Archiving, Indexing and Accessing Web Materials: Solutions for large amounts of data

David Minor[1], Reagan Moore[2], Bing Zhu[3], Charles Cowart[4]

San Diego Supercomputer Center
University of California: San Diego
9500 Gilman Drive, La Jolla, CA 92023

1. (858)534-5104 minor@sdsc.edu
2. (858)534-5073 moore@sdsc.edu
3. (858)534-8373 bzhu@sdsc.edu
4. (858)534-5065 charliec@sdsc.edu

ABSTRACT

The archiving of Internet materials presents two major challenges: the dynamic nature of the content and the massive number of individual web pages. These two characteristics impact the choice of methods for storing, indexing and providing fast access to archives of materials retrieved from periodic web crawling activities. At the San Diego Supercomputer Center, we have applied two different technologies for archiving, indexing and accessing web archived materials for two projects: a Library of Congress / San Diego Supercomputer Center joint storage project (LOC/SDSC) and the National Science Digital Library (NSDL) persistent archive. In the LOC/SDSC project, we deployed the Wayback software for indexing and accessing a large archive collection from the Library of Congress. Our experience demonstrates that the Wayback software (release 0.6.0) is mature and suitable for handling small collections. Our modifications to the Wayback machine enabled the management of distributed collections. In the NSDL project, we used the Storage Resource Broker (SRB) data grid to index and archive web materials. This approach enabled the use of distributed and heterogeneous storage systems, while supporting access from multiple types of clients, ranging from web browsers, to workflow systems, to Perl and Python load libraries.

Categories and Subject Descriptors

H.3.7 [**Digital Libraries**]: Web Digital Object Preservation – *Indexing of Web materials, access of archived web content, Scalable solution, data intensive computing.*

General Terms

Management, Design, Reliability, Verification.

Keywords

Web Archive, Large Index, Distributed Storage, Wayback, Digital Library, Digital Preservation, Storage Resource Broker.

1. INTRODUCTION

Currently, there are many open source web crawling technologies that retrieve web pages. Examples include Heritrix, Wget, HTTrack and WIRE [1]. Of these only Heritrix addresses the real issues of archiving, indexing and accessing large collections of web crawled materials for long term digital preservation. Most of the crawlers store the crawled files as original files, for example, and are targeted for small collections for offline viewing. This model is obviously not suitable for handling large collections of web archive data, like those from the LOC/SDSC and NSDL [2] projects for long term web digital preservation. These projects encounter multiple challenges in the efficient archiving, indexing and fast access or retrieval of web pages from the massive amount of archival materials. For example, WIRE suggests kernel tuning in a machine to maximize the capacity of i-nodes in its software implementation [3]. When tens of millions of web pages exist within a collection, aggregation mechanisms are required to avoid over-loading the name space of the storage system. Other challenges are the scalability of the solution when responding to user requests and the ability to use distributed and heterogeneous storage systems for archiving the web pages.

The Library of Congress and SDSC have been working with web archive materials as part of a 1-Year grant project funded by the LOC. The focus of the project is to demonstrate the feasibility and performance of current approaches for a production digital Data Center to support the Library of Congress' requirements. As part of the project SDSC is hosting approximately 6 terabytes of web crawl data amassed by the LOC as part of their Web Capture Program. SDSC has taken this data and provided indexing and access services for it.

The National Science Digital Library manages a central repository of URLs that point to educational material for all science disciplines for all grade levels. The URLs are indexed using education extensions to the Dublin Core metadata, and multiple access services are provided. The San Diego Supercomputer Center maintains an archive of the material present at each URL. Periodic web crawls are made of the registered URLs. The crawled materials are deposited into the Storage Resource Broker data grid [4] and archived on both disk and tape storage systems. An Open Archives Initiative interface is provided for interacting with the archive. Links to the archived web pages are provided from the NSDL portal residing at Cornell University.

The experience in maintaining the archive points out the need for preservation of web pages. Originally, the retrieval error rate (file not found, server offline) was about 1.5%. After four years, the retrieval error rate is as high as 25% when accessing web material which has had no funding support for two years. Thus a significant amount of material is preserved in the NSDL archive which is no longer accessible from the original web site.

The NSDL web archive manages over 50 million web pages, representing a sequence of periodic crawls of the registered URLs. Challenges in managing the archive include access statistics, aggregation of web pages in containers before deposition onto a storage system, removal of duplicate pages, and tracking of retrieval error statistics.

2.1 Indexing LOC collection

The LOC/SDSC project deployed the Wayback software [5] to index and provide web access service for a collection with 48,369 ARC files from the Library of Congress. With an average file size of 128MB, the size of the entire collection is about 5.2 Terabytes. Each ARC file held about 1000 web pages.

Developed by the Internet Archive team, the Wayback software is used for indexing and replaying Internet materials archived using a special file format, called the ARC format. ARC files contain multiple web pages and are created automatically by some web crawlers, such as Heritrix [6], as illustrated in Fig 1. Web pages archived from the Internet typically contain large numbers of small files. Thus a method to save aggregated files into a single file like a UNIX tar file is an efficient way to store web crawls. However, the UNIX tar format itself is not sufficient to support Internet archives since it doesn't have a sophisticated metadata model that can be used for describing and indexing the archived materials. In the ARC format, many small-sized web files are compressed and glued into one file along with their indices, which are used for searching and replaying the archived materials. Typical metadata (index data) for ARC files are tuples of values for:

URL: the original web address for the page or material.

Capture Date: the date the web crawl was carried.

origHost: original web host.

mimeType: MIME type for the document.

httpResponseCode: HTTP protocol response code

md5Fragment: unique MD5 generated by crawler.

redirectUrl: a URL for re-direction.

compressedOffset: the offset value in number of bytes for an archived web page or other material.

arcFileName: original ARC file name.

This set of index data is uniquely determined by the URL and Capture Date. In database language, the schema has a primary key of *(URL, Capture Date)*. For each ARC file, the index data can be generated by a Wayback's Java command-line utility tool, org.archive.wayback.cdx.indexer.ArcIndexer, or by a Java class 'org.archive.io.arc.ARCReader' from the Heritrix software package.

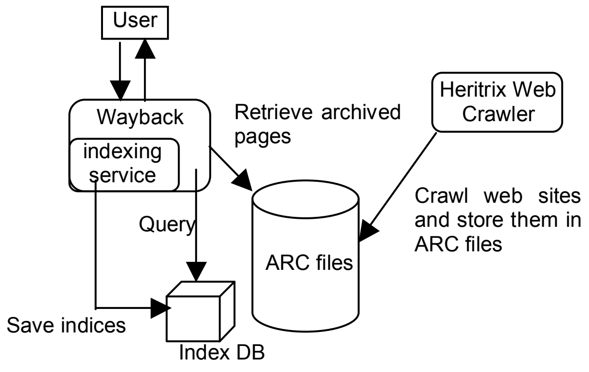


Fig 1. A Schematic Diagram of Wayback Software

Wayback has a background indexing process to check periodically if there are new ARC files in a specified disk repository and will index newly arrived ARC files. We observed that the indexing rate that can be sustained by a single Wayback instance is about 1000 files per day. This implies that we would need about 42 days of constant computing to index the entire collection. This was not acceptable and there was a need to run several indexing processes in parallel to speed up the indexing process when dealing with such a large collection of data. With Wayback 0.6.0, we chose to run 18 Wayback instances to index the LOC 2004 ARC collection in parallel. The indexing of the entire collection was finished in a week. The total size of the Berkeley DB database files is around 100 Gigabytes.

In Wayback's recent release 0.7.0 some command-line tools are provided to create text-based index files, which then can be merged and used as a file-based index. With these tools, indexing a large collection in parallel is possible by running multiple index processes creating multiple text-based index files.

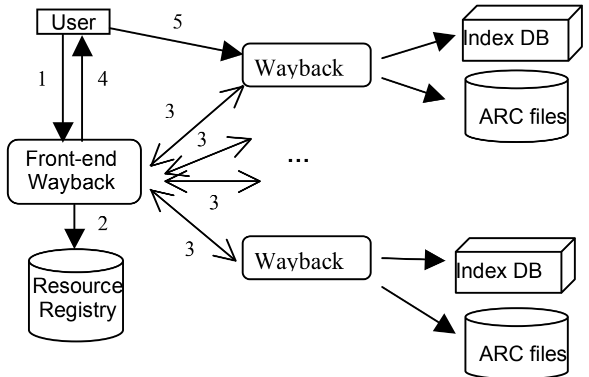
2.2 Building Distributed Wayback Machines

Even with the enhanced tools in Wayback 0.7.0, two issues still exist:

(1) Scalability of either a single Berkeley DB database or text – based indexing files as the amount of indexing information gets bigger and bigger.
(2) Access to multiple ARC files that are distributed across multiple storage systems.

Once we had solved the problem of indexing by running 18 Wayback instances, we needed a solution to re-integrate the multiple Wayback instances for the purposes of display. Built on top of the Wayback software release, a master Wayback was developed that virtualizes other Wayback instances as sub-collections. Each Wayback instance is registered as a Wayback resource in the Master Wayback. When a user sends a web page query to the Master Wayback, it distributes the query to the registered instances and merges the query results for the user. Since we indexed the LOC ARC materials using 18 Wayback instances, we ran 18 Wayback instances as backend subcollections and deployed a single Master Wayback as the front-end for accessing the LOC 2004 collection. The approach was an immediate success. But extensive performance tests showed the approach had a performance problem, especially for browsers like Firefox and Opera. The Wayback query service for page retrieval also retrieves files (such as images) linked on pages inside the same ARC file. Through use of Java profiling tools, it was observed that too many queries are generated from Firefox and Opera browsers compared to Microsoft Internet Explorer. For example, 5 queries were generated for retrieving one page if the request is from MS IE while 129 queries were generated from Firefox or Opera. When processing each query, the Master Wayback had to go through the registered Wayback instances to get index data and thus created performance issues for web page retrieval.

To improve the performance, we implemented another idea based on re-directing the page request to its original Wayback instance. When a user sends a query for a particular web page to the Master through the Wayback interface, the Master will forward the query to all registered Wayback instances, collect index information and merge them for the user. In this process, only one query is created for each back-end instance. When the user gets the query results back, a list of links to the archived materials for the specified web site is displayed in the browser. When the user clicks a link to view the actual archived page, the request is sent directly to the appropriate Wayback sub-collection allowing us to by-pass the performance problem as described above. This idea not only solves the performance problem but also creates a distributed Wayback machine in which the front-end (or Master) Wayback machine will simply treat other Wayback instances as sub-collections (See Fig 2).



(1) User sends a query request.
(2) Front-end Wayback gets a list of Wayback resources from resource registry.
(3) Front-end Wayback sends the query to distributed resources.
(4) User gets search results for archived materials.
(5) User's page retrieval request is re-directed to original provider.

Fig 2. Fully Distributed Wayback Machines Implemented at SDSC

2.3 Build a Scalable Wayback

Although we haven't encountered any problems with the distributed Wayback deployed at SDSC, the approach still may have performance problems when the number of backend resources gets large. Going through all backend index repositories for each request is less efficient than storing the index data in a single parallel database. When dealing with distributed resources, breaking a giant collection into many portions also will create management overhead for file and software maintenance. The Internet Archive, for example, is managing a collection of over 500 Terabytes of compressed Internet archived materials. Assuming the index size is 2% of the total size, the index is around 2 Terabytes. In this case, a desirable solution would be to have a high performance database such as Oracle store the huge amount of index data. Also grid middleware such as the Storage Resource Broker (SRB) can be deployed to handle distribution of ARC files across multiple storage systems.

The choices for storing Wayback index data in the most recent release 0.7.0 is either a Berkeley database or sorted text files. It is possible to modify Wayback software so that it can have a flexible choice for storing index data in either Berkeley DB, sorted text files, Oracle or PostgreSQL. A convenient way to add index data into a database such as Oracle would be to use an existing command-line tool in Wayback 0.7.0 to generate text formatted index files and then bulk load the index content into the Oracle database.

For accessing ARC files that are distributed across heterogonous storage systems, the SRB data grid can be used as a middleware solution (See Fig 3). This can be accomplished by making some simple changes in the Wayback software to replace file I/O calls with corresponding calls from SRB's Jargon package, which is a collection of SRB APIs implemented purely in Java. If the Wayback is running on an operating system such as Linux, no modification is needed. The SRB collection can be mounted using SRB's Linux Userland File system interface.

Note that there is no need to upload files into the SRB. It is also not required that the files be mounted through NSF or other mechanisms to be accessible by the Wayback Machine. All ARC files can stay where they are and be registered into the SRB collection. In this model, the SRB provides a virtual global UNIX box for files sitting in storage systems that are even in different geographical locations. We will describe use of the SRB data grid in more detail in the next section on the NSDL project.

This model will be able to handle efficiently large collection of ARC files stored in heterogeneous storage hardware while our distributed model can be deployed to federate distributed collections.

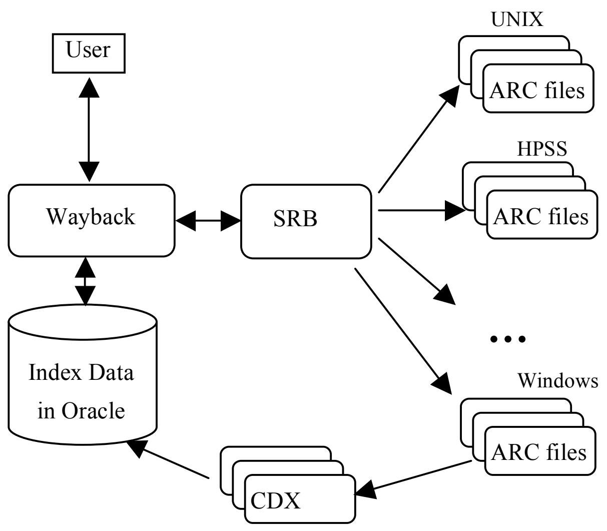


Fig 3. A Scalable Solution using Oracle database and SRB

3.1 NSDL Web Crawl Archive

In building the archive for the National Science Digital Library, the SRB team has been using the 'Wget' [7] software to crawl web sites that are registered by the NSDL community. A client-server based software system was developed on top of 'Wget' to handle load balancing among parallelized web crawl clients. In this approach, the web crawl server is responsible for assigning web URLs to clients, each of which launches 'Wget' to do the actual web crawling. The web files from the crawlers are stored on local disk as flat files and then are uploaded into SRB containers and stored in an archive.

Similar to UNIX tar files, SRB containers [8, 9] are designed to store large numbers of small files in an archival storage system. The SRB supports direct read and write of web pages from the containers. Containers may be replicated, metadata can be linked to each file within the container, and checksums are managed for each file. Replicated containers may be synchronized along with validation of the checksums. Access controls can be assigned to individual web pages. Under data grid control, it is possible to move containers between storage systems without losing either the access controls or links to metadata.

Unlike storing the manifest in its own tar file, the manifest for a SRB container is stored in SRB's metadata catalogue (MCAT).

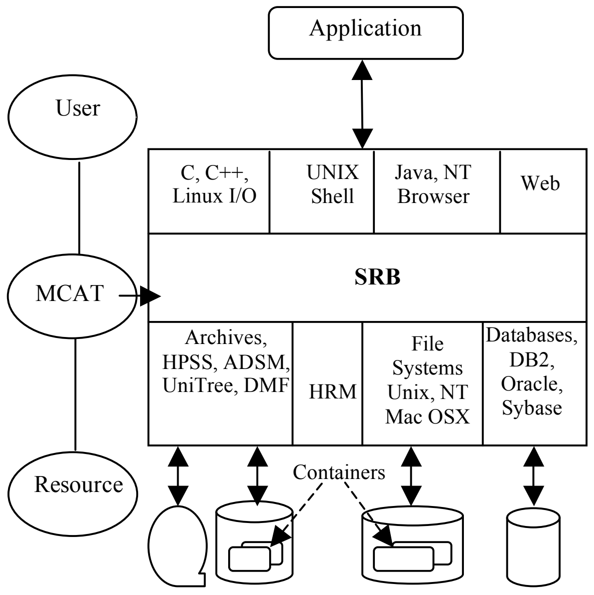

Application
User
C, C++, Linux I/O
UNIX Shell
Java, NT Browser
Web
MCAT
SRB
Archives, HPSS, ADSM, UniTree, DMF
HRM
File Systems Unix, NT Mac OSX
Databases, DB2, Oracle, Sybase
Resource
Containers

Fig 4. SRB: distributed data grid software

The metadata catalog (MCAT) is implemented in an Oracle database at SDSC for the NSDL project. When browsing a SRB collection, a file inside a container is just like a regular file. When a user issues a SRB command to access the file, the SRB software will contact the MCAT for the metadata information about the location of the file inside the container and then retrieve file data from its remote storage location.

3.2 Indexing and Accessing of NSDL Archived Web Pages

Not surprisingly, the key indices for the NSDL web archived materials stored within SRB are also paired values of URL and Capture Date. The archived files are stored under their corresponding SRB collection name, which has the Capture Date as the top level collection, followed by a sub-collection labeled by a MD5 string, which is created during the NSDL crawl and which has a one-to-one relationship to the URL. NSDL web archived materials are indexed automatically in this collection and sub-collection structure when web files are registered into the SRB. Usually containers are replicated onto multiple storage systems to minimize risk of data loss. The location of the replica is additional metadata that is stored in MCAT. The metadata associated with each web page includes:

Capture Date: The date the web crawl was done.

MD5 string: a unique string constructed from a hash of the URL to avoid problems between Unix and Windows naming conventions.

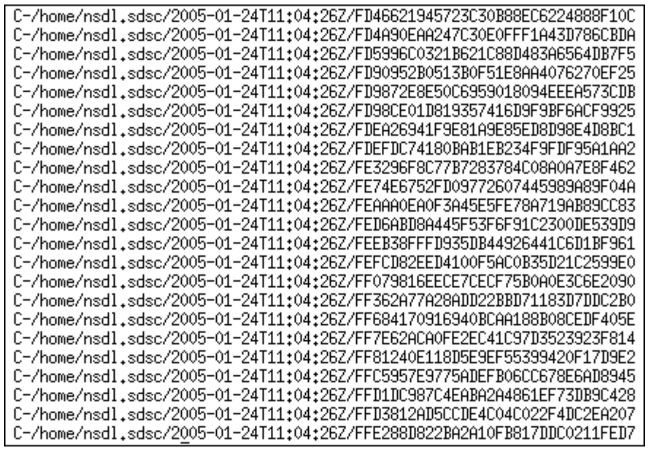
Storage host name: The machine name where a container is located.

Storage type: The type of storage system. It can be a UNIX file system, HPSS tape archive, Windows file system, etc.

Container name: The name of the container storing the web materials.

Offset: The offset in bytes serving as a starting point for the data for the archived web file.

File size: The file size in bytes for the archived web file.



```
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FD46621945723C30B88EC6224888F10C
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FD4A90EAA247C30E0FFF1A43D786CBDA
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FD5996C0321B621C88D483A6564DB7F5
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FD90952B0513B0F51E8AA4076270EF25
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FD9872E8E50C6959018094EEEA573CDB
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FD98CE01D819357416D9F9BF6ACF9925
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FDEA26941F9E81A9E85ED8D98E4D8BC1
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FDEFDC74180BAB1EB234F9FDF95A1AA2
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FE3296F8C77B7283784C08A0A7E8F462
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FE74E6752FD09772607445989A89F04A
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FEAAA0EA0F3A45E5FE78A719AB89CC83
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FED6ABD8A445F53F6F91C2300DE539D9
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FEEB38FFFD935DB44926441C6D1BF961
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FEFCD82EED4100F5AC0B35D21C2599E0
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FF079816EECE7CECF75B0A0E3C6E2090
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FF362A77A28ADD22BBD71183D7DDC2B0
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FF684170916940BCAA188B08CEDF405E
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FF7E62ACA0FE2EC41C97D3523923F814
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FF81240E118D5E9EF55399420F17D9E2
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FFC5957E9775ADEFB06CC678E6AD8945
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FFD1DC987C4EABA2A4861EF73DB9C428
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FFD3812AD5CCDE4C04C022F4DC2EA207
C-/home/nsdl.sdsc/2005-01-24T11:04:26Z/FFE288D822BA2A10FB817DDC0211FED7
```

Fig. 5 Web crawl index is embedded in the collection structure in SRB Metadata catalogue.

As a comparison between ARC files and SRB container technology, we summarize these two approaches for storing archived web materials in the following table.

	Wayback	**NSDL@SDSC**
Web Crawler	Heritrix	Parallelized 'Wget'
Archive Format	ARC format	SRB container
Index Data	inside each ARC file	SRB MCAT
Index Key	(URL, Capture Date)	(URL, Capture Date)
Index Store	Berkeley DB, CDX files	Oracle, PostgreSQL, DB2, etc.

Access software	Wayback	CGI script, Windows browser, workflow
Data Compression	Yes	Hardware IDRC

With the NSDL collection structure containing embedded index information, the access of the NSDL materials is done through a CGI script that issues SRB I/O calls to retrieve data quickly from SRB containers. A page retrieval request for a URL is thus a CGI request with a value of the full path for the web page inside SRB. Using SRB as a middleware, containers can be distributed across the network and can be even in different storage systems supported by SRB.

Summary

The management of material from massive web crawls presents significant scalability issues. These include not only management of the name space for identifying web pages, but also constructing a scalable system for page retrieval. Two systems are illustrated that are capable of managing web crawls with tens of millions of web pages.

ACKNOWLEDGMENTS

We would like to thank the Library of Congress for providing the impetus for the collaborative project for Internet archiving and digital preservation. We also would like to thank Brad Tofel from Internet Archive team for his technical support and useful discussion in developing our software at SDSC. The NSDL persistent archive is supported under the NSF grant "Core Integration of the National SMETE: Digital Library" under a subcontract from UCAR, contract number S02-36645.

REFERENCES

[1] http://en.wikipedia.org/wiki/Web_crawler

[2] http://www.nsdl.org

[3] http://www.cwr.cl/projects/WIRE/doc/

[4] Moore, R., "Building Preservation Environments with Data Grid Technology", American Archivist, vol. 69, no. 1, pp. 139-158, July 2006.

[5] http://www.waybackmachine.org

[6] http://crawler.archive.org

[7] http://www.gnu.org/software/wget

[8] http://www.sdsc.edu/srb

[9] http://www.sdsc.edu/srb/index.php/Container