Infrastructure for Supporting Exploration and Discovery in Web Archives

Jimmy Lin[1,2,3], Milad Gholami[2,3], Jinfeng Rao[2,3]

[1]The iSchool, [2]Institute for Advanced Computer Studies, [3]Dept. of Computer Science
University of Maryland, College Park

jimmylin@umd.edu, mgholami@cs.umd.edu, jinfeng@cs.umd.edu

ABSTRACT

Web archiving initiatives around the world capture ephemeral web content to preserve our collective digital memory. However, unlocking the potential of web archives requires tools that support exploration and discovery of captured content. These tools need to be scalable and responsive, and to this end we believe that modern "big data" infrastructure can provide a solid foundation. We present Warcbase, an open-source platform for managing web archives built on the distributed datastore HBase. Our system provides a flexible data model for storing and managing raw content as well as metadata and extracted knowledge. Tight integration with Hadoop provides powerful tools for analytics and data processing. Relying on HBase for storage infrastructure simplifies the development of scalable and responsive applications. We describe a service that provides temporal browsing and an interactive visualization based on topic models that allows users to explore archived content.

Categories and Subject Descriptors: H.3.4 [Information Storage and Retrieval]: Systems and Software—Distributed systems

Keywords: HBase; Hadoop

1. INTRODUCTION

Web archiving refers to the systematic collection and preservation of web content for future generations. Since 1996, the Internet Archive has captured and made publicly accessible hundreds of billions of web pages totaling over ten petabytes. Today, many libraries, universities, and other organizations have ongoing web archiving initiatives [10]. At an intuitive level, it is obvious why such activities are valuable. The web has become an integral part of our daily lives and captures our "collective memory", recording everything from major world events to the rhythm of commerce. Even personal minutiae are valuable in that they offer a snapshot of our society, much in the same way that a diary from the 17th century provides insight into what the world was like back then. It would not be an exaggeration to say that the web has become an important part of our cultural heritage, just as worthy of preservation as old books and historical buildings. Since web pages are ephemeral and disappear with great regularity [19], the only sure way of preserving web content for posterity is to proactively crawl and store portions of the web.

Ultimately, web archives are only useful if they provide mechanisms for users to access their contents. These may be historians digging in the digital past, digital humanists reinterpreting historical documents, lawyers looking for evidence of malfeasance, or journalists trying to recover lost stories. In this respect, we feel that web archives fall short of an unequivocal success. While the Internet Archive boasts impressive access statistics,[1] a recent presentation by Hockx-Yu of the British Library paints a more muted picture of scholarly use [13]. The usage statistics of many national web archives are depressingly low. Although ethical and legal restrictions hamper broad web-based access, there are additional technical hurdles that prevent web archives from living up to their full potential.

We believe that there is a lack of responsive, scalable tools to support exploration and discovery in web archives, and that the construction of such tools requires modern "big data" infrastructure to provide a stable foundation. To this end, we present Warcbase, an open-source platform for managing web archives.[2] Our platform takes advantage of HBase, the open-source implementation of Google's Bigtable [5], for storing web content, metadata, and extracted knowledge. We rely on Hadoop, the open-source implementation of the MapReduce programming model [8], for scalable analytics and data processing. Warcbase provides browsing capabilities that allow users to access historical versions of captured web pages, similar to the Wayback Machine. On top of this infrastructure, we describe processing pipelines for analyzing web content and an interactive visualization based on topic models that supports exploration and discovery. While none of the techniques presented in this paper are novel, we feel that our contribution lies in articulating the design of a web archiving platform that integrates a number of mature open-source technologies.

2. BACKGROUND AND RELATED WORK

Web archiving is a complex activity that includes many interlocking processes and workflows, many of which are not

[1] twitter.com/brewster_kahle/status/364834158285041665

[2] warcbase.org

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2579045.

primarily technical in nature. Any effort begins with deciding what material to archive, the scope of collection efforts, and the periodicity of acquisition attempts—these decisions are primarily based on the objectives of the organization and the amount of resources available. Then comes the task of crawling itself, a non-trivial engineering feat. Despite its conceptual simplicity, crawling at volume is non-trivial, and many aspects of crawl management are more of an art than a science. These aspects of web archiving are not the focus of our work—we simply assume that web content has already been harvested and stored in standard WARC or ARC containers.

Researchers and practitioners have developed a number of open-source tools for managing web archives. The most basic of these capabilities is to browse archived content—to view a particular version of a web page, to move forward and backward in time to examine different captured versions, and to follow links to contemporaneous pages. The open-source Wayback Machine[3] is the most popular and widely deployed application that provides these capabilities. However, the system was primarily designed to run on a single server and engineered as a monolithic, tightly-integrated stack—as a result, it can be difficult to scale in production environments. In particular, CDX-based indexes are cumbersome to maintain and update for large collections. While it is possible to scale by exploiting more powerful individual servers backed by network-attached storage with large capacities, this is an expensive proposition. Running multiple Wayback instances over a partitioned collection is possible, but the software itself does not provide any guidance on how to integrate temporal browsing services with large-scale storage infrastructures. We believe that Warcbase provides a solution to these challenges: by using HBase to manage storage, we can scale out on commodity machines using proven infrastructure (HDFS), and temporal browsing can be supplied by a lightweight HBase client.

Browsing capabilities are only useful if one knows the exact URL of the desired content. Since this is often not the case, temporal search capabilities are the next most desired feature in web archives [7]. Unfortunately, most web archives do not support full-text search, with a few exceptions such as the Portuguese Web Archive [9], the British Library, and the Internet Archive's Archive-It service. There has been academic work on searching timestamped collections [18, 12, 3, 11, 22], but these systems have not been deployed in production at scale. Regardless, most previous work on full-text search in web archives has focused on technical issues such as the layout and organization of inverted index structures and index compression optimizations. Although important, such work is not sufficient, as we have little insight on archive-based search tasks and the types of interfaces that are needed to support them. While temporal search is an important problem, it is not the focus of our project. However, there has been work in integrating search into the Hadoop environment (e.g., Cloudera Search[4]) that we hope to leverage in the future.

Big data technologies such as Hadoop, HBase, and Pig have rapidly matured over the past few years. Hadoop has emerged as the *de facto* standard platform for large-scale data analytics, with widespread adoption in industry and substantial research interest from academia. HBase, in part due to its tight integration with the Hadoop ecosystem, has gained significant popularity for storing semi-structured data and for applications that require low-latency access. However, there are surprisingly few studies applying Hadoop and HBase to web archiving and, more generally, cultural preservation. A comprehensive review of big data technologies is beyond the scope of this paper, so here we focus specifically on archives, libraries, and other cultural heritage institutions. Rasheed [21] explored using the Hadoop Distributed File System (HDFS) as the storage backend for the Fedora Commons digital object repository system. The feasibility study confirmed that such a design was indeed possible and provided attractive properties, but Fedora Commons was not specifically designed for web archiving. Neudecker and Schlarb described Hadoop-based workflows at the Austrian and Dutch national libraries [17], but their study was exploratory in nature. While the Internet Archive uses Hadoop for certain processing and analytical tasks, the platform is not widely deployed within the organization. HBase has been used for assisting in the web crawling process, but deployment is at best limited.[5] Mignify [2] is a platform for storing web documents as well as extracted knowledge in HBase. It shares many of the goals of our project, but unlike our effort, is not open-source.

3. ARCHITECTURE AND DATA MODEL

This section describes the data model of our open-source Warcbase platform for web archiving. We present an application for temporal browsing and elaborate on the advantages of HBase in supporting scalable data processing via Hadoop MapReduce and management of knowledge extracted from raw web content. We conclude by describing a prototype interactive visualization for exploring web archives based on topic modeling.

HBase is best described as a sparse, persistent, multiple-dimensional sorted map.[6] An HBase table maintains a mapping from a 4-tuple to arbitrary values as follows:

(row key, column family, column qualifier, timestamp) → value

Conceptually, values are identified by rows and columns. A row is identified by its row key. Each column is decomposed into "family" and "qualifier", where the family provides a mechanism for the developer to specify groupings of qualifiers that should be physically co-located. The timestamp supports storage of multi-versioned values. Rows are lexicographically sorted, and thus an important element in HBase schema design is to leverage this property for the application's benefit. HBase provides a number of basic operations on the client end, including gets, puts, and range scans, along with so-called co-processors that allow processing to be pushed over to the server side. The consistency model guarantees single row transactions, but nothing more.

A table in HBase is divided into regions, which are ranges of consecutive rows. Each region is assigned to a RegionServer, which is responsible for handling operations on rows in the assigned regions. RegionServers coordinate assignment of regions via ZooKeeper [14], a highly-available replicated state machine and coordination service. HBase data

[3] `github.com/iipc/openwayback`

[4] `blog.cloudera.com/blog/2013/06/cloudera-search`

[5] Internet Archive, personal communication.

[6] Although HBase is derived from Google's Bigtable [5], in this paper we adopt the terminology of HBase.

are physically stored in HFiles, which reside on the Hadoop Distributed File System (HDFS). HDFS achieves durability and availability through replication, and HBase benefits from this design transparently. The entire stack was designed to be fault-tolerant and scalable, and in production has been demonstrated to scale to petabytes [1].

Since one of Bigtable's original use case was storing web crawls, schema design for Warcbase is relatively straightforward. In Warcbase, each collection is kept in a separate HBase table. We assume that each resource (e.g., web page, image, PDF, etc.) can be uniquely identified by its URL and a timestamp (the crawl date). Each resource's domain-reversed URL serves as its row key, i.e., `www.house.gov` becomes `gov.house.www`. This design allows different pages from the same domain to be physically kept together, thus allowing client requests to benefit from reference locality. The raw content of a resource (HTML, PDF, image, etc.) is stored as a value in the column family "c" with the MIME type (e.g., "text/html") as the qualifier. Each version of the crawl has a different timestamp. The use of the MIME type as the column qualifier saves us from a separate query, compared to an alternative design in which the MIME type is stored separately. The tradeoff, however, is that we no longer know the qualifier *a priori* when issuing a "get" to fetch a resource, and therefore must execute a (short) scan of the columns. This, however, does not have a material performance impact since in many cases we do not know the exact timestamp of a resource and need to scan through different versions to find the most appropriate one anyway.

As previously discussed, web crawling is beyond the scope of our work, as we assume that content has already been captured in existing ARC or WARC containers—we have built an ingest program that populates the appropriate HBase table according to this schema.

In addition to the raw content, HBase provides a flexible and extensible framework for storing extracted information alongside the raw content, thus allowing us to accumulate rich metadata and knowledge about the archived resources. Metadata are stored in separate column families: because qualifiers belonging to different column families are physically separated, operations on metadata will be more efficient since they will not require scanning the much larger column family holding the raw content. Within the metadata column family, extracted information such as language, plain text extracted from HTML pages, etc., are kept as values with different qualifiers. Finally, hyperlinks and anchor text are separately handled, since links are a pervasive and important part of the web. Following Chang et al. [5], we reserve a special column family "a". In it, source URLs serve as qualifiers, with the anchor texts as values.

3.1 Browsing Captured Web Content

Once a web crawl has been ingested into Warcbase, HBase handles the complexities of storage management in a scalable, fault-tolerant, and distributed manner. This includes maintaining the mapping from rows to RegionServers responsible for serving the records, splitting regions as more data are ingested, rebalancing region-to-RegionServer assignments, etc. When a RegionServer fails, HBase transparently handles failover to ensure availability. Underneath HBase, HDFS handles replication and other aspects of managing the raw data blocks. Because of this design, user-facing services can be implemented as HBase clients that encapsulate the custom application logic.

Figure 1: Screenshot of the Warcbase browser which provides access to archived content.

The Warcbase browser is one such application. It is written as a Jetty service that provides a webapp to the user. A standard webform allows the user to enter a URL and select a version of the content to examine. A screenshot is shown in Figure 1. Currently, we have been experimenting with crawls of the U.S. Congress in collaboration with the Library of Congress (more details later). On top of the rendered page a banner supplies metadata, including the URL of the content and the number of captured versions available. Clicking on a link takes the user to a contemporaneous version of the target page inside the archive.

The browser maintains a persistent connection to HBase and translates users' input into appropriate HBase queries. Retrieved raw content is transformed on the fly for correct page rendering, i.e., redirecting image links to content inside Warcbase and rewriting hyperlinks to point to a contemporaneous version inside the archive. Following the design of the Wayback Machine, all resources in the archive can be referenced with the following URL scheme:

```
congress108/20040124034300/http://www.house.gov/
```

which specifies the collection, the timestamp, and the URL requested. If a resource is not available with a matching timestamp, the closest version is returned.

The Warcbase browser mirrors the functionalities of the Wayback Machine (down to the URL schema) but has several advantages. The browser handles user interactions and page rendering, while offloading the storage management to HBase. This separation of concerns leads to a cleaner overall architecture and a much higher degree of scalability—the size of the web archive is limited only by the scalability of HBase itself, and since the browser service is stateless, we can load balance across multiple instances easily.

3.2 Scalable Analytics

Our use of HBase to store web archives provides two additional advantages. First, we gain access to tools in the Hadoop ecosystem for analytics and data processing. HBase seamlessly integrates with Hadoop MapReduce as both a data source and a data sink, which means that HBase rows

can serve as input key–value pairs to MapReduce jobs and that reducer output can be written back into HBase tables. HBase also integrates seamlessly with higher-level dataflow languages built on Hadoop and other open-source packages. Second, our data model provides a flexible and extensible framework for storing extracted information in HBase alongside the raw content, thus allowing us to accumulate rich metadata and knowledge about the archived resources.

In the simplest case, with Hadoop MapReduce we have a flexible scale-out execution framework to run analytical pipelines on web content in an embarrassingly-parallel manner. Components in a basic pipeline might perform webpage cleanup, boilerplate removal, language identification, and conversion of PDF and other file formats into plain text. A more advanced processing pipeline might apply natural language processing tools, starting with sentence chunking, part-of-speech tagging, named-entity recognition, and relation extraction. At each stage, intermediate data can be stored and managed in HBase, e.g., in a separate column family. Clients can easily access metadata and extracted knowledge for a wide variety of applications.

Beyond simple per-document processing, we have also implemented basic tools for extracting and manipulating webgraphs from archived content. We begin by first constructing a mapping from URLs to unique integer ids. Minimal finite-state transducers (FSTs) provide a compact and efficient way of encoding such a mapping (we use the implementation in the open-source Lucene search engine). With this mapping, we can then straightforwardly extract the webgraph contained in the archive and encode it using a compact integer-based adjacency list representation. From this, it is easy to run a variety of graph algorithms such as PageRank using well-known implementations [15]. One challenge of webgraphs in web archives is that the hyperlinks are often "temporally entangled". We need to make sure that a hyperlink does not to point to a future or past version of the page. Similarly, a web page should only receive an inbound link from a contemporaneous page. We solve the problem in a manner similar to Gomes et al. [9] by leveraging the fact that most web content is gathered in periodic (e.g., monthly) crawls, and thus we can process the webgraph from each crawl separately. We can easily determine where these "break points" should be with a Hadoop job to build a histogram of document timestamps.

In many cases, however, we do not believe that MapReduce is the best abstraction for data analytics and building analytical pipelines. Writing raw Hadoop MapReduce programs is slow and verbose. In industry, where data scientists are more interested in insights than algorithms, using a higher-level dataflow language has become the norm. One popular choice is Pig [20], which is a high-level dataflow language that "compiles down" to Hadoop jobs. Pig allows the user to manipulate large datasets in terms of relational operations such as group bys, joins, projections, etc. Along with many built-in primitives for manipulating atomic datatypes, Pig allows arbitrary code to be executed via custom user-defined functions (UDFs). This dataflow language allows users to perform complex analytical processing without writing Hadoop code in Java, potentially opening up the contents of web archives to users with more limited technical backgrounds. As a simple example, a Pig script to perform anchor text inversion, a standard webgraph processing task, takes only around half a dozen lines. Scripts to perform

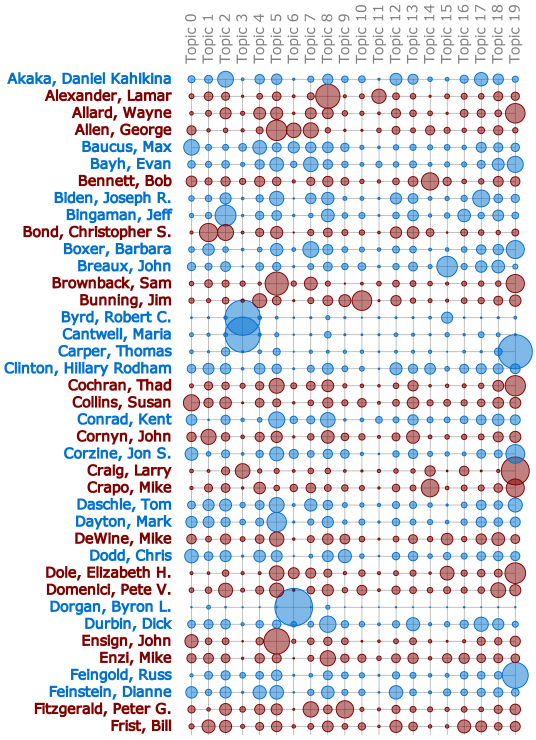

Figure 2: Interactive visualization of topic models.

temporal trend analysis, e.g., to generate graphs similar to the Google n-gram browser, are similarly succinct.

Finally, since HBase is well integrated into the Hadoop ecosystem, we gain access to a variety of open-source tools for complex data analysis. Examples include different processing frameworks such as Apache Giraph [16], specifically designed for iterative graph processing, and Spark [23], an alternative general-purpose data analytics engine capable of taking advantage of distributed memory. Additional examples include Apache Mahout, a popular package for machine learning at scale, and Mr.LDA [24], which provides a scalable MapReduce implementation of Latent Dirichlet Allocation (LDA) and related topic modeling techniques.

3.3 Exploration and Discovery

To illustrate how Warcbase enables applications that support exploration and discovery in web archives, we present an interactive visualization that takes advantage of topic modeling. The problem we are attempting to address is that users of an archive often don't know "where to start" and desire a tool that provides an overview of collection content. Latent Dirichlet Allocation (LDA) [4] is a popular technique for uncovering *latent* "topics" that are in the form of multinomial distributions over terms.

Currently, we view the collection as a sequence of temporal slices, where each slice corresponds to a monthly crawl. On each slice we run LDA, and the induced topic models are then visualized with a custom variant of Termite [6], as shown in Figure 2. The main visualization area displays a person-by-topic matrix, where the rows represent websites that are associated with U.S. senators and the columns represent the topics (due to space restrictions not all senators are shown). Names of Democrats are shown in blue and

Republicans in red; the sizes of the circles encode the prevalence of topics on that senator's website. Although there are quality issues due to the noisy nature of web pages, we can clearly identify topics that are part of the political discourse, e.g., topic 1 is about the environment, topic 2 is about Iraq, topics 9 and 10 are both about health care, topic 18 is about education, etc. Our interface provides drill-down capabilities whereby users can examine pages in which a topic is prominently expressed. We hope that such a tool provides a useful entry point for browsing archived content.

4. CURRENT STATUS

We have completed a working prototype of Warcbase in Java that includes components of all the functionalities described in this paper, including the core data model, the temporal browsing application, bindings for Pig support, simple analytical pipelines, and a simple interactive visualization based on topic models. Development and experimentation have been conducted on a Hadoop (YARN) and HBase cluster running Cloudera's Distribution of Hadoop (CDH) at the University of Maryland. The cluster comprises 16 compute nodes, each of which has two quad-core Xeon processors, 24GB RAM, and three 2TB disks.

Our current working dataset is a crawl of the 108th U.S. Congress gathered by the Library of Congress. It can be characterized as a "narrow but deep" crawl of the websites of members of the House of Representatives and the Senate from January 2003 to January 2005 at monthly intervals. The collection totals 1.15 TB of gzipped ARC files, containing approximately 29 million captures of 7.8 million unique URLs; of those, 23.8 million captures are HTML pages. We are exploring novel uses of this archive for applications in computational social science and digital humanities.

5. CONCLUSION

This paper presents Warcbase, an open-source platform for web archiving built on HBase and Hadoop. Since our system is built on proven "big data" technologies, it provides a scalable foundation for content analytics and applications that support exploration and discovery. We described an interactive visualization tool based on topic models, and possibilities abound for future innovative applications that help users derive value from web archives.

6. ACKNOWLEDGMENTS

This work has been supported by NSF under awards IIS-1144034 and IIS-1218043. Any opinions, findings, conclusions, or recommendations expressed are the authors' and do not necessarily reflect those of the sponsor. We thank the Library of Congress for providing web archive data and Andy Jackson for helpful comments. The first author is grateful to Esther for her loving support and dedicates this work to Joshua and Jacob.

7. REFERENCES

[1] A. Aiyer, M. Bautin, G. Chen, P. Khemani, K. Muthukkaruppan, K. Spiegelberg, L. Tang, and M. Vaidya. Storage infrastructure behind Facebook Messages: Using HBase at scale. *IEEE Data Engineering Bulletin*, 35(2):4–13, 2012.

[2] S. Barton. Mignify: A big data refinery built on HBase. *HBaseCon*, 2012.

[3] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. *SIGIR*, 2007.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[5] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A distributed storage system for structured data. *OSDI*, 2006.

[6] J. Chuang, C. Manning, and J. Heer. Termite: Visualization techniques for assessing textual topic models. *AVI*, 2012.

[7] M. Costa, D. Gomes, F. Couto, and M. Silva. A survey of web archive search architectures. *WWW Companion*, 2013.

[8] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *OSDI*, 2004.

[9] D. Gomes, D. Cruz, J. Miranda, M. Costa, and S. Fontes. Search the past with the Portuguese web archive. *WWW Companion*, 2013.

[10] D. Gomes, J. Miranda, and M. Costa. A survey on web archiving initiatives. *TPDL*, 2011.

[11] J. He, J. Zeng, and T. Suel. Improved index compression techniques for versioned document collections. *CIKM*, 2010.

[12] M. Herscovici, R. Lempel, and S. Yogev. Efficient indexing of versioned document sequences. *ECIR*, 2007.

[13] H. Hockx-Yu. Scholarly use of web archives, 2013.

[14] P. Hunt, M. Konar, F. Junqueira, and B. Reed. ZooKeeper: Wait-free coordination for Internet-scale systems. *USENIX*, 2010.

[15] J. Lin and C. Dyer. *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool Publishers, 2010.

[16] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A system for large-scale graph processing. *SIGMOD*, 2010.

[17] C. Neudecker and S. Schlarb. The elephant in the library: Integrating Hadoop. *Hadoop Summit Europe*, 2013.

[18] K. Nørvåg. Space-efficient support for temporal text indexing in a document archive context. *ECDL*, 2003.

[19] A. Ntoulas, J. Cho, and C. Olston. What's new on the web? the evolution of the web from a search engine perspective. *WWW*, 2004.

[20] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A not-so-foreign language for data processing. *SIGMOD*, 2008.

[21] M. Rasheed. Fedora Commons with Apache Hadoop: A research study. *code4lib Journal*, 22, 2013.

[22] S. Song. *Long-Term Information Preservation and Access*. PhD thesis, University of Maryland, 2010.

[23] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing. *NSDI*, 2012.

[24] K. Zhai, J. Boyd-Graber, N. Asadi, and M. Alkhouja. Mr. LDA: A flexible large scale topic modeling package using variational inference in MapReduce. *WWW*, 2012.