Studies on the scalability of web preservation

Rory Blevins
Tessella
26, The Quadrant,
Abingdon Science Park
Oxfordshire
OX14 3YS UK
Rory.Blevins@tessella.com

Ismail Patel
Tessella
Chadwick House,
Birchwood Park,
Warrington,
WA3 6AE UK
Ismail.Patel@tessella.com

Jack O'Sullivan
Tessella
26, The Quadrant,
Abingdon Science Park
Oxfordshire
OX14 3YS UK
Jack.O'Sullivan@tessella.com

Ashley Hunter
Tessella
Chadwick House,
Birchwood Park,
Warrington,
WA3 6AE UK
Ashley.Hunter@tessella.com

Robert Sharpe
Tessella
26, The Quadrant,
Abingdon Science Park
Oxfordshire
OX14 3YS UK
Robert.Sharpe@tessella.com

Pauline Sinclair
Tessella
26, The Quadrant
Abingdon Science Park
Oxfordshire
OX14 3YS UK
Pauline.Sinclair@tessella.com

ABSTRACT

This paper describes a mechanism for improving the scalability of preservation actions on large linked archives, such as WARC and ARC files produced from the archiving of web sites.

To enable accurate but efficient preservation actions, information on the files embedded within a container object, such as the file formats of the embedded files, are aggregated and recorded as properties of the container object. This occurs during the ingest of objects into the archiving system, specifically at the characterization stage when files are identified and validated. To ensure that the details of all embedded files are also recorded, nested archives are recursively unpacked and their contents characterized to identify all files in a package. Information about the embedded files is then stored as properties of the container object: this allows us to efficiently aggregate information about the contents of a container as queryable properties of the container.

This storage of the embedded file type information on the container object reduces the number of objects and properties which have to be queried to perform a preservation action, such as migration to a more recent file type. The database can be queried for a specific file type, and all files of that type, and archives containing files of that type will be returned without needing to query each embedded object individually.

Archives containing files in need of preservation are temporarily unpacked and the files in need of transformation identified and migrated. Following the preservation action, the internal links within the archive are updated to maintain the integrity of the archive and the modified objects are re-ingested back into the system.

This approach results in minimal extra overhead at the ingest stage of preservation, but substantially reduces the number of entities which need to be queried to identify objects at risk when performing preservation actions. In the case of large web archives, this may be several orders of magnitude, producing a corresponding increase in performance and scalability.

KEYWORDS

Scalability, Web Archiving, Characterization

1. INTRODUCTION

Many organizations now regularly perform large scale web crawls [Pennock][1]. For example, Bibliothèque nationale de France (BnF) have been performing large scale archiving of web sites since 2002 and by 2011 had accumulated approximately 200TB held within 1.5 million ARC files [2]. These crawls have been managed using web crawling software: initially HTTrack [3], then Heritrix [4] and finally adding the NetarchiveSuite [5], developed by the Royal Library of Copenhagen and the University Library of Aarhus. As can be seen from the size of the accumulated collections, the actual process of collecting web sites can be performed in a reasonable time frame and thus already scales fairly well.

However, typically, such web crawls are not as well characterized as other digital material being ingested into an archival repository. The normal method of such characterization can vary but would typically involve [6]:

- Identification of the format of each file
- Validation of the format of each file
- Property extraction from each file
- Embedded object extraction from each file
- Recursive characterization of such embedded objects using the steps above.

The first three steps are relatively simple and straightforward since these crawls produce a container file in either ARC [7] or WARC [8] format with well-defined properties.

It is also relatively easy to extract the embedded objects from such a container (to produce the original files that manifested the sites crawled) and to characterize each file in turn.

This can produce a very large number of entities to be characterized. For example, the 200TB in the BnF collections are estimated to contain 50 billion embedded objects [9].

Each of these embedded objects can then be assessed to see if they can be adequately preserved in the long-term. This can be done, for example, by comparing the properties (e.g., format of the files) against known issues and then migrating problem files to a new format [6]. This has previously been performed on a small experimental scale [10] but not yet, to the best of the authors' knowledge, on a larger scale.

In fact, it is quite controversial whether or not such format obsolescence exists [11]. A recent study of web material [12] has shown that while most formats have persisted for a decade or more, not all do so and older versions of formats fade from popularity. In a sense, this argument is not relevant to this paper anyway since it is mainly focused on the scalability of bulk operations on large web archives collections and migration is just an example of such operations.

The size of the problem places at least two scalability demands on the ability to preserve websites after crawling that are addressed in this paper:

- Ability to characterize such a large number of files
- Ability to use such properties to determine future migration strategies.

This is caused by both the amount of computing resources needed to characterize, say, 50 billion entities and the ability to cope with the amount of information that such characterization produces and still make it useful in future preservation actions in a timely manner.

The latter issue (i.e. coping with this amount of information) occurs because it will strain the ability of any indexing system to enable searches to be made that can return information in reasonable timescales. Given that the quantity of material on the web is still rapidly increasing and future scans are likely to be more frequent and more comprehensive, this problem is likely to become more pronounced over time, probably outpacing improvements in indexing capabilities.

In this paper we describe an approach whereby we break the problem down into two parts to remove this indexing issue:

- Describing properties at the container level in sufficient detail to determine whether the container requires some action to be applied to it or its content.
- Dealing with each container (and its contents) in turn.

This approach is described in more detail in section 3, and the impact on the characterization process is described in section 4. Section 5 describes the impact on preservation actions (using format migration as an example). Finally, some general conclusions are drawn.

2. METHOD

This work has been carried out using Tessella's Safety Deposit Box (SDB) software. This has an existing suite of web crawling and web characterization functionality that enabled the specific problems to be addressed efficiently. The testing in this paper used version 4.3 of this software.

Performance testing was carried out using an Amazon EC2 M1 medium instance; a single-core Linux instance with the approximate processing power equivalent of two 1.2 GHz Opteron Processors [13].

SDB is commercial software, however most of the tools described in this paper are open-source and the methodology described in this paper should be generally applicable to the preservation of web archives irrespective of the underlying software used to implement the digital preservation repository.

3. CONTAINER VS EMBEDDED OBJECT PROPERTIES

ARC files were developed by the Internet Archive to enable efficient storage of data from web crawls and other archives of website data. WARC files are an ISO-certified extension of this format which allows recording of additional information, such as HTTP request headers and additional metadata (including file conversion records, which hold metadata on files which have been converted into a different format). They have the same basic format: a header block, followed by a series of URL records, which may themselves be compressed with a compression algorithm such as gzip. While they are efficient at storing the results of web crawls, accessing their embedded files requires temporarily unpacking the objects, which can be computationally expensive in the case of large archives.

To correctly preserve the objects embedded in the archive files, a preservation system must be able to identify and characterize both the archive container object, and the objects contained within the archive. It must later be able to query key properties of the embedded objects to determine if they are in need of preservation actions, such as file format migration.

Standard archiving and storage systems can either ignore the contents of container formats, or attempt to record properties of all embedded objects. If a system does not hold information about the individual objects embedded in archive files, any attempt to preserve the archived files, such as migrating them to a newer file format, will either ignore the embedded files completely or else require the system to extract all files from all stored archives to determine which files need preservation actions to be applied. Alternatively, if a system stores a complete set of technical metadata information on each embedded object in the database, accurate searching is possible, but this can result in a very large number of entities which cannot be queried within a reasonable timeframe.

In this study, to reduce the number of objects which have to be queried to locate files at risk, we associate queryable information about the embedded objects as properties of the container object. In the case of file format migration these properties would be the formats of the files embedded in an entity, each file format being stored as a separate property of the container object with a name such as “this container contains objects of type”, with the value being one of the file formats embedded in the object.

This approach requires two modifications to standard workflows: firstly, the characterization process must correctly characterize embedded files and associate the required properties of these files with the container. As archives may themselves contain other archives, this process must be recursive, and characterize all files in all archives contained in a particular object.

Secondly, the preservation process must search for embedded objects in need of preservation by checking the properties of each container object to determine if it contains objects in need of preservation, and then unpack and process those archives which do contain objects in need of preservation.

Because this approach requires that only each container is queried, and not every embedded object, considerable reductions in the number of entities which need to be queried can be achieved. In the case of web archives containing many hundreds or thousands of objects, which is not uncommon in the case of archived web crawls, this can result in a reduction of several orders of magnitude in the number of entities queried, producing a concomitant increase in the speed of identifying objects in need of preservation.

4. CHARACTERIZATION OF WARC FILES

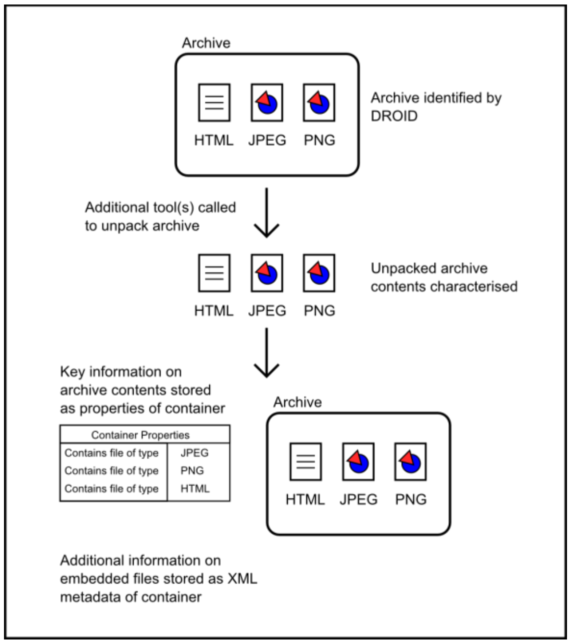
Archive
HTML
JPEG
PNG
Archive identified by DROID
Additional tool(s) called to unpack archive
HTML
JPEG
PNG
Unpacked archive contents characterised
Key information on archive contents stored as properties of container
Container Properties
Contains file of type JPEG
Contains file of type PNG
Contains file of type HTML
Archive
HTML
JPEG
PNG
Additional information on embedded files stored as XML metadata of container

Figure 1.

Schematic of basic steps to aggregate information on archive contents

4.1 Ingest

To correctly preserve web archives, it is necessary to ingest them into the archiving system. This requires a number of steps which are usually automated via a workflow to allow the efficient ingest of objects. In the case of a web archive, the steps required for ingest will typically involve:

- Crawling a given URL and creating a submission information package (SIP) for ingest into the archiving system
- Checking the produced SIP for viruses
- Checking the integrity of the produced SIP: for example, that the number of files in the SIP matches the number of files in the associated technical metadata
- Characterization of the files in the SIP, as described below.
- Storage of the physical files on storage systems
- Storage of the SIP metadata in the repository database

Simultaneously with the ingest, the system also generates XML metadata which describes the ingested objects, their relationships and key properties (e.g. Significant and/or Transformational Information Properties). This metadata is also stored in the database on successful ingest.

4.2 Characterization of files

Characterization is one of the key processes in ingest, and typically involves three steps: identification, validation and property extraction.

To characterize the files embedded in WARC or ARC files, the system must first identify the archive file. SDB identifies files using the open-source DROID (Digital Record Object IDentification) tool, originally developed for The National Archives [14]. DROID identifies files from their byte sequences by searching for signatures specific to file types. The current version of DROID is capable of detecting over one thousand different file types, and its signature definitions are continually updated to improve DROID’s capabilities. If files are not identified by DROID, additional tools may also be called.

A file whose format has been identified with DROID will have its technical metadata updated to associate it with a specific Persistent Unique Identifier (PUID) as defined in the PRONOM technical registry, a publically available registry of technical information on file formats. As well as PUIDs, PRONOM and other technical registries provide technical information for the preservation of different file formats. For example, this can include software tools for validation, extraction of key properties or extraction of embedded objects for specific file formats. They may also provide information on tools and pathways for migration between different file formats.

SDB incorporates the data from PRONOM in its own technical registry, which it uses to determine the appropriate tools for characterizing and migrating each of the different file formats.

Following identification, additional tools will be called to validate file formats and to extract key properties of files to ensure accurate long term preservation of the object can take place within the managed digital repository. Information from each of these steps is written to the metadata element which represents each file.

4.3 Extraction of Embedded files

After a file has been identified, characterized and undergone property extraction, the registry is checked to determine whether object extraction tools exist for each object in the SIP. Once an object extraction tool has been identified for an archive file format, such as a (W)ARC or ZIP files, the tool is called to extract the contents of the container into a temporary work area. Files extracted from the WARCs are passed to DROID for identification. DROID will attempt to uniquely identify the file format of the file and, if a file is identified, may pass it on to further tools (determined by querying the technical registry) which will validate the file format and extract properties of the object to be maintained as technical metadata within the repository system.

This information is stored as part of the XML metadata of the container object, allowing information on the objects inside a container to be retrieved without recharacterizing the entire contents of the web archive container file.

As discussed in section 3, to enable efficient preservation actions, key properties of the embedded files are aggregated and stored as properties of the container object. For example, the file types of embedded files are stored as separate individual properties of the container file for use in migration and other archival operations that require the efficient identification of archives containing specific file formats.

This process of extraction and characterization is recursive: if archive files are found in the unpacked archive, these too are unpacked and their contents are in turn characterized. Key properties are recorded for embedded archives, both in the metadata entry for the embedded archive, and as part of the properties added to the parent container, so that the original archive file has properties which aggregate information from all levels of embedded file within the archive. For example, in the case of recording file format information for file format migration, the top-level container will have properties, including transformational information properties, representing each of the file formats embedded in any of the contained files. In addition there will be entries in the XML metadata for each embedded file, and in the case of nested archives, this metadata will include properties representing each of the file formats embedded in the nested archive.

4.4 Conceptual Characterization

In addition to the above physical characterization, web sites pass through a conceptual characterization process. This identifies the existence of technology-independent information objects (e.g., a web page, an image or a document) that can be manifested in a variety of different technologies (each potentially changing the number and arrangement of files as well as file formats). This allows the identification of links between these information objects (e.g., the link between a web page and an image). This is then subsequently used to identify information objects that might need modification even though it had not been directly affected by a preservation action (e.g., the need to edit a HTML page if an image has changed format and thus extension).

4.5 Performance of Characterization

To measure the performance of the characterization step on typical web archives, we performed some basic benchmarking to test whether the characterization step was sufficiently fast for efficient web archiving. This involved running SDB 4.3 on the single core cloud computing instance described above, measuring the performance during ingest of a selection of public websites.

The first thing to note is how the relative speed of characterization compared to other ingest steps. Analyzing the time spent on each ingest step (Figure 2) clearly shows that the dominant steps are:

- Crawling (28%)
- Thumbnail creation (58%)
- Characterization (8%)

Web crawling is known to be a limiting step, but the elapsed time is largely dependent on wait times and bandwidth issues.

Creating thumbnails is very process intensive since SDB creates an image of all archived HTML files as they originally appeared, complete with any embedded objects. If increased throughput is required the thumbnail creation step can be disabled, which would result in a considerable improvement in the overall rate of ingest.

Characterization took just 8% of the time of a typical ingest equating to a typical speed of 60MB/min. This is a considerable

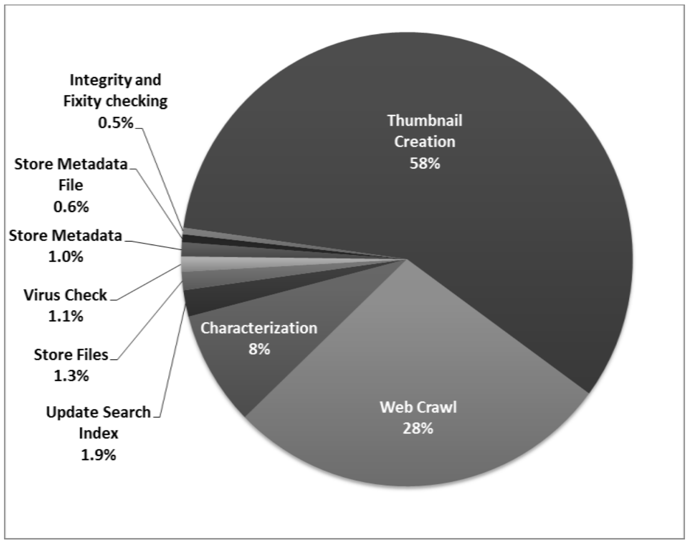

Figure 2

Typical division of time to process ingest steps, as a proportion of total ingest time improvement on the results reported by the State and university of Denmark using FITS [15] which averaged below 4MB/min. We are not sure of the reason for the discrepancy. One possible reason is that, while SDB will only use a single tool for each characterization process, FITS can attempt to use several. Another possibility is the way jobs are packaged within the workflows of each system could be different.

Figure 3 shows the breakdown of the percentage of the time taken to perform various tasks within the characterization process. This shows that decompressing the WARC files is the most time-consuming of the tasks, taking 51% of the time taken for characterization. It took a total of just 4% of the time for DROID to identify the WARC files and a further 5% for DROID to identify the embedded files post extraction. Jhove and other tools (e.g., SDB's built-in XML validator) took most of the rest of the time to validate and extract file properties (31%). Conceptual characterization took the remaining 9% of the time.

All other ingest processes (i.e. excluding crawling, thumbnail creation and characterization) took just 6% of the time. This includes creating a SIP from the crawl, performing initial quality control checks (for SIP integrity, fixity and virus checks) plus the overhead in storing the resulting content files in a file store, storing the metadata in both a database and a file store and updating a SOLR search index.

One thing that is clear is the extra process of aggregating embedded object properties still allows efficient characterization and ingest of large archives.

4.6 Scaling up

This study did not have access to significant hardware, only using a single-core medium size amazon EC2 cloud instance for benchmarking, which is considerably underpowered compared to the multi-server setups used in many modern web archiving systems. This makes it hard to quantify exactly how this approach would scale if deployed on more significant hardware.

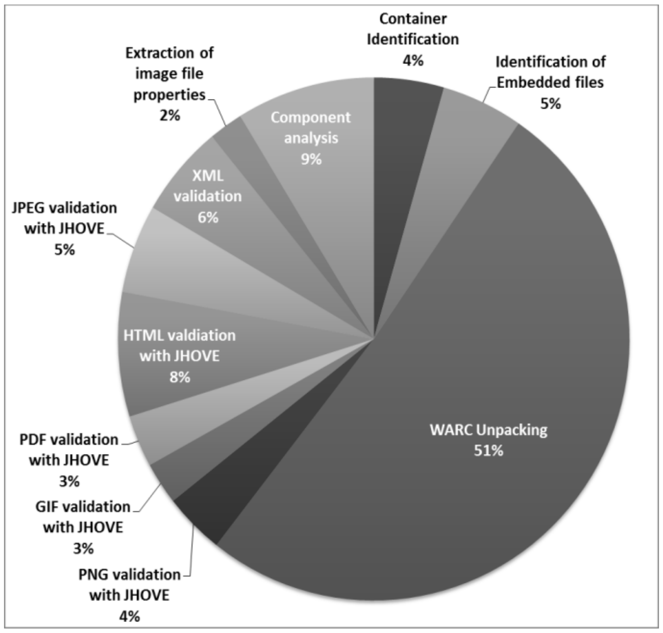

Figure 3

Proportion of characterisation time taken by individual tools

However, it is known that the approach used in this study has been used to achieve total ingest (including characterization) rates in excess of 20TB/day by FamilySearch using relatively modest processing power (2 Dell 2950 servers each with 2 Intel Xeon E5430 processors, with 4 cores clocked at 2.66GHz and 32GB RAM) [16]. Even though this has modest cost for a production system (c. $10k at today's prices) it is many times the processing power of the single core benchmarking instance used in this study. Also, unlike in this study, it enables ingest (and particularly characterization) of different content to be run dozens of time in parallel. Hence, while that study related to much larger files (typically 10-30MB) it did find that the fundamental limit on scalability was the ability to read files fast enough from disk and transfer them across the network and not the processing speed of the server. This required the use of high performance switches and drive arrays to reduce this bottleneck.

While processing more files is likely to lead to a higher overhead, it is still reasonable to expect the method proposed in this study will also parallelize well by adding more computing cores and more server machines.

5. MIGRATION INSIDE WARC FILES

To take advantage of the stored embedded object properties, preservation actions, such as file format migration, must use these properties to reduce the number of entities which must be queried to determine which entities require action.

As with ingest, file format migration in an archiving system normally requires a number of steps to occur through an automated workflow, although a number of key steps can also require human intervention:

- File formats at risk are selected, either by manually selecting a list of PUIDs to migrate or by choosing a risk threshold above which to migrate files
- Files to migrate are chosen from the files at risk
- Pathways to migrate the files at risk are chosen
- The files are migrated, as described below
- The SIP is re-ingested into the database, in a similar manner to the ingest workflow.

5.1 General approach

To migrate files inside (W)ARC containers we used the approach of breaking the problem down into parts:

- Finding which of the millions of containers are in need of a preservation action to be applied to them
- Determining which entities within each container then need action, and extracting these from the container to a temporary working area of the system.
- Performing that action (in this paper a migration will be used as an example of such an action)
- Re-wrapping the content into a new container

This aggregation of the file types of the embedded objects as properties of the container results in a dramatically smaller number of objects to query during preservation actions. This in turn results in a significant improvement in the scalability of performance related to the preservation of web archives and other container formats.

5.2 Finding containers in need of action

In the case of file format migration, file formats at risk are selected either by manually choosing specific file formats which are at risk, or by selecting a risk threshold, a value which indicates how at risk of obsolescence a file is. To determine which file formats are at risk using a risk threshold value, the archiving system must query a technical registry to retrieve file formats which are above this risk threshold. The risk threshold for each file format is determined by answering a number of risk questions about each file format in the technical registry, such as whether it is an open-source or proprietary format. The responses to these question are then weighted (weightings for each question are set by the system user, depending on their requirements) and combined to create a risk value.

Either method produces a list of file formats which require migration, identified by their PUIDs. To locate files in the repository requiring preservation we then query the database for:

- Files of a file format type at risk
- Files which have a property "contains file(s) of type" matching one of the formats at risk.

As technical metadata about these file objects are stored in the system's database, these can be queried easily with a SQL query on the relevant database tables. The resulting list of files and containers at risk is passed onto the next step of the migration process: determining which individual files require migration.

5.3 Determine at risk content within a container

Once an archive has been identified as requiring migration, all files within it are extracted into a temporary work area. As with characterization, this is a recursive process, as archive files may in turn contain further archives. All archive files in a particular archive are in turn unpacked. From this temporary unpacked copy of the archive, the files in need of migration need to be identified.

The properties on a container only indicate that an archive contains a particular file format at risk and not which files within the archive are of that format. This means that once a container containing files at risk has been determined, individual files at risk within the container must be identified. This is achieved by parsing the XML metadata associated with the container for elements representing embedded objects of the file format at risk, or embedded elements which also have a property indicating that they contain files at risk. This occurs recursively, to identify all files at risk even in multiple nested archives.

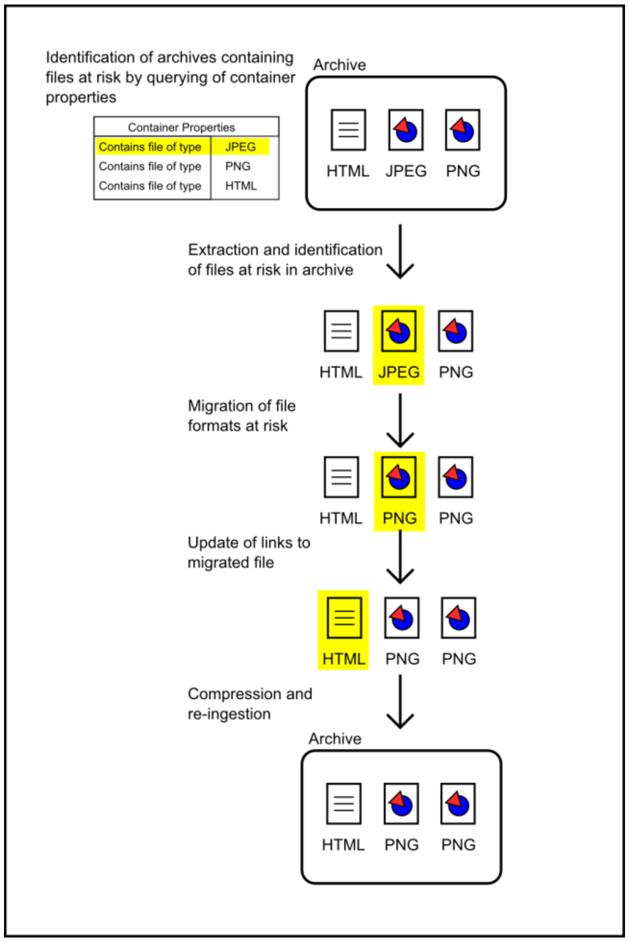
Identification of archives containing files at risk by querying of container properties
Container Properties
Contains file of type JPEG
Contains file of type PNG
Contains file of type HTML
Archive
HTML JPEG PNG
Extraction and identification of files at risk in archive
HTML JPEG PNG
Migration of file formats at risk
HTML PNG PNG
Update of links to migrated file
HTML PNG PNG
Compression and re-ingestion
Archive
HTML PNG PNG

Figure 4

Schematic of basic steps required to perform file format migration on embedded files. In this example, a JPEG to PNG migration is performed.

5.4 Performing migration

Once files at risk have been identified, the user chooses a pathway to transform files at risk into more current formats. Possible migration pathways for a file format at risk are obtained from the technical registry, and one is manually chosen for each format at risk by the user.

The exact pathway and tool used for migration depends on the pathways available in the technical registry, but the general procedure is simple: the tool is invoked, either through its API or through the command line, to convert the file into the new file format in a temporary work area.

Migration requires that not only are particular file formats migrated successfully, but that the conceptual components that they are part of are also migrated successfully: for example, in the case of migrating an image embedded in a web page, not only must the image be migrated, but the integrity of the webpage must be maintained. This involves updating links to the image maintained by other information objects within the archive so that the migrated format is correctly linked from those objects.

How exactly these updates are managed depends on how the archive file is recompressed, as discussed below.

5.5 Validation of migration

Following migration each conceptual entity identified during conceptual characterization must be checked to ensure that it still exists, still has the same links to other conceptual entities and still has the same transformational information properties. For example, for images, these properties may include the histogram spread of red green and blue pixels while for documents it typically includes the number of pages. To validate successful migration, physical and conceptual characterization is performed on the migrated files. These properties are compared to the original, to confirm that they have not been changed by the migration, which would indicate a failure in migration.

5.6 Rewrapping content back in containers

At the end of the migration, it is important that a (W)ARC file is recreated so it can be utilized by the appropriate access workflows, e.g., in the Wayback machine [17]. This means that the (W)ARC containers need to be recreated using the appropriate combination of migrated and non-migrated files. As discussed earlier, to maintain the integrity of migrated web pages, links to the migrated files must also be updated appropriately.

The reconstruction of WARC files creates a specific practical problem: the specification for WARC files includes protocols for migrating files inside a WARC container and recording the details (provenance) of the migration in conversion record metadata in the WARC. However, most WARC access workflows, such as the Wayback machine, do not currently support conversion records, so WARC files migrated in this way will not be properly displayed. This required the development of two different workflows for creating migrated WARC files: one, which is formally correct according to the WARC standard, and maintains the integrity of the WARC schema, and a second which is more pragmatic, and produces a file that can be displayed correctly by current WARC viewers. This pragmatic workflow can also be used for the migration of container formats which do not support conversion records, such as ARC files.

If the formally correct workflow is chosen, then the workflow creates conversion records for each migrated file, which reference the original WARC file pre-migration. To reconstruct the full archive, both the original WARC file and the new WARC file containing the conversion records are required. Links and other references to converted files are not updated, as a strict implementation of the WARC viewer should be able to retrieve the most recent version of the updated files.

If the pragmatic workflow is chosen, then the archive simply replaces the unmigrated version of the file in the archive with the new, migrated version. To maintain the integrity of the migrated webpages, links are updated where possible to refer to the migrated files, for example, updating files extensions where necessary. A new archive file is created which contains migrated files, files modified because they reference migrated files and files from the original archive which have been unaffected by the whole process..

In either workflow, once the new (W)ARC containers have been created, they are re-ingested into the archive, and the associated archive object metadata is updated to reflect the provenance of the transformation action that has been performed.

5.7 Limitations on validation

The migration process involved the following conceptual steps;

1. Unpacking of the (W)ARC files
2. Migrating the at risk files and modifying affected files
3. Repacking of the (W)ARC files

Ideally it would be possible to directly compare the transformational information properties of the (W)ARC file as it exists before step 1 and the (W)ARC file existing after step 3. The original characterization does indeed take place before step 1 but it uses the same unpacking process as step 1 before characterizing so it is equivalent to taking place after step 1. The same is true in reverse for the second characterization step meaning that the only true verification of the above process is taking place by comparing properties produced before and after step 2. This is probably reasonable since the process of packing and unpacking (W)ARC files is unlikely to lead to information loss or data corruption. However, it might still be better to have alternative implementations of packing and unpacking in migration and in characterization so that the process could be independently verified.

6. CONCLUSION

By using the initial characterization process to aggregate information on the objects contained in a web archive, and by storing these aggregated properties as properties of the container object, we considerably reduce the number of entities that need to be searched to perform preservation actions, and hence increase the scalability of web preservation, while maintaining efficient characterization during ingest.

While described using file format migration as an illustrative example, this method is not limited to describing file formats: any property which can be aggregated across the archive could also be recorded and retrieved using this method.

While this approach was developed to deal with the challenges of large scale web crawls, it would also have advantages across a large number of other situations in which characterization and file format migration (or other, similar operations) need to be performed across embedded file formats, provided that suitable software tools are available for identification/validation of the container objects and extraction of the embedded files formats. For example, the same approach has been successfully applied to the preservation of other container formats, such as .zip files.

7. ACKNOWLEDGMENTS

This research was partly funded by the European Union as part of APARSEN - Alliance Permanent Access to the Records of Science in Europe Network- under FP7-ICT-2009 agreement 269977.

8. REFERENCES

[1] For a recent review see "Web Archiving", Maureen Pennock, DPC Technology Watch Report, March 2013

[2] Clément Oury, Sébastien Peyrard. From the World Wide Web to Digital Library Stacks: Preserving the French Web Archives. In Proc. iPRES2011, Singapore, 2011.

[3] www.httrack.com/

[4] crawler.archive.org/index.html

[5] netarkivet.statsbiblioteket.dk/

[6] Robert Sharpe. Active Preservation of Web Sites. In Proc. International Web Archiving Workshop IWAW 2010, Vienna, 2010.

[7] http://archive.org/web/researcher/ArcFileFormat.php

[8] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44717

[9] Information from Sébastien Peyrard in panel session at iPres 2011.

[10] Strodl S., Beran P. and Rauber A. Migrating content in WARC files 2009 The 9th International Web Archiving Workshop (IWAW 2009) Proceedings", (2009), 43 - 49

[11] Rosenthal, David S.H.; (2010) "Format obsolescence: assessing the threat and the defenses, Library Hi Tech, Vol. 28 Iss: 2, pp.195 – 210

[12] Jackson, Formats over Time: Exploring UK Web History, arXiv:1210.1714

[13] http://aws.amazon.com/ec2/instance-types

[14] http://www.nationalarchives.gov.uk/information-management/our-services/dc-file-profiling-tool.htm /

[15] http://openplanetsfoundation.org/blogs/2013-01-09-year-fits

[16] Jason Pierson, Mark Evans, James Carr and Robert Sharpe. Considerations for High Throughput Digital Preservation. In Proc. iPRES2011, Singapore, 2011, Page 267

[17] http://archive.org/web/web.php