Deduplicating Bibliotheca Alexandrina's Web Archive

Youssef Eldakar
Bibliotheca Alexandrina
PO Box 138
Alexandria 21526
EGYPT
youssef.eldakar@bibalex.org

Magdy Nagi
Bibliotheca Alexandrina
PO Box 138
Alexandria 21526
EGYPT
magdy.nagi@bibalex.org

ABSTRACT

Archiving web content is bound to produce datasets with duplication, either across time or across location. The Bibliotheca Alexandrina (BA) has a web archive legacy spanning a period of 10 years and is continuing to expand the collection. Initial assessment of this very large store of data was conducted. Given a high enough rate of duplication, deduplication would lead to sizable savings in storage requirements. The BA worked through the International Internet Preservation Consortium (IIPC) to compile best practices for recording duplicates in ISO 28500, the WARC File Format. To deduplicate legacy web archives "after the fact," the BA is implementing the WARCrefs deduplication tools. Following implementation and testing, the BA plans to put the tools to use to deduplicate its one petabyte of archived web content.

General Terms

case studies and best practice

Keywords

web archiving, deduplication, hash algorithms, ISO 28500, WARC File Format, WARCrefs, WARCsum

1. INTRODUCTION

The International Internet Preservation Consortium (IIPC) defines *web archiving* as "the process of collecting portions of the World Wide Web, preserving the collections in an archival format, and then serving the archives for access and use" [9]. During the collection phase of the process, a *crawler* is used to explore a network of hyperlinks, starting off at a set of seeds, fetching resources it visits. This process is typically repeated periodically to capture changes over time.

However, even though the web is quite a dynamic place, a resource will not necessarily be modified during the interval between one visit and a subsequent revisit. In addition, even though the web is quite a diverse place, a resource will not necessarily be unique within the web archive when compared to other resources at different locations. For the former situation, consider, for instance, the text of the constitution of some country on the government's website, which can be expected to remain unmodified for years, where archiving subsequent identical snapshots of the resource as-is introduces duplication across time into the archive. For the latter situation, consider, for instance, a photo of some event that is posted to a blog, a social networking site, as well as a personal homepage, where archiving all identical instances of the resource as-is introduces duplication across location into the archive.

The Bibliotheca Alexandrina (BA) in Alexandria, Egypt, has in its holdings a legacy web archive of broad web crawls provided by the Internet Archive in San Francisco. This archive of a decade's web history starting in 1996 plus BA's own collection of focused web crawls started in 2011 total approximately 80 billion records and are stored in approximately one petabyte in compressed form. For backup, an additional petabyte is required. And even though the data is hosted on a commodity hardware computer cluster, at such large scale, reduction in storage requirements even by a relatively modest percentage would lead to sizable financial savings and offer more room for expanding collection activities.

Beyond using storage more efficiently, the desire to deduplicate web archive data is driven by a few extra motivations. Kristinn Sigurdhsson, in a 2006 publication [16], takes a look at arguments for deduplication in a web archive. It is noted therein that reduction in storage requirements is the most notable benefit in addition to improving the quality of the collection or its presentation. Other benefits also mentioned therein but applying only to deduplication during crawl time based on HTTP headers are reduction of load on web servers as well as reduction in bandwidth consumption. We may also add to the benefits of deduplication improvement in performance on the access interface, as knowing which resources are duplicates of which resources would improve the caching implementation.

Could the rate of duplication in the BA web archive be significant enough that the benefits to be achieved merit the effort?

2. IDENTIFYING DUPLICATES

Before evaluating how much more efficient use of the storage infrastructure would become should the BA web archive be free of duplicates, a method for telling whether two resources are identical is needed. The most rudimentary one is to compare the data streams byte-by-byte. Given a set of n data streams, $n(n-1)/2$ comparisons will be required, because each data stream will have to be compared with all data streams in the set but itself, forming a complete graph with n vertices. To optimize, comparisons where the data streams are not equal in size and therefore cannot be identical will be skipped. To further optimize, comparisons will also be skipped where the data streams have already been found identical via an indirect route on the graph. Yet, where n is very large, and where the data is hosted on a distributed storage infrastructure, this method will not scale well, because data will have to be marshalled heavily on the network during the repetitive reads and compares.

An alternative method is *hashing*. Hash functions are algorithms that map a data stream of arbitrary length to a fixed-length *hash value*, which uniquely identifies the data stream [3]. Using this method, each data stream will be read once and hashed. Each hash value along with a reference to the data stream will be inserted into a list. The list is sorted on the hash value, clustering entries for identical data streams together. Costly repetitive reads and compares in the former method are replaced with a much lighter merge-sort.

Password verification, data integrity checking, and even automatic *deduplication* built into modern file systems such as ZFS [7] and Btrfs [1] are examples of today's common applications of hash algorithms. Of hash algorithms, MD5, SHA-1, and SHA-2 seem to be de-facto standards in the industry.

A hash algorithm is reliable up until it is shown to entail a risk for *collisions*, where two unidentical data streams are mapped to the same hash value. As presented in CWI's "Cryptanalysis of MD5 and SHA-1" [17], the possibility for collisions is demonstrated for MD5 in theory as well as in practice, and only in theory for SHA-1. To date, there seems to be no published work demonstrating susceptibility of SHA-2 to collisions. However, the more reliable SHA-1 and SHA-2 algorithms come at a cost. In Crypto++ 5.6.0 benchmarks [2], MD5 performed 65 percent faster than SHA-1, and well over twice as fast as 256-bit SHA-2 (SHA-256) and 512-bit SHA-2 (SHA-512). See Figure 1.

Falsely identifying resources in the web archive as duplicates shall not be tolerated, as that would lead to corruption in web history. Here, one model to take as reference is how deduplication is managed in the ZFS file system, where false positives also are not tolerated. ZFS uses SHA-256 to identify duplicates but also runs positives through collision resolution, which compares the files byte-by-byte to rule out collisions. Such a model seems quite suitable for application in web archive deduplication. In fact, in the safety of collision resolution, even MD5 could be considered for its significant speed advantage, and in hopes collisions will not be very frequent. Further, statistics on collisions in such very large dataset as a web archive could be of value to the cryptography community.

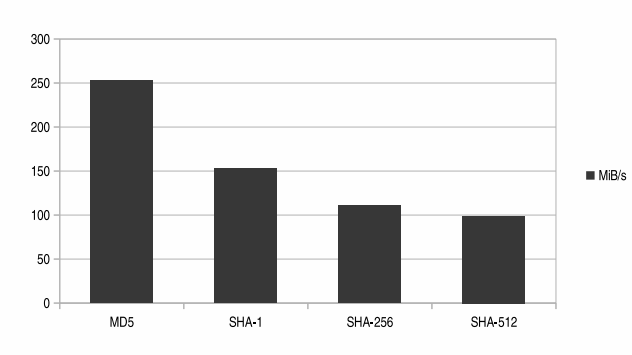

Figure 1: Crypto++ hash algorithm benchmarks.

3. INITIAL ASSESSMENT

In 2012, the BA sampled close to 10 percent of data in its web archive. Hash values for the resources were computed and sorted into a list. Out of this sample, the rate of duplication was found to be approximately 14 percent. With a bigger sample, it may be reasonable to optimistically hope the rate of duplication will be higher. Still, even just a saving of 14 percent within the petabyte of data in the BA web archive would translate into 140 terabytes of space. Multiplying this number by two to account for the backup, the total storage saved becomes 280 terabytes. Such saving on storage would yield a considerable cut-down on expenses, which may possibly be invested towards widening the scope of web archiving at the institution.

In addition to the duplication assessment on the BA web archive, other web archives also report significant rates of duplication. For instance, in a presentation during the 2011 IIPC General Assembly [15], the National Diet Library, Japan, estimated deduplication would reduce the Japanese monthly web archives by 80 percent, and the quarterly archives by 45 percent. Further experiences shared during discussions at IIPC meetings did also describe the rate of duplication as being significant in web archive collections at other institutions.

4. STANDARDIZING THE RECORDING OF DUPLICATES

The de-facto standard format used to store resources in web archives is the ARC File Format initially conceived at Alexa Internet [11], to which ISO 28500, the WARC File Format, is a more comprehensive successor [14].

In Sigurdhsson's 2006 publication [16], it is noted in the conclusion, "While there are difficulties in presenting collections that have been [deduplicated], the introduction of the WARC File Format should greatly alleviate that" [16]. However, even though the ISO specification provides for `revisit` records, the specifics for practical usage are not clearly outlined. In 2013, the BA worked through the IIPC Harvesting Working Group (HWG) to draft a "Proposal for Standardizing the Recording of Arbitrary Duplicates in WARC Files" [10]. Later that same year, the proposal earned IIPC Steering Committee adoption as recommended best practices.

The proposal recommends the use of the following fields in

revisit records with the identical-payload-digest profile in the WARC File Format to replace duplicate resources with references to the initial capture:

WARC-Refers-To-Target-URI: The URI of the original resource.

WARC-Refers-To-Date: The date of the original resource.

In addition, the proposal discourages the use of "fields specifying the actual WARC file name and offset," as such usage is "potentially very brittle."

5. TOOLS

Deduplication of web archive data may be carried out at either of two phases: during a crawl, or after the crawl.

5.1 During a Crawl

In Sigurdhsson's 2006 publication [16], modules implemented for the Heritrix crawler [4] to stop processing of duplicate resources are presented. The DeDuplicator module depends on hash values to identify duplicates using an index of previously crawled resources as reference. In addition, the implementation also provides an alternative method for identifying duplicates based on the datetime and/or ETag in HTTP headers fetched during the crawl: the DeDupFetchHTTP module.

5.2 Post-Crawl

Even though the DeDuplicator and DeDupFetchHTTP modules are quite effective in eliminating duplicates during a crawl, the BA requires a different type of solution that enables the archive keeper to sort out duplicates "after the fact" in a legacy collection. For this, the BA has implemented the WARCrefs set of tools for identifying duplicates and converting them to references in a web archive collection after crawl time.

BA's developed solution is divided into two separate packages: WARCrefs for doing the deduplication, and WARCsum for generating hash manifests of web archive resources, doing collision resolution, and post-processing the manifests, which serve as input to WARCrefs. WARCrefs is implemented in Java, because well-maintained WARC manipulation APIs are available in this language. WARCsum, on the other hand, is implemented in C, because hash generation as well as collision resolution are time-consuming tasks, particularly when dealing with big data, which makes C as a lower-level language a good choice when seeking to tune performance. The software is operated at the command line.

Stage 1 in the deduplication process is warcsum. This tool takes as input a list of WARC files. For each record of type response in each file in the input, one line of output is written to the hash manifest. Each hash manifest line consists of six fields: the WARC file name, the file offset in bytes at which the record is located, the length of the record also in bytes, the URI the record captures, the date of the capture, and the hash value of the content in the payload, excluding headers. The following is an example of what warcsum writes for a record in the input (for readability, each field is on a separate line):

```
TGvr4fAfmc.warc.gz
3901
635
http://www.akhbarway.com/robots.txt
2012-04-08T20:13:38Z
sha1:aa20238aab9cea0696a9b5d5f7a44a42de16adfc
```

warcsum can be configured using command-line options. warcsum uses hash functions from OpenSSL [5]. MD5, SHA-1, SHA-256, and SHA-512 are supported. For records where a hash value is already present in the record headers, warcsum can be set to use the existing value or recompute the hash. Records with empty content can be skipped or treated as normal records.

warcsum is to be run on each host in the computer cluster that makes up the data store where WARC files are kept. Output from all instances is to be aggregated and sorted on the hash value field.

Stage 2 is warccollres, provided in the WARCsum package. This tool performs the collision resolution, acting as a safety measure against false positives due to hash collisions. warccollres takes as input the aggregated hash manifest generated by warcsum. For each cluster of lines having the same hash value, the content is fetched from the data store and compared byte-for-byte to verify whether the records are indeed duplicates. The result of the collision resolution is saved to the hash manifest line by appending a seventh field: a *hash extension*, which is a sequential number that distinguishes unidentical records incorrectly given the same hash value. The output from warccollres is an extended hash manifest.

To access the data store, warccollres looks up file names in a MySQL database to determine where they are located then fetches the records via HTTP. The de-facto standard web archive access system, the OpenWayback [6], already depends on HTTP for fetching records, hence going with HTTP as the first choice for the fetch method and reusing an existing infrastructure was natural. In the future, alternative fetch methods may be implemented into warccollres.

warccollres is to be run on one or more hosts with HTTP access to the data store. When run on multiple hosts to distribute the collision resolution workload, the input hash manifest is to be partitioned across each host such that each cluster of lines having matching hash values is fully contained within a single partition. warcsumsplit is a tool provided in the WARCsum package for this purpose. Output from all warccollres instances is to be aggregated and sorted on the hash value and hash extension fields.

Stage 3 is warcsumproc, also provided in the WARCsum package. This tool post-processes the extended hash manifest, further extending it such that each line encapsulates all the information the deduplication stage needs to operate on the record the line is for. Post-processing looks at the hash value and hash extension in each line in the context of the line before it and writes a *copy number* as the eighth field on the line. Thus, a line where the copy number is 1 is an original record to be kept intact, while a line where the copy number is greater than 1 is a duplicate to be con-

verted into a reference, i.e., a `revisit` record. In addition, where the copy number is greater than 1, the post-processed hash manifest also has as the ninth and tenth fields the URI and date, respectively, of the original record, which is information needed to construct the `revisit` record. The post-processed hash manifest is to be sorted on the file name and offset fields.

The post-processing functionality is also available in `warccollres` and can be enabled using a command-line option, in which case, `warccollres` outputs a post-processed hash manifest. This is more efficient than performing the post-processing in a separate stage. However, given enough confidence that the hash algorithm being employed is not likely to have collisions, with some risk, the collision resolution may be skipped in order to save time, in which case, `warcsumproc` is needed. Needless to say, opting not to perform the collision resolution is quite inadvisable.

Both `warcsum` and `warccollres` read WARC files using `libgzmulti`, a library the BA developed as a wrapper around `zlib` [13] for working with multi-member GZIP files, of which WARC files are a type.

Stage 4 is where the post-processed hash manifest produced by the toolchain provided in the WARCsum package is turned over to WARCrefs to perform the deduplication. Similar to `warcsum` (stage 1), the `warcrefs` tool takes as input a list of WARC files but also now has access to post-processed hash manifest lines for records in the files it is to operate on. `warcrefs` iterates through each WARC file in the input and also concurrently through corresponding lines in the post-processed hash manifest. Each record with a copy number greater than 1 in the corresponding manifest line is converted into a `revisit` record, where `WARC-Refers-To-Target-URI` and `WARC-Refers-To-Date` in the record headers are set to the URI and date, respectively, of the original resource, and payload headers are transferred as-is into the `revisit` record. Otherwise, if the copy number is 1, or if no corresponding line is in the manifest, the record is not altered.

`warcrefs` uses the Java Web Archive Toolkit (JWAT) [12] for WARC file IO. `warcrefs` can be configured to rewrite files in-place or save to a new file.

`warcrefs` is to be run on all hosts in the data store. The post-processed hash manifest is to be split across the hosts such that each host only has lines corresponding to records in WARC files on the host. Further, as the absence of a manifest line for a record implies the record is not a duplicate, lines where the copy number is 1 are to be omitted to reduce the amount of manifest data `warcrefs` has to process.

Figure 2 illustrates the deduplication process.

WARCrefs, WARCsum, and `libgzmulti` will be available open-source on GitHub [8].

6. EXECUTION

With the solution implemented, the next objective is to put the WARCrefs deduplication tools to use to deduplicate the full BA web archive. The plan is as follows:

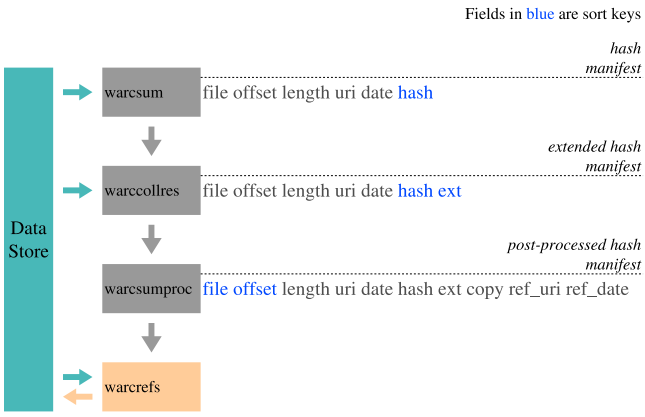

Figure 2: WARCrefs deduplication process.

1. **Test the tools.** In deduplication, records are identified as duplicates and deleted, substituting in references to what is allegedly the original record. In the event that either the identification or the deduplication makes a bad move, this can result in data loss, where all copies of a resource are converted to `revisit` records, or where `revisit` records are set to point to something that is not a copy of the converted record. Moreover, as deduplication is data rewriting, data corruption is also a concern. If any of these errors occur, the damage will be irreversible once the rewriting is committed to both copies of the data. Therefore, extensive testing scenarios must be thought out and carried out before putting the tools to production use.

2. **Generate hash manifest.** The legacy web archive collection that was provided to the BA by the Internet Archive in San Francisco is in the old ARC File Format. Therefore, at this point, `warcsum` is to process ARC as well as WARC files.

3. **Convert ARC to WARC.** As the described deduplication process is designed to work with the new WARC File Format, conversion has to take place. The JWAT toolkit [12] will be used to convert one of the two copies of the data, leaving the other copy untouched as a fallback measure.

4. **Validate the conversion.** Generate a new hash manifest for the all-WARC copy of the data. Compare this manifest to the one generated pre-conversion. Investigate and act on discrepancies as necessary.

5. **Deduplicate.** Carry on with the deduplication process on the all-WARC copy of the data starting at stage 2 (`warccollres`). Be sure to use the post-conversion manifest, as record offsets and lengths in the WARC version of a file are different from those in the ARC version.

6. **Validate the deduplication.** Generate yet another hash manifest for the deduplicated data and compare to the post-conversion manifest to confirm non-duplicate records were not altered. Also, select a random sample of deduplicated records for testing through the access system. Investigate and act on issues as necessary.

7. **Commit to second copy.** If confident the deduplication process resulted in no damage to the data, commit the deduplicated set over to the second set that was kept unaltered throughout the process.

7. CONCLUSION

Deduplication is an effective technique for making smarter use of the storage infrastructure that supports a web archive, and also comes with a few desirable side effects, such as improving the quality of the collection. Proper identification of duplicates based on hash values and applying a second check to rule out collisions ensure the deduplication target is selected accurately. Best practices for standardizing how duplicates are represented in the WARC File Format have been drafted within the International Internet Preservation Consortium. The Bibliotheca Alexandrina is developing the tools needed to execute post-crawl deduplication of its web archive, and hopes to report on results and lessons learned from this petabyte-scale data rewriting job in a future venue. Other institutions involved in web archiving are welcome to put the tools to test on their own collections as well as contribute to improving the software.

8. ACKNOWLEDGMENTS

The Bibliotheca Alexandrina wishes to thank colleagues in the IIPC Harvesting Working Group, most notably, the National and University Library of Iceland, for work drafting the "Proposal for Standardizing the Recording of Arbitrary Duplicates in WARC Files."

9. REFERENCES

[1] Btrfs Wiki. `https://btrfs.wiki.kernel.org/index.php/Main_Page`.

[2] Crypto++ 5.6.0 Benchmarks. `http://www.cryptopp.com/benchmarks.html`.

[3] Hash function. Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Hash_function`.

[4] Heritrix Wiki. `https://webarchive.jira.com/wiki/display/Heritrix/Heritrix`.

[5] OpenSSL Wiki. `http://wiki.openssl.org/index.php/Main_Page`.

[6] OpenWayback. IIPC website. `http://netpreserve.org/openwayback`.

[7] Oracle Solaris ZFS Administration Guide. `http://docs.oracle.com/cd/E19253-01/819-5461/`.

[8] The BA web archive on GitHub. `https://github.com/arcalex`.

[9] Web Archiving. IIPC website. `http://netpreserve.org/web-archiving/overview`.

[10] Proposal for Standardizing the Recording of Arbitrary Duplicates in WARC Files. IIPC internal document, September 2013.

[11] M. Burner and B. Kahle. *ARC File Format Reference*, September 1996. `http://archive.org/web/researcher/ArcFileFormat.php`.

[12] N. Clarke. *Java Web Archive Toolkit (JWAT) Documentation*, October 2012. `https://sbforge.org/display/JWAT/Documentation`.

[13] J.-L. Gailly and M. Adler. *zlib 1.2.8 Manual*, April 2013. `http://www.zlib.net/manual.html`.

[14] J. A. Kunze, A. Arvidson, G. Mohr, and M. Stack. *The WARC File Format*, January 2006. `http://archive-access.cvs.sourceforge.net/viewvc/archive-access/archive-access/src/docs/warc/warc_file_format.html?revision=1.10`.

[15] M. Shibata. Web archives of devastated area sites and deduplication project. IIPC General Assembly presentation, 2011. `http://www.netpreserve.org/general-assembly/2011/Overview`.

[16] K. Sigurdhsson. Managing duplicates across sequential crawls. In *6th International Web Archiving Workshop (IWAW06), Alicante, Spain*, 2006.

[17] M. Stevens. Cryptanalysis of MD5 and SHA-1. Centrum Wiskunde en Informatica (CWI), Amsterdam, the Netherlands. `http://2012.sharcs.org/slides/stevens.pdf`.