, queries for matching record in the CDXJ, fetches the corresponding header and payload from the IPFS Store (using references returned from the index record), combines them, and performs necessary transformation to build the response to the user.

3. EVALUATION

We tested our ipwb prototype on a data set from an Archive-It collection[7] about the 2011 Japan Earthquake consisting of 10 WARC files, each about 100 MB when compressed, totaling 1.0 GB on disk.

We indexed the WARCs using pywb's cdx-indexer and ipwb's indexer to generate a standard CDXJ file and one containing the IPFS-hashes (as described in Section 2), respectively. 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 and 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 many small files to the IPFS network[8] consumes a large part of the time required for indexing.

To evaluate the replay time, we fetched 600 sample URI-Ms from each of pywb and ipwb independently, with 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 because of how IPFS works, however, it provides infinite cacheability and can benefit from CDNs. 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. With the tested data set where there is very 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 response body content is much more prevalent in a collection of WARCs, the file representative of the response body of the duplicated content will not need to be added to the IPFS, requiring only the file representative of the unique HTTP header (Section 2) 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).

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

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

4. 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.

5. 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[9], where we developed the prototype.

6. REFERENCES

[1] S. Alam, M. L. Nelson, H. Van de Sompel, L. Balakireva, H. Shankar, and D. S. H. Rosenthal. Web Archive Profiling Through CDX Summarization. In *Proceedings of TPDL '15*, pages 3–14.

[2] J. Benet. IPFS - Content Addressed, Version, P2P File System. Technical Report arXiv:1407.3561, 2014.

[3] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to Heritrix, an Archival Quality Web Crawler. In *Proceedings of IWAW '04*, September 2004.

[4] H. Van de Sompel, M. Nelson, and R. Sanderson. HTTP Framework for Time-Based Access to Resource States – Memento. IETF RFC 7089, December 2013.

[9]http://archivesunleashed.ca