Building Scalable Web Archives

Leïla Medjkoune; Stanislav Barton; Florent Carpentier; Julien Masanès; Radu Pop; Internet Memory Foundation; Amsterdam, Netherlands

Abstract

This paper aims at introducing the Internet Memory Foundation platform based on its distributed infrastructure and the associated tools and workflows that facilitate data management and preservation actions at large scale. IMF's main concern over the past years has been related to scalability issues in terms of crawling, indexing, preserving and accessing content. To answer these issues, the Foundation developed its own crawler and built a new infrastructure.

This paper aims at presenting our infrastructure and crawler and at sharing challenges met while building them as well as the approach taken to solve preservation issues inherent to scalable archives. It will also highlight new horizons arising for web archives in relation to analytics use cases.

Introduction

The Internet is a media of our time. Its importance as well as its estimated size grows continuously together with the diverse means of publication and an exponential number of publishers. As this ephemeral and heterogeneous content is a key resource for future generations, it is essential to tackle its intrinsic capture, access and preservation challenges. Many cultural institutions launched web archiving programs in the past years but most of them choose the selective approach rather than targeting larger portions of the web. If this remains valuable, we believe it is not sufficient to preserve significant fragments of the information deluge produced through Internet; archiving at large-scale therefore seems essential.

To answer these challenges and build a valuable memory of the web, the Internet Memory Foundation chose to develop a shared platform based on a distributed infrastructure and crawler, associated to tools and workflows that facilitate data management and access as well as preservation actions at large scale. This paper aims at introducing these and at sharing lessons learned as well as future developments planned.

Large-scale crawls

As introduced above, the growing size and complexity of the web and of content published on the Internet constantly raises new challenges. If more and more preservation institutions conduct a web archiving project (in-house or externalised), these projects consist in most cases in crawling regularly a short list of known websites or at most a portion of a given national domain. Reasons for doing so often come from legal restrictions or resources and budget limitations. But beyond these limitations, the technical aspect should also be taken into account as crawling at large-scale has proven to be anything but an easy process. It requires experimented human resources that can build adequate infrastructures, and that can implement tools and rather complex workflows. The crawl quality will then depend on the team knowledge of these tools and of the Internet that will enable them to avoid, as much as possible, unwanted content and to get the best crawl quality through optimised parameters. Indeed, the larger the amount to capture is, the more complex it is to keep a correct level of temporal coherence within the web archive and manage URLs discovered through what is commonly called the crawl frontier (URL store).

To attempt to solve these issues, the Internet Memory Foundation developed its own crawler, MemoryBot, in 2010 with the support of the EU funded LAWA project (http://www.lawa-project.eu/) and through an initial collaboration with the University of Milan [2], and constantly improved it since then depending on internal needs. MemoryBot is built in Erlang (with a combination of Python and C), a language commonly used for distributed, fault-tolerant systems. The data and process management are also facilitated by the use of Erlang built-in database, Mnesia, which centralises modifications and/or corrections and spreads them thoroughly within the system. MemoryBot is designed to be fully distributed (based on consistence hashing) and produces standard WARC files.

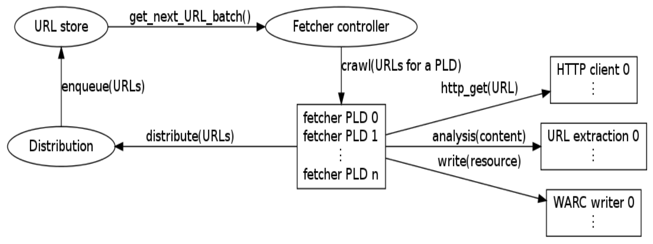

***Figure 1:** MemoryBot architecture*

Figure 1 shows the main processes on each cluster node. The rectangles are associated to processes with same function when the ovals represent individual processes or subsystems made of many processes.

The fetcher controller's role is to create fetchers that will then crawl a set of URLs. It requests the URL store for a batch of URLs, all belonging to the same pay level domain (PLD), resolves the domain name to an IP address and ensures no other fetcher in the entire cluster is crawling this IP address. It then spawns a fetcher and passes it the URL batch. Each fetcher receiving a batch of URLs to crawl also receives parameters to be applied (e.g. robots.txt file, politeness, etc.). This way each fetcher crawls URLs sent following the defined parameters and respecting the required politeness delay between each fetch.

For each resource, three main steps are performed:

1. Fetching (HTTP request);

2. Analysing the document according to its type in search of new URLs.
3. Writing the content plus extracted or derived information into a WARC file
4. Filtering according to the scope configuration before sending to the distribution module.

In order to be scalable at a reasonable cost, a crawler requires to be distributed and to have a URL store that can scale up to billions of URLs. The distribution has always been a top requirement as our initial goal was to handle the crawling of very large portions of the web (several Top Level Domains) much faster and in a more efficient manner than any other tested crawlers had allowed us to do till then. Large-scale crawls performed so far showed very good results, such as an impressive throughput and a rather low performance drop as a crawl progresses.

Figures 2 and 3 below show part of our crawler's monitoring interface on a running large-scale crawl. The period displayed on both figures is approximately of 10 days.

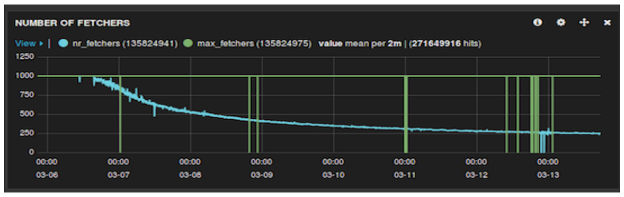

Figure 2: *Number of fetchers evolving during crawl*

Figure 2 shows how the number of fetchers drops over time as discovered resources are crawled. MemoryBot crawls dozen million resources per day with an average rate of around 100 WARC writes per second per server (figure 3).

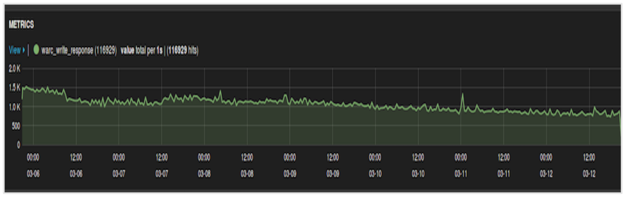

Figure 3: *Number of WARC writes per second*

As long as the number of URLs is small and everything fits in memory, traditional centralised databases are sufficient. However, when scaling to billions of URLs, to keep the number of servers and the amount of RAM low, the URL store design becomes critical. MemoryBot's URL store design borrows a lot from DRUM, as described in [1]. It relies on performing batch operations on sorted URL files, leveraging the good performance of sequential reads on low-end hard disks. MemoryBot URL store is therefore essential when targeting to achieve large-scale crawls. As a web archiving foundation, IMF required these large-scale crawls to reach a good level of quality. If our primary goal was indeed to handle large-scale crawls, the quality of these crawls remains critical for us in relation to our preservation and accessibility mission. To add what can be called archiving quality to MemoryBot crawls and to allow more flexibility in terms of scoping (e.g.: to fit researcher needs), we added a number of features to our crawler. Some were considered as a baseline requirement such as the support of HTTPS, retries on server failures or streaming of large files. It has a fast C implementation of a comprehensive and configurable URL canonicalization. It also allows to base crawling priorities upon URLs whitelisting and blacklisting or even trap detection as explained below. It enables detection of the real MIME type and of language documents and can extract metadata from HTML pages (e.g.: outlinks with type). It also employs a fully-fledged and extensible per-domain configuration framework with parameters including budget, minimum and maximum delay between two fetches. Crawler fetchers subscribe to updates of parameter values and use the new configuration immediately. This multi-store model therefore supports more complex requirements, such as for instance, fast revisit of RSS feeds to collect regularly newly published resources.

Further large-scale crawls are planned for 2014 that should allow testing and improving MemoryBot further. Several features could indeed be improved such as the management of discovered URLs for instance. The issue met with management of URLs discovered is amplified by the number of domains we crawl concurrently. For very large-scale crawls, there is always a risk of having too many URLs discovered in comparison with what can effectively be crawled. Among these, (i) many can be identified as crawler traps and should be avoided altogether; (ii) others would qualify as quality issues. Indeed, from the archiving and human perspective, the website visual structure (homepage, level 1, 2 etc.) is meaningful. Crawling URLs in a random order within one single PLD might lead to collect deep level pages and miss top ones (if not all URLs are exhausted during crawl).

(i) To avoid traps we currently use two methods, one being URL pattern identification and the other duplicate detection inside each PLD.

Dealing with (ii) requires supporting priorities in the URL store and determining the priority of each resource. MemoryBot has priority levels support in the URL store, the top priority levels being crawled more often than the lower levels. We already send to the top priority level all URLs with an empty path (assumed to be home pages), when all other URLs will be sent to lower priority levels. Finer grained prioritisation can be investigated, such as, implementing breadth-first within PLDs or even, applying classification methods.

Internet Memory platform

Our platform has been developed in collaboration with the Internet Memory Research, a spin-off of the Internet Memory Foundation, which is specialised in web-scale crawling and data extraction. The IMF infrastructure is based on a sophisticated distributed storage, relying on HBase (https://hbase.apache.org/) and HDFS (Hadoop file system) that automatically manages replication and fault-tolerance. IMF platform is designed to be scalable and can easily process billion of data items. Both the data repositories and the analytic processes operate in a distributed infrastructure, which can be extended to cope with very large datasets, up to web scale volumes.

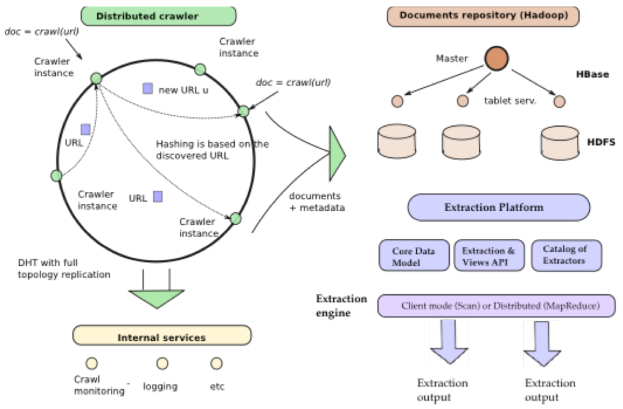

Figure 4: *IMF platform*

From the web archiving perspective, our platform offers several advantages in comparison to traditional storage and management system. Once data is crawled, all WARCs produced by MemoryBot are stored directly into HBase/HDFS: all WARCs metadata are extracted and stored into HBase before storing the WARC itself into HDFS. At this stage, the system takes care of data replication automatically; ensuring data is safe and preserved. This means no tool or additional workflow is required to replicate and preserve data and metadata, which is of great importance when dealing with a very large amount of resources. The use of HBase to access crawled content also proved to provide better performance at access time as the storage structure eases and speeds up search within web content.

In addition to improving existing access and storage replication, this new infrastructure also facilitate indexing processes as well as data process (made through batch process) and opens new horizons in terms of management and access to data collected at large scale such as for instance data characterisation and data mining (analytical APIs, filters, extractors, etc.). Indeed, HBase is by definition a distributed key value store (multidimensional map) that allows:

- Inherent versioning (timestamped values)
- Real time access (cache, index, key ordering)
- Column oriented storage
- Seamless integration with Hadoop

This means an inherent temporal aspect for each data stored (versions) that is essential when looking at analysing web content within an historical web archive. It also means having several views on same data: raw, extracted and/or analysed level, which allows treatment of raw data as well as treatment of pre-processed data in a very easy and natural manner. A typical web archiving use case would be to extract MIME type information and store it into HBase to allow later fast process. Once data is stored it can be processed and reprocessed any time following any use case and need. Another example of a possible use case would be to perform full text indexing of crawled data, to extract text only content and store it into HBase. Once done, text resources can be processed in the same way and crossed reference with any other data or metadata available within HBase.

Extraction and filtering processes are organised to minimize the overhead of data access and network exchanges allowing more extensive characterization and data mining actions. Indeed, if systems are essential to process data at large scale, infrastructure costs cannot be ignored. In the past years, IMF worked actively on its infrastructure (manufacture and design) through a collaboration with a technology company specialised in Green IT, NoRack (http://www.no-rack.com/), to develop a new generation of servers and infrastructure (see figure and build an efficient and green data center dedicated to massive storage. This allowed us not only to reduce our storage and processing costs, which is crucial when preserving petabytes of data and providing access to it, but also to build a highly scalable infrastructure with a very low consumption that does not require cooling (free cooling system). The free cooling system is enabled through a very low thermic diffusion (for 72 nodes, our data center is set between 5300W and 6300W depending on servers' type and configuration) and thanks to an innovative design, which turns heat toward a simple ventilation system. The Power Usage Efficiency is under 1,2, which is quite low compared to the average of 1,65 according to the Uptime Institute 2013 survey [3] (survey of around 1000 data centers, the average was of 1,89 in 2011). All archives users share this infrastructure, which ensures maximizing storage use, reducing the number of devices requires and saving energy and cost while allowing virtualization of processes, better performances and faster processes.

Figure 7: *NoRack servers*

Although moving to these new infrastructure and storage is quite challenging, we believe it will allow us to scale in a faster manner and will enable us to share more content with web archive users, up to providing specific access to researchers, not only to access content but also to experiment with collected resources.

Long-term preservation and quality control

Beyond characterisation, data mining and access challenges, the long term preservation and quality control of crawled web content also gets more challenging as the size of web archives grows. Indeed, as crawlers and storage infrastructures allow a more comprehensive capture of the Web, quality assurance methods and tools must evolve.

As part of the EU funded project Scalable Preservation Environment (SCAPE: http://www.scape-project.eu/), the Internet Memory Foundation takes part to experiments and developments aiming at improving management, characterisation and quality assessment of data at large scale, with a focus on web resources. If characterising content is a pressing issue, controlling the quality of crawls is another one. Here again, the growing size of crawls and web archives makes it difficult to evaluate the quality of content crawled. Developing scalable quality assurance methods is therefore crucial. Indeed, quality assurance work is currently very costly and time consuming as it is most of the time done manually, if done at all. As time goes by and technologies evolve, questions about the accessibility to content crawled also becomes more and more pressing for preservation institutions. Within the SCAPE project, and as a leader on the QA applied to web content work package, IMF worked jointly with the University Pierre et Marie Curie (http://www.upmc.fr/), to enhance and adapt a visual and structural comparison tool with the aim of applying it to web archiving. Our QA team worked closely with the UPMC team and annotated hundreds of pairs of URLs to train the visual comparison open source tool, Pagelyzer (http://www.scape-project.eu/tools).

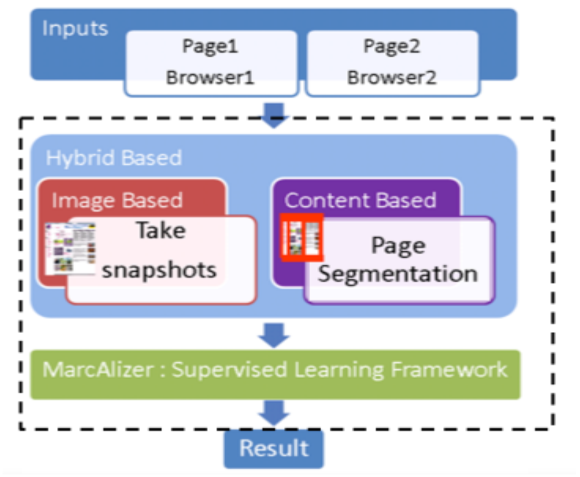

Figure 6: *Pagelyzer schema (UPMC)*

The idea behind this is not only to cut QA costs but also to improve processes by applying an automated layer to our QA methodology. QA applied to web archives can currently be conducted by several means. It can be done by checking crawl statistics during or after a crawl, to find out if any crawling issue arose. Such statistics can for instance be the size of the crawl in comparison to known figures (estimated size of such national domain). Another type of QA consists in checking that resources belonging to a targeted domain are indeed captured. To automate this process, some web archives tried to create tools to detect missing content such as 404s by using proxies or by executing web pages. If all these methods are extremely valuable when it comes to checking selective and/or large-scale crawls, these do not allow checking rendering issues met by potential end users.

The Internet Memory, as part of its web archiving activities applies several QA methods, including a visual comparison of crawled web pages to their live version with a minimum of two different browsers. Our trained QA team checks samples of content crawled in a methodical manner, based on our knowledge of crawling and access technology used, as soon as possible after a crawl is complete. Within SCAPE, we tried to mimic this human visual comparison of web pages to make it applicable at large-scale. The Pagelizer tool allows comparing two versions of a web page rendered through an access tool automatically. This can be done visually, structurally or using both comparison algorithms. As shown in the Pagelizer schema provided by UPMC (figure 6), the tool generates a similarity score once comparison is made that classifies pairs of URLs controlled into two lists: similar pages and dissimilar ones. As scores are based on our QA team annotations, used to train the Pagelyzer tool, these are very close to the human evaluation. This means that pages will only be classified as dissimilar if results are significantly different (for example if a resource is not rendered on the page). This tool thus leverages QA to the level of rendering detection issues and opens new possibilities in term of QA processes for regular basis checks as well as for longer-term preservation actions.

To allow processing at large scale, IMF developed a wrapper application around the Pagelyzer technology. This application orchestrates the main building blocks required for the comparison to be performed: Selenium framework (http://docs.seleniumhq.org/) that takes screenshots of web pages and Pagelyzer that performs comparison. The Selenium framework was chosen because it can manage several instances in parallel as well as different browsers and browsers' versions. This is a must have when checking rendering quality which can vary from one browser to another (and the same with browsers' versions) as explained above. The current workflow is as shown in figure 7 and as follows:

1. Web pages screenshots are automatically taken using Selenium framework (using one or several browsers and browsers versions).
2. Visual comparison is performed between pairs of screenshots using the Pagelyzer tool.
3. Rendering issues are detected within web pages, based on the comparison results.

The browser versions currently experienced and tested are: Firefox, Chrome, Opera and Internet Explorer. The comparison tool is implemented as a MapReduce job to parallelize the processing of the input. The input is a list of URLs with a list of browser versions that are used to render the screen shot. The output is made available through a set of XML files where each file represents one pair of browser shots. By implementing this wrapper application and implementing it on our platform, we already managed to cut the processing time to half in comparison to our first tests.

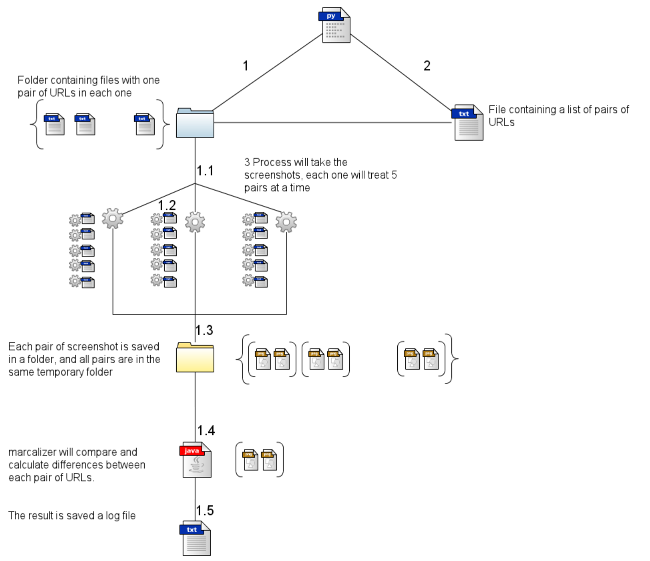
py
1
2
Folder containing files with one pair of URLs in each one
txt
File containing a list of pairs of URLs
1.1
3 Process will take the screenshots, each one will treat 5 pairs at a time
1.2
1.3
Each pair of screenshot is saved in a folder, and all pairs are in the same temporary folder
1.4
marcalizer will compare and calculate differences between each pair of URLs.
java
1.5
The result is saved a log file

Figure 7: *Wrapper application*

We currently test workflows where rendering is automatically checked through several browsers within our production platform with the aim of using this on a large sample of our regular crawls and of improving the whole web archive quality by limiting, through another mean, loss of content. So far tests were made on about 13 000 entries with the supported browser versions - Firefox and Opera. The correctness benchmarking made as part of the project activities showed a result of around 75% of correct assessment. The average time required to complete the whole workflow is around 4 seconds per pair compared (2 seconds per snapshots on average and 2second for the comparison). Although performance, robustness and correctness results must be improved by the end of SCAPE project, we believe this application will be of a real use to web archives in the future.

We also wish to test more complex workflows where rendering issues through browser and browser versions could be stored to enrich our knowledge of the technological landscape evolution and hopefully help triggering preservation actions in the future.

Conclusion

As web technologies evolve and the Internet grows in size and complexity, institutions such as web archives or national archives and libraries face new challenges. As one of them, the Internet Memory Foundation tries to tackle these challenges by taking the risks national bodies cannot always take. We build innovative means of capturing, managing, accessing and preserving web content at large scale. We believe the role of web archives is to preserve as much web content as possible, moving from the selective to the broad crawling approach. If doing so requires building complex and distributed tools and infrastructures, it also means enhancing tools allowing scoping, characterisation and quality assurance of web content crawled.

Crawling and preserving content at such scale also implies developing new methods and tools for this content to be easily accessible and useful to the research community as well as to end users. This is what IMF is aiming at through its shared platform. We indeed believe allowing a fast and easy analytical treatment of data will be crucial for research in the near future.

References

[1] H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov, "IRLbot: Scaling to 6 Billion Pages and Beyond". ACM Transactions on the Web, vol. 3 (2009).
[2] Paolo Boldi and Bruno Codenotti and Massimo Santini and Sebastiano Vigna , "UbiCrawler: a scalable fully distributed Web crawler". Softw., Pract. Exper. vol. 34, 711-726 (2004).
[3] Uptime Institute,"2013 Data Center Industry survey, 4, (2013)

Author Biography

Leila Medjkoune is graduated from the French National School of Library and Information Sciences (ENSSIB) with a Master's Degree. She joined Internet Memory in 2007 and is currently Head of Web Archiving Services. She manages all stages of quality archiving, from capture, to quality assurance and designs new services, tools and methodologies to improve Web Archives. She actively participates to EU funded projects, such as LivingKnowledge, LIWA or SCAPE, by offering a functional expert view.