Tools and Services for Long-Term Preservation of Digital Archives

Joseph JaJa
Institute for Advanced Computer Studies
University of Maryland
College Park, MD. 20742
joseph@umiacs.umd.edu

Mike Smorul
Institute for Advanced Computer Studies
University of Maryland
College Park, MD. 20742
toaster@umiacs.umd.edu

Sangchul Song
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD. 20742
scsong@umd.edu

ABSTRACT

We have been working on a technology model to support the preservation and reliable access of long term digital archives. The model is built around a layered object architecture involving modular, extensible components that can gracefully adapt to the evolving technology, standards, and protocols. This has led to the development of methodologies, tools and services to handle a number of core requirements of long term digital archives. Specifically, we have built flexible tools for implementing general ingestion workflows, active monitoring and auditing of the archive's collections to ensure their long-term availability and integrity, storage organization and indexing to optimize access. These tools are platform and architecture independent, and have been tested using a wide variety of collections on heterogeneous computing platforms. In this paper, we will primarily focus on describing the underpinnings of our software called ACE (Auditing Control Environment), and report on its performance on a large scale distributed environment called Chronopolis. Built on top of rigorous cryptographic techniques, ACE provides a policy-driven, scalable environment to monitor and audit the archive's contents in a cost effective way. In addition, we will briefly introduce some our recent efforts to deal with storage organization and access of web archives.

Long term preservation is a process that must begin before an object is ingested into the archive and must remain active throughout the lifetime of the archive. The ACE tool provides a very flexible environment to actively monitor and audit the contents of a digital archive throughout its lifetime, so as to ensure the availability and integrity of the archive's holdings with extremely high probability. ACE is based on rigorous cryptographic techniques, and enables periodic auditing of the archive's holdings at the granularity and frequency set by the manager of the archive. The scheme is cost effective and very general, does not depend on the archive's architecture, and can detect any alterations, including alterations made by a malicious user. ACE can gracefully adapt to format migrations and changes to the archive's policies.

Proceedings of the Indo-USWorkshop on International Trends in Digital Preservation, March 24-25, 2009, Pune, India.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software; H.3.6 [**Information Storage and Retrieval**]: Library Automation; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries

General Terms

Algorithms, Management, Measurement, Performance, Design, Human Factors, Verification.

Keywords

Digital archiving, digital preservation, data integrity, web archiving, information discovery and access.

1. INTRODUCTION

The issue of long-term preservation of digital information has received considerable attention by major archiving communities, library organizations, government agencies, scientific communities, and individual researchers. Major challenges of long term digital preservation include the development of institutional and business models, technology infrastructure, and social and legal frameworks, which need to be addressed to achieve long-term reliable access of digital information. At the same time, a significant number of initiatives have been set up to develop technology prototypes to deal with various types of immediate needs and to address some of the long term challenges. These initiatives include the Internet Archive [2], the Electronic Records Archive program at the National Archives [1], the Library of Congress National Digital Information Infrastructure and Preservation (NDIIPP) [3], the DSpace project [13], the Fedora project [9, 12], the National Library of Australia's PANDORA project [4], LOCKSS [10], PORTICO [5], and the Transcontinental Persistent Archives Prototype (TPAP) [6].

Current proposed technical approaches developed through these prototypes provide functionalities such as tools for curation and ingestion, some approaches for dealing with format obsolescence, replication through a distributed infrastructure, and digital library type tools for access. In spite of these advances, systematic methodologies to deal with core infrastructure requirements are still lacking. These requirements include:

- Each preserved digital object should encapsulate information regarding content, structure, context, provenance, and access to enable the long term maintenance and lifecycle management of the digital object.
- Efficient management of technology evolution, both hardware and software, and the appropriate handling of technology obsolescence (for example, format obsolescence).
- Efficient risk management and disaster recovery mechanisms either from technology degradation and failure, or natural disasters such as fires, floods, and hurricanes, or human-induced operational errors, or security breaches.
- Efficient mechanisms to ensure the availability and integrity of content, context, and structure of archived information throughout the preservation period.
- Ability for information discovery and content access and presentation, with an automatic enforcement of authorization and IP rights, throughout the lifecycle of each object.
- Scalability in terms of ingestion rate, capacity and processing power to manage and preserve large scale heterogeneous collections of complex objects, and the speed at which users can discover and retrieve information.
- Ability to accommodate possible changes over time in organizational structures and stewardships, relocation, repurposing, and reclassification.

We have developed systematic technical approaches to deal with some of the core issues stated above, and recently released a number of related tools and services. Our overarching framework, called ADAPT (Approach to Digital Archiving and Preservation Technology), is a model based on a layered, digital object architecture that includes a set of modular tools and services built upon open standards and web technologies. These tools are designed so that they can easily accommodate new standards and policies while gracefully adapting to the underlying technologies as they evolve. Our tools include:

- *Producer – Archive Workflow Network (PAWN)* [14]. This is a mature software platform that is quite flexible in implementing distributed ingestion and processing workflows into a centralized or distributed archive. This software was developed in collabroation with the National Archives and Records Administration (NARA) to manage large scale distributed ingestion from widely distributed producers. PAWN presents a secure and reliable environment that enables distributed producers to easily prepare and submit their records to the archive, while the system automatically provides packaging and context, as well as, allow automated processing chains to be defined and executed on PAWN objects. PAWN has been tested extensively on the TPAP grid for ingesting and organizing a wide variety of records including NARA and SLAC records. Many groups have expressed interest in PAWN including the Library of Congress and several NDIIPP partners.
- *Auditing Control Environment (ACE)* [15]. We have developed a systematic method based on rigorous cryptographic techniques, which will actively monitor and audit the archive's contents to ensure the availability and integrity of each object in the archive. ACE Version 1.0 was released in October 2008 after extensive tests conducted through the Chronopolis project. More details about ACE and some of the experimental results will be presented later in this paper.
- *Space Organization and Indexing for Web Archiving.* The web has become the main medium for publishing information covering almost every facet of human activities but the web is an ephemeral medium whose contents are constantly changing. Web contents present unique challenges well beyond those encountered in typical archives due to their highly dynamic state, the underlying linking structures, and the widely heterogeneous types of contents. We have developed tools to organize and index web contents to be archived in such a way to enable compact space usage and fast exploration of archived web contents within a temporal context.
- *Information Discovery, Access and Presentation.* We have recently introduced methodologies for the effective exploration and information discovery of a web archive, including the ability to browse the contents within a temporal context, and to generate effective summarization of available temporal contents that satisfy the user's query. In particular, we have extended information retrieval techniques to include temporal contexts seamlessly into the web archiving architecture. Our focus has been on high level, temporal search, and on possible ways of presenting the matching results so as to enhance exploration of archived web contents.

2. BENCHMARKS AND TESTBEDS

We have used a wide variety of collections to test and validate our tools and services. These collections were managed by several archiving environments, including the following two distributed environments.

- *Transcontinental Persistent Archives Platform (TPAP).* This distributed storage environment is a collaboration between the San Diego Supercomputer Center (SDSC), the University of Maryland, and the National Archives. The main goal of this project is to demonstrate the use of data grid technology in support of digital archiving in a distributed environment, which includes the management of significant NARA-selected collections across the three main sites at SDSC, University of Maryland (UMD), and NARA. Over 5TB of data were managed by the TPAP environment, which was supported by a federation of the Storage Resource Broker (SRB) data grid middleware. PAWN was used on the TPAP environment for ingesting and organizing a wide variety of collections including NARA and SLAC records. Another tool developed by our group, Replication Monitoring Portal, was extensively used to audit the state of replication for various collections in TPAP.
- *The Chronopolis Environment.* Chronopolis is a geographically distributed archiving environment, built on top of substantial storage residing at SDSC, UMD, and the National Center for Atmospheric Research (NCAR), which is actively managed by preservation tools and services.

Chronopolis is currently managing substantial collections from the NDIIPP partners, including the California Digital Library (CDL) Web-at-Risk collection, the InterUniversity Consortium for Political and Social Research (ICPSR) collection, and collections from the Scripps Institution of Oceanography – Geological Data Center (SI-GDC), and North Carolina Geospatial Data Archiving Project (NC State). Chronopolis has been using ACE to monitor and audit these collections for almost a year.

3. ACE – ACTIVE MONITORING AND AUDITING OF DIGITAL COLLECTIONS

3.1 Background

Digital information is in general very fragile and can easily be altered resulting in changes that are very hard to detect in general. There are many potential risks that range from hardware and software failures to major technology changes rendering current software and hardware unusable, to the every growing number of computer security breaches. Note also that most of an archive's holdings may be accessed very infrequently, and hence several cycles of technology evolution may occur in between accesses, thereby causing corrupted files to go undetected until it is too late to fix the problem. Other potential long-term risks include operational errors (mishandling of the archive's holdings by a staff member) and natural hazards and disasters such as fires and floods. Another significant risk, especially in the long term, is malicious alterations performed either by users internal or external to the archive. Malicious alterations are the hardest errors to detect. In fact, most of these risks may cause unnoticeable changes to the archive, which may last for a long time before they are detected. Two additional factors complicate matters further. First, a number of transformations may be applied to a digital object during its lifetime. For example, format obsolescence can lead to a a migration of the current holdings in the obsolescent format to a new format. Therefore information stored about the old version of the object needs to be updated appropriately. Second, cryptographic techniques, used by all current integrity checking mechanisms, are likely to become less immune to potential attacks over time, and hence they will need to be replaced by stronger techniques. Therefore these two problems need to be also addressed in any approach to ensure the integrity of a digital archive.

3.2 Core Technical Principles

All known approaches to manage the integrity of a digital archive revolve around the following three techniques.

- A majority voting scheme using replicated copies of the object or their hashes.
- Build a directory containing the hash values of all the archived objects. To audit an object, compute its hash value and compare it to the stored value in the directory.
- Using a Public Key Infrastructure (PKI), create a digital signature of the object and save it in the archive. The auditing process involves the use of the public key of the archive or a third party depending on the scheme used.

As we have argued in [15], each of these schemes has significant limitations that render them unsuitable for long term archives.

Before describing our approach, we informally introduce the notion of a cryptographic hash function. Essentially such a function takes an arbitrarily long bit string and apply a certain mathematical function that generates a fixed length bit string called the hash value or digest of the object. The main characteristic of the function is that it is easy to apply, but it is computationally infeasible to determine an input string for a given hash value. Another related property is the following fact. Given an input string, it is computationally infeasible to construct another different string with the same hash value. A complication about using hashing functions is that there are no known functions that can be shown to satisfy the properties stated above. However there are several hash functions (MD5, SHA-1, SHA-256, and RIPEMD-160) that have been in widespread use even though we cannot prove the computational infeasibility mentioned above.

The starting point of our approach is the scheme based on the storage of the hash values of all the objects in a directory. Hence the auditing of an object consists of computing the hash value of the object followed by a comparison with the stored value. This scheme will work correctly as long as the stored hash values are kept intact, and the hashing scheme used remains secure. This approach is not strong enough to satisfactorily manage the integrity of long-term digital archives for the following reasons. First, a malicious user can easily modify the object and its stored hash value since the hash function is known, in which case the scheme will not be able to detect the alterations made to the object. Second, we note the fact that the number of hash values grow linearly with the number of archived objects, and hence ensuring the integrity of all the hash values amounts to a non-trivial problem that is, in many ways, very similar to the initial problem we are trying to solve.

Our approach reduces the number of hash values to a very small set of hash values called "witnesses," essentially one per day which amounts to a total size of around 100KB per year regardless of the number of objects stored in the archive. The correctness of our scheme depends only on maintaining the integrity of the witnesses. Any alteration to an object, whether malicious or otherwise, will be detected using one of the witness values. Given the small-size of the witness values, they can be easily saved on read-only media such as papers or archival quality read-only optical media (with periodic refreshing of the contents). To make this approach practical, we need an easy way to link a digital object to the corresponding witness value. We provide the details next.

We organize the processing of the objects to be monitored and audited into rounds, each round defined by a temporal index. During each round, the digests of all the objects being processed are aggregated into a single hash value, called the *round summary value*. There are many possible schemes that can be used to perform the aggregation, the simplest of which is to process the digests in sequence, concatenating the previous hash value with the running round summary value, and applying the hash function to the concatenated string to update the round summary value. However in our method, we make use of the Merkle tree [11], which works as follows (see Figure 1). The digests of all the objects being processed in a round form the leaves of a balanced binary tree such as the value stored at each internal node is the hash value of the concatenated values stored at the children. The value computed at the root of the tree determines the round hash

value. Given the properties of a hash function, a change to any of the objects being processed during a round will immediately result in a different round hash value with extremely high probability.

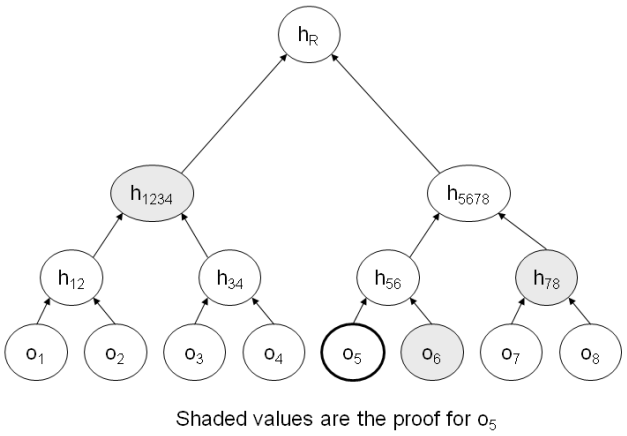

Figure 1. Merkle Tree

We now introduce the notion of the *proof* of the digest of an object. Given an object that is represented by a leaf of the Merkle tree, the proof is the sequence of the hash values of the siblings of all the nodes on the unique path from the leaf to the root. Note that the number of hash values defining the proof is logarithmic in the total number of objects processed during a round, and hence will be quite small in general. In our scheme, each round is restricted to a maximum time interval, and will be indexed by a time stamp. Hence the total number of round hash values depends on the total number of active rounds and not on the total number of objects. To track the correspondence between an object and the corresponding round hash value, we create an integrity token for each object, which contains the corresponding round proof and its temporal index (time stamp). An alteration of the object can be detected by using its integrity token to compute the corresponding round hash value, and comparing it with the stored round hash value.

We re-iterate the process by aggregating the sequence of round hash values generated each day using the Merkle tree scheme. The root hash value defines the witness for that day. Any alterations in the object will result with very high probability to a different witness value.

In summary, our scheme is based on processing the objects, in temporarily-indexed rounds to compute the round summary values, and then combine all these values generated during a single day into one hash value, called the witness for that day. An alteration to an object can be detected by using its integrity token to compute the corresponding round hash value, followed by computing the corresponding witness value using the proof of the round hash value. The resulting witness value will be different with extremely high probability than the stored witness value, if the original object has been modified.

This approach, coupled with a strategy to proactively monitor and audit the holdings of a digital archive, was implemented in a software system called ACE. We will provide an overview of the ACE software architecture, and outline its basic functionality in the remainder of this section

3.3 Basic ACE Architecture

The basic ACE architecture consists of two main components, a local component called Audit Manager (AM) that can be configured at the archive to monitor and audit local data, and a third-party component called Integrity Management Component (IMC) that can generate integrity tokens and round hash values upon requests from the archive. An IMS can support multiple archives, including distributed archives. The IMS not only generates the integrity tokens, but also maintains a database of the round hash values, as well as generating the daily witness values. The IMS is an independent entity whose function is to handle requests from an archive, which typical includes the hash of the object but not the object itself or a time stamp. The IMS is very simple to implement, and is not assumed to be fully trusted (and hence can itself be audited if necessary). In ACE, each integrity token contains several pieces of information in addition to the proof and the time stamp, such as the ID of the hash algorithm used by the IMS (which could be different than the hash function used by the archive), and the last time the object was audited. The AM forms a bridging component with the IMS, whose function is to send requests to the IMS to generate integrity tokens for a number of objects, and requests for the round hash value corresponding to a particular time stamp. Figure 2 shows the overall ACE architecture of the general case of different archives being supported by a single IMS.

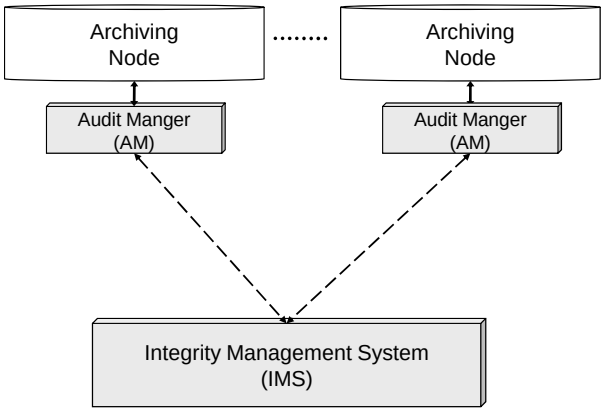

Figure 2. ACE Architecture

3.4 ACE Workflow

In this section, we discuss a typical workflow with ACE, which includes two main operations: registration and auditing as described next.

3.4.1 Registration of Objects into ACE

For each object to be registered into ACE, the audit manager creates a registration request and submits it to the IMS. The IMS aggregates all the requests for the round and returns the integrity token of the object to the requesting AM. In the meantime, the IMS runs a continuous series of aggregation rounds, each round closing either when the round receives the maximum number of registration requests, or when the maximum amount of time allocated for a round is reached, whichever comes first. These parameters are assigned by the IMS administrator. However this round interval policy can be overridden through a special object registration request, which forces the current run to immediately

close and return an integrity token. Object registration requests received during a round are aggregated together along with a number of random values through the Merkle tree. The random values are added as necessary to ensure a minimum number of participants in a round. The resulting round hash value is managed internally within the IMS, and the integrity token is issued to each AM who participated in that round (that is, the audit managers who sent requests during the round).

The IMS constructs daily a witness value using the hash round values of the day, and sends the witness value to all the participating archives as well as store the value on a reliable read-only medium. These witness values are cryptographically dependent on the round hash values, similar to the dependence of each round hash value on the digests of the objects participating in the same round. Copies of the witness values are stored on reliable read-only media at the various archives.

3.4.2 The ACE Auditing Process

ACE Version 1.0 can be configured to monitor and audit the archive's holdings by the manager of the archive. The periodic auditing policy can be set either at the object level or collection level. For example, a policy for a certain collection could involve auditing all the objects in the collection every three months, while the policy set for another collection could require auditing all the objects in that collection every six months. A default policy (audit every six months) will be set during registration time unless the archive manager overrides the default setting. In addition, the auditing process could be invoked by the manager or authorized users at any time on any object.

The process of auditing an object *O* consists of the following steps.

1. The audit manager computes the hash value of *O* and retrieves its integrity token.
2. Using the computed hash value of *O* and the proof contained in the integrity token, the audit manager computes the round hash value.
3. Using the round time stamp contained in the integrity token, the audit manager requests the corresponding round hash value from the IMS.
4. The audit manager successfully terminates the auditing process if the computed hash value in Step 1. is equal to the hash value stored in the integrity token, and the two round hash values computed in Steps 2 and 3 are equal. Otherwise, it sends an error alert to the archive manager.

Note that the successful termination of the auditing process establishes the correctness of the object and its integrity token with extremely high probability given the properties of the hash functions. However occasionally we may want to apply a stronger auditing process, independent of the trustworthiness of the archive and the IMS.

An independent auditor can request the proof of the round hash value from the archive and the proof of the corresponding daily witness value from the IMS. Using these two values, the independent auditor can easily compute the value of the witness value, which is then compared to the stored value on the reliable read-only media. If the two values are equal, then the object is intact and both the archive and the IMS can be trusted. A failure of this process will automatically invalidate the object under consideration

3.5 Audit Manager

The ACE Audit Manager is a web application designed to be installed locally at an archive. It provides an easy-to-use portal that will provide an overview of the collections registered into ACE, manage all the ACE audit operations, and provide detailed logging of the status of all monitored items. In addition, it has been designed to monitor multiple collections across different types of storage. A screenshot of an Audit Manager installed on the Chronopolis environment is shown in Figure 3.

Figure 3. Audit Manager Screenshot

3.6 Audit Manager Performance

ACE Version 1.0 has been used extensively on the TPAP and the Chronopolis testbeds. We focus here on the performance achieved within the Chronopolis environment. ACE has been deployed on that environment for nine months, working almost flawlessly during that period. The current default auditing policy is to audit files at the University of Maryland every 30 days. Table 1 illustrates the performance of a single audit manager on the collections audited at the University of Maryland, amounting to approximately 6 million files. A large fraction of the time is the result of the Storage Resource Broker overhead, and the overall time is dominated by the time to access the collections across the network.

Table 1. Audit Manager Performance

Collection	No. of Files	Size (GB)	Audit Time (hh:mm)
CDL	46,762	4,291	20:32
SIO-GDC	197,718	815	6:49
ICPSR	4,830,625	6,957	122:48
NC State	608,424	5,465	32:14

During the auditing of the CDL collection, a single audit manager was able to run at the rate of about 60 MB per second on average, almost fully utilizing the file transfer bandwidth available. For the other collections, where there were many small files, the audit speed was further limited by the overhead accessing each file. For example, on the ICPSR collection, the audit manager ran at the rate of 13 MB per second, having to open up each of about 4.8 million files. These results indicate that the actual time spent by an audit manager to perform the core auditing process is quite small – in fact so small as to be effectively hidden by the unavoidable overhead for accessing the collections. We note that multiple audit managers can be run concurrently on different collections to increase the performance almost linearly as long as the network bandwidth can support the simultaneous access to the collections.

A benchmark driver has been developed to help eliminate bottlenecks in the Audit Manager. Two tests were performed; first a collection containing 1.25 million digests was simulated and registered in ACE. Digests were not actually generated, but rather simulated. ACE still requested tokens, performed logging, and registered items as if it were a real collection. Registration took 3 hours, 6 minutes with a sustained rate of 112 objects per second being registered. An audit of the registered collection was completed in 1hour, 17m showing an audit rate of 270.6 objects per second.

The test was re-run with 1.25 million 1MB simulated files to see how the digest algorithm would affect ACE performance. For this test, computing the digest and registration were performed serially; for all production ACE drivers, computing the digest is performed in parallel with registration. Registration took 6 hours, 7 m for a sustained rate of 56.8 objects/s and 56MB/s. A subsequent audit was performed in 4 hours, 30 minutes for a combined rate of 77.2 objects/s and 77.2MB/s. Subtracting the ACE overhead, computing the digests was achieved at the rate of a little over 110MB/s.

While there is certainly room for further optimization of the ACE audit manager, comparing the benchmarks with throughputs from the Chronopolis archive shows that even for the fastest collection, the Audit Manager was bottlenecked by the underlying storage system.

4. WEB ARCHIVING

A recent major thrust of our group has been on the storage organization and indexing of web objects for fast information retrieval and access to web archives. The web has become the main medium that holds information regarding almost every facet of human activities worldwide. However the web is an ephemeral medium whose contents are constantly disappearing, sometimes shortly after they have appeared. Moreover, the web is the only medium where a substantial amount of important information is being published, and hence unless we actively archive the critical information, we may suffer a permanent loss of part of our cultural and scientific heritage on a regular basis.

Our work has dealt with some of the technical challenges facing the organization, preservation, and access of web archives. Web contents present unique challenges well beyond those encountered in other types of archives. To start with, a web object is usually not well-delineated because of all the links that it contains, which makes it hard to determine the boundaries of a web object. Add to that the fact that many web objects contain highly dynamic contents that change at unpredictable rate, and the fact that a large fraction of the web contents reside in the deep or hidden web, and hence cannot be extracted through the typical web crawling process.

4.1 Storage Organization and Indexing for Fast Access

The most popular method currently in use by most web archives, including the Internet Archive, stores a multitude of small web objects into relatively large containers. An emerging standard for such containers is the WARC format [7]. Typically, an external indexing server is maintained to provide the mapping between hyperlinks inside a WARC file and the location of the archived object that the hyperlinks point to. In [18], we addressed the problem of storing and indexing web contents using the crawling strategy but avoiding the storage of any duplicate web contents examined between two consecutive crawls. We developed a new scheme that achieves precisely this goal while ensuring quick access to the archived contents based on a temporal query covering any period during the archival time span. The scheme, called PISA – Persistent Indexing Structure for Archives - involves a novel indexing structure based on the concept of multi-version B-tree and a very efficient duplicate detection algorithm.

To illustrate the benefits of our scheme, we ran two types of experiments. The first type dealt with the impact of the duplicate elimination, and the other type dealt with the performance of our scheme versus that of the standard B+-Tree. For the first type of tests, we chose two datasets from the Stanford's WebBase project – a news web archive, and a governors' web archive to represent respectively fast-changing websites, and relatively static websites. For the second set of tests, we used the governor's web archive that happened to have more crawl dates (thus more versions) than the news web archive, and hence presented an opportunity to illustrate performance on temporal queries.

The results of the first set of tests showed that 82% of the archived web pages in the governors' web archive were duplicates. We, therefore, could save about 73% of storage space with our scheme. For the fast-changing news web archive, we could still see 23% of the archived web pages were duplicates, which accounted for 12% in storage space.

The second set of tests separately indexed the governors' web archive using B+-Tree, and PISA, and ran several access operations (inserts, and several temporal and non-temporal queries) on each index. The result showed a substantial performance gain for some important temporal operations. For example, we saw as much as ten times faster performance for time-slice queries ("get all web pages that were valid at a certain time point"). We are currently in the process of testing this

scheme on several terabytes of web archive data in collaboration with the Internet Archive and the Library of Congress.

4.2 Efficient Exploration of Archived Web Contents

The conventional way of constructing WARC containers does not take into consideration the linking relationships among the web objects stored across multiple containers. This conventional way is to store the web objects as they are crawled into a container until the container is full. Although this method is simple, it can result in performance bottlenecks when users navigate through the archived web objects that are stored in many different containers.

In [16], we addressed the problem of how to organize the web objects across multiple containers so that we will be able to navigate through the linking structure of the web objects as efficiently as possible. In particular, we considered a web graph where each constituent web page is represented by a vertex, and each incoming/outgoing link corresponds to a directed edge. Given a web graph, we use a variant of the PageRank algorithm [8] to assign each edge a weight according to its chance to be taken in a browsing session. The weighted web graph is then partitioned in such a way to minimize the amount of overhead required to navigate through the archived pages.

In order to evaluate our approach, we ran empirical tests on two datasets. The first is the web graph of the University of Maryland Institute for Advanced Computer Studies (UMIACS) web site, located at http://umiacs.umd.edu domain. We crawled every Web page within a five-hop distance (or depth) under this domain, and constructed the web graph corresponding to this crawling. The second dataset is the Stanford web graph which was generated from a crawl of the stanford.edu domain created by the Stanford WebBase project. Over each data set, we ran 1000 random walk sessions and counted the number of containers each session needed to retrieve.

In the tests, we observed that when the graph partitioning scheme is used, most random walks only needed to access very few containers. Our method reduced the average number of containers from five to one for the UMIACS web graph, and seven to four for the Stanford web graph.

4.3 Information Discovery, Retrieval and Presentation of Archived Web Contents

The holdings of a web archive are highly temporal since each archived object is only valid for a fixed time span. Therefore, web archives should be organized to support search and information exploration within a temporal context. In this ongoing work [17], we consider the mechanisms needed to enable information exploration, search, and access of archived web contents within a temporal context while effectively handling the highly unstructured, complex contents and their linking structures, all typically at very large scale.

In developing our strategy, we consider some of the most notable approaches currently adopted in existing web archives, and identify their limitations. The main approach used by The Wayback Machine (Internet Archive), is to list archived web pages chronologically based on a specific URL provided by the user. However, requiring prior knowledge of the URL or even specific information about the contents of a web site, severely limits the future use of the archive. Note that even if the URL is currently well-known, this specific URL may be completely forgotten in the future.

Another approach categorizes contents into a hierarchy through which users can navigate down until the desired object is found. Some web archives, such as the Minerva project of the Library of Congress, have taken this approach for some of its collections, such as the United States presidential election of 2000 that has about 800 sites archived daily between August 1, 2000 and January 21, 2001. However, as the size of the collection becomes enormous, the categorization becomes almost impossible and not effective.

The last, and the most promising, approach provides full-text search capability, based on an extension of information retrieval techniques. An example can be found in the open source archiving project WERA. Although more realistic and effective, current information retrieval strategies do not take into account temporal contexts, nor do they appropriately handle the evolution of the importance of a web page over time. For example, they fail to consider the fact that a page that contains “September 11” can be totally irrelevant to the September 11 Attack if the page was only valid before 2001.

In this ongoing work, we are developing methodologies for the effective exploration and information discovery of a web archive, including the ability to browse the contents within a temporal context, and to generate effective summarization of available temporal contents that satisfy the user’s query. In particular, we extend information retrieval techniques to include temporal contexts seamlessly into the architecture. In [17], we present initial results using high level, temporal search, as well as, include possible ways of presenting the matching results so as to enhance exploration of archived web contents.

5. CONCLUSION

In this paper, we have provided an overview of our overall technology model to support long term digital archives. Of particular importance to us has been the development of sound technical approaches to ensure the integrity of and reliable access to the holdings of a digital archive. The realization of these technical approaches has been based on the development of configurable, extensible, modular components, which can adapt gracefully in a cost effective way as newer technologies, standards, and protocols emerge. Given the critical importance to proactively monitor the archive’s holdings, we have presented the underpinnings of ACE – a cost effective software system that can ensure the integrity of long term archives – and which has been tested and validated in a large scale distributed environment. We also provided of some of our current efforts to deal with organization and access of archives dealing with web contents. Given the dominant role of the web as the publication medium of information related to almost all major human activities, web archives will become the major source of historical information in the future. Our focus there is in enabling information discovery and mining for a large scale web archive within a temporal context.

6. ACKNOWLEDGMENTS

Our thanks to the Library of Congress under the NDIIPP program, the National Archives and Records Administration under the Electronics Record Archive program, the National Science

Foundation, the Internet Archive, the Mellon Foundation, and the University of Maryland for partially supporting several components of our research program as articulated in this paper.

7. REFERENCES

[1] *The Electronic Records Archive (ERA), the National Archives and Records Administration*. URL: http://www.archives.gov/era. Accessed:2008-06-09 (Archived at: http://www.webcitation.org/5YSZCfoK5).

[2] *The Internet Archive: The Wayback Machine*. 2008 URL: http://www.archive.org/index.php. Accessed:December 11 2008 (Archived at: http://www.webcitation.org/5SCSL2r8e).

[3] *The National Digital Information Infrastructure and Preservation Program, the Library of Congress*. URL: http://www.digitalpreservation.gov/. Accessed:June 9 2008 (Archived at: http://www.webcitation.org/5YSZBCgxW).

[4] *Pandora - Austrailia's Web Archive*. URL: http://pandora.nla.gov.au/. Accessed:April 22 2008 (Archived at: http://www.webcitation.org/5XHOp9Kso).

[5] *PORTICO*. URL: http://www.portico.org/. Accessed:Feb 10 2009.

[6] *The Transcontinental Persistent Archives Prototype (TPAP)*. URL: http://www.archives.gov/era/research/tpap.html. Accessed:2009-02-13 (Archived at: http://www.webcitation.org/5eYpbiKsy).

[7] *WARC, Web ARChive file format*. URL: http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml. Accessed:December 15 2008 (Archived at: http://www.webcitation.org/5d5ZyngyG).

[8] Brin, S. and L. Page. *The anatomy of a large-scale hypertextual Web search engine*. in *Proceedings of Proceedings of the Seventh International Conference on World Wide Web 7*. 1998. Brisbane, Australia: Elsevier Science Publishers B. V.

[9] Lagoze, C., et al., *Fedora: an architecture for complex objects and their relationships.* International Journal on Digital Libraries, 2006. **6**(2): p. 124-138.

[10] Maniatis, P., et al., *The LOCKSS peer-to-peer digital preservation system.* ACM Trans. Comput. Syst., 2005. **23**(1): p. 2-50.

[11] Merkle, R.C. *Protocols for Public Key Cryptosystems*. in *Proceedings of IEEE Symposium on Security and Privacy*. 1980.

[12] Payette, S. and C. Lagoze. *Flexible and Extensible Digital Object and Repository Architecture (FEDORA)*. in *Proceedings of ECDL '98*. 1998. Heraklion, Crete, Greece: Springer-Verlag.

[13] Smith, M., et al., *DSpace: An Open Source Dynamic Digital Repository*, in *D-Lib Magazine*. 2003.

[14] Smorul, M., M. McGann, and J. JaJa. *PAWN: A Policy-Driven Environment for Implementing Producer-Archive Interactions in Support of Long Term Digital Preservation*. in *Proceedings of Archiving 2007*. 2007: IS&T.

[15] Song, S. and J. JaJa. *ACE: A Novel Software Platform to Ensure the Integrity of Long Term Archives*. in *Proceedings of Archiving 2007*. 2007: IS&T.

[16] Song, S. and J. JaJa. *Fast Browsing of Archived Web Contents*. in *Proceedings of the 8th International Web Archiving Workshop (IWAW 2008)*. 2008. Aarhus, Denmark.

[17] Song, S. and J. JaJa. *Search and Access Strategies for Web Archives*. in *Proceedings of Archiving 2009 (to appear)*. 2009. Arlington, VA: IS&T.

[18] Song, S. and J. JaJa, *Web Archiving: Organizing Web Objects into Web Containers to Optimize Access (UMIACS Technical Report No. UMIACS-TR-2007-42)*. 2007, University of Maryland Institute for Advanced Computer Studies: College Park, MD.