Building a Research Library for the History of the Web

William Y. Arms
Computer Science Department
Cornell University
Ithaca, NY 14853
1-607-255-3046
wya@cs.cornell.edu

Selcuk Aya
Computer Science Department
Cornell University
Ithaca, NY 14853
1-607-255-7316
ayas@cs.cornell.edu

Pavel Dmitriev
Computer Science Department
Cornell University
Ithaca, NY 14853
1-607-255-5431
dmitriev@cs.cornell.edu

Blazej J. Kot
Information Science Program
Cornell University
Ithaca, NY 14853
1-607-255-4654
bjk45@cornell.edu

Ruth Mitchell
Cornell Theory Center
Cornell University
Ithaca, NY 14853
1-607-254-8689
mitchell@tc.cornell.edu

Lucia Walle
Cornell Theory Center
Cornell University
Ithaca, NY 14853
1-607-254-8775
lwalle@tc.cornell.edu

ABSTRACT

This paper describes the building of a research library for studying the Web, especially research on how the structure and content of the Web change over time. The library is particularly aimed at supporting social scientists for whom the Web is both a fascinating social phenomenon and a mirror on society.

The library is built on the collections of the Internet Archive, which has been preserving a crawl of the Web every two months since 1996. The technical challenges in organizing this data for research fall into two categories: high-performance computing to transfer and manage the very large amounts of data, and human-computer interfaces that empower research by non-computer specialists.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries – *collection, systems issues, user issues.* J.4 [**Social and Behavioral Sciences**]: *sociology.*

General Terms

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

Keywords

history of the Web, digital libraries, computational social science, Internet Archive.

1. BACKGROUND

1.1 Research in the History of the Web

The Web is one of the most interesting artifacts of our time. For social scientists, it is a subject of study both for itself and for the manner in which it illuminates contemporary social phenomena. Yet a researcher who wishes to study the Web is faced with major difficulties.

An obvious problem is that the Web is huge. Any study of the Web as a whole must be prepared to analyze billions of pages and hundreds of terabytes of data. Furthermore, the Web changes continually. It is never possible to repeat a study on the actual Web with quite the same data. Any snapshot of the whole Web requires a crawl that will take several weeks to gather data. Because the size and boundaries of the Web are ill defined, basic parameters are hard to come by and it is almost impossible to generate random samples for statistical purposes.

But the biggest problem that social scientists face in carrying out Web research is historical: the desire to track activities across time. The Web of today can be studied by direct Web crawling, or via tools such as the Google Web API[1], while Amazon has recently made its older Alexa corpus commercially available for the development of searching and related services[2]. However, the only collection that can be used for more general research into the history of the Web is the Web collection of the Internet Archive[3].

1.2 The Internet Archive

Everybody with an interest in the history of the Web must be grateful to Brewster Kahle for his foresight in preserving the content of the Web for future generations, through the not-for-profit Internet Archive and through Alexa Internet, Inc., which he also founded.

JCDL'06, June 11-15, 2006, Chapel Hill, North Carolina, USA.

ACM 1-59593-354-9/06/0006.

[1] The Google Web Search API allows a client to submit a limited number of search requests, using the SOAP and WSDL standards. See: http://www.google.com/apis/.

[2] See http://websearch.alexa.com/welcome.html for the Alexa corpus made available by Amazon. This site also has a description of the relationship between Alexa Internet and the Internet Archive.

[3] The Internet Archive's Web site is http://www.archive.org/.

The Internet Archive began to collect and preserve the Web in 1996. With a few gaps in the early years, the collection has added a full crawl of the Web every two months since then. Most but not all of this data comes from the Alexa crawls. Statistics of the sizes of the separate crawls are complicated by the fact that a single crawl may contain several variants of the same URL, but in August 2005 the total volume of data was 544 Terabytes (TB). This is the size of the compressed data. As discussed below, the overall compression ratio is about 10:1, so that the total size of the collection is approximately 5 to 6 Petabytes uncompressed. Table 1 gives estimates of the size of the individual crawls for each year.

Table 1. Estimates of crawl sizes (compressed)

Year	Web pages (TB per crawl)	Metadata (TB per crawl)
1996	1	0.2
1997	2	0.4
1998	3	0.6
1999	4	0.8
2000	10	1.2
2001	15	2
2002	25	3
2003	30	4
2004	45	6
2005	60	10

We are working with the Internet Archive to build a research library based on this collection. In summer 2005, we began work on the system that is being used to transfer a major subset of the data to Cornell and to organize it for researchers, with a particular emphasis on supporting social science research. This paper describes the technical design, performance testing, and progress in implementation.

The overall goals of the library and plans for its use in research are described in a separate paper [1].

1.3 User Studies

In building any library, the objective is to organize the collections and services so that they provide the greatest range of opportunities for users, both now and in the future. Inevitably the design is a trade-off between predictions of what users will find helpful and the practicalities of building and maintaining the library. This trade-off is particularly important for a library of the whole Web because of the computing challenges of managing very large amounts of data.

Therefore, the design of the library began with interviews of potential users to identify how the collections might be organized to be most valuable to them. Two users studies were carried out, with sociologists and with computer science researchers.

1.3.1 Sociology

In fall 2005, Cornell received support from the National Science Foundation's Next Generation Cybertools program for a project that combines sociology research with continuing development of the Web library[4]. In this project, the specific areas of research are diffusion of ideas, including polarization of opinions and the spread of urban legends. Conventionally, sociologists have studied such phenomena by analysis of small surveys with hand-coded data. One aim of the project is to develop a new methodology for such research built around very large-scale collections of Web data, with automated tools used to extract, encode and analyze the data.

Social science researchers identified a number of specific studies that they would like to carry out using historical Web data. Many of the studies have the same general structure: (a) extract a subset of the Web for detailed analysis, (b) encode selected attributes of the pages in that subset, (c) repeat for the corresponding subsets at several different dates, (d) analyze the changes over time.

The criteria by which a portion of the Web is chosen for analysis are extremely varied. Some desirable criteria are impossible with today's computing, e.g., they require understanding of the content of a page. However, simple criteria such as domain names provide a good starting point for many purposes, particularly when combined with focused Web crawling to refine the subsets for analysis. Once a subset has been extracted, social science researchers want to analyze the text, for which full text indexes are important. They also wish to analyze the structure of links between pages for the social relationship that they represent.

Such research requires interdisciplinary efforts by computer scientists and social scientists. Some of the analysis tools already exist, e.g., using full text indexes of Web pages to trace the movement of individuals. Others tools are themselves subjects of computer science research in natural language processing and machine learning, e.g., to analyze the text of Web pages for sentiments, opinions, and other features of interest to social scientists.

1.3.2 Computer Science

Ten computer scientists who carry out research on the Web contributed to the user studies. Their research areas include the structure and evolution of the Web, data mining, digital libraries, machine learning, and natural language processing. Most of their interest focuses on the textual content of Web pages and the structure of the Web as revealed by the graph of links between pages. Several of the researchers commented that they expend ninety percent of their effort gathering test data; even then they have difficulty in determining how robust the results are across time.

A fundamental tool for such research is the Web graph of links between pages. Studies of the graph are very important in understanding the structure of the Web, and the graph is the basis of practical tools such as PageRank [3] or Hubs and Authorities [9]. Despite its importance, there have been few studies that have looked at changes in the Web graph over time. Many of the classical studies of the Web graph were based on early AltaVista crawls and have never been repeated. Algorithmic research needs graphs of at least one billion pages, preferably stored in the main memory of a single computer.

For textual research on the Web there are two additional requirements. The first is snapshots that are repeated across time

[4] Michael Macy (principal investigator), et al., "Very Large Semi-Structured Datasets for Social Science Research". NSF grant SES-0537606. http://www.infosci.cornell.edu/SIN/cybertools/

that can be used for burst analysis, and other time based research. The second is full text indexes of substantial numbers of Web pages.

Focused Web crawling is of particular importance in digital libraries research. Part of the original motivation for developing this library was an interest in automatic selection of library materials from the Web [2, 10]. Using the actual Web for research in focused crawling is technically difficult and the results are often hard to interpret since no experiment can ever be repeated with exactly the same data.

2. ARCHITECTURE

2.1 Data Management

The Internet Archive uses highly compressed file formats developed in conjunction with Alexa Internet. Compressed Web pages are packed together in large files using the ARC format [4]. The pages in each ARC file are in essentially random order, usually the sequence in which the Web crawler originally captured them. Every ARC file has an associated DAT file, which contains metadata for the pages including URL, IP address, crawl date and time, and hyperlinks from the page. The files are compressed with gzip. Ten years ago the decision was made that ARC files should be approximately 100 MB, which seemed big at the time, but this size is now too small for efficiency and will need to be increased. The sizes of the DAT files depend on the number of pages in the associated ARC files, but average about 15 MB. The compression ratios also vary widely. The ratio is more than 20:1 for text files but close to 1:1 for files that are already compressed efficiently, such as videos. The overall ratio for ARC files is about 10:1.

2.1.1 The Database

The Cornell Web library uses a relational database to store metadata about the Web pages and a separate Page Store to store the actual pages. In addition, the unprocessed ARC and DAT files received from the Internet Archive are copied to a tape archive. In choosing a relational database, we considered but rejected two approaches that have been successful in related applications.

The first option was to use a modern digital library repository with support for rich data models, such as XML and RDF, and search services that support semi-structured data, such as XQuery. Such capabilities are appealing, but we know of no repository system that can manage tens of billions of objects. The scale of the Web precludes such an approach.

The second option was to follow the model of production services for the Web, such as Google [7] and Yahoo. They provide low cost processing and data storage by spreading their systems across very large numbers of small, commodity computers used as servers. This is the approach used by the Internet Archive to store its collections and for its very popular Wayback Machine[5]. We rejected this architecture for a research library for two principal reasons: (a) there are many algorithmic computations on large datasets where a single large computer is intrinsically more efficient than a distributed cluster of smaller machines, and (b) even when the research can be done effectively, clusters of computers are more difficult to program by researchers who are carrying out Web-scale research. As an example, each server at the Internet Archive has an index of the files stored on it, but there is only a very limited central index. The Wayback Machine allows a user to retrieve all the pages in the entire collection that have a given URL. It relies on a protocol in which an identifier is broadcast and each server responds with a list of matches. This is very efficient for this specific purpose, but it would be extremely difficult to extract the flexible subsets required by social science researchers with this organization of data.

A relational database has many advantages for the Web library and one major disadvantage. Foremost among the advantages is scalability. Commercial relational database systems are highly optimized for storing, loading, indexing, extracting, backing-up, and restoring huge volumes of data. Usability is another important advantage. A relational schema provides a single image of the collection, expressed in a manner that is familiar to many researchers. The disadvantage is a loss of flexibility. The design and implementation of the database attempt to reconcile the expected uses that will be made of the library against scalability constraints, but it will be difficult to make major changes to the schema without rebuilding the entire database.

The actual Web pages are stored in a separate Page Store. At the Internet Archive, if two Web pages are identical they are stored twice. With the new Page Store, duplicate pages are stored only once. Rather surprisingly, there is as yet very little data about how many pages remain unchanged between crawls, but we expect that elimination of duplicates will save significant online storage, especially with large audio and video files.

The Page Store is implemented as a set of compressed files, one file for each page received in the ARC files. Since many pages on the Web do not change between crawls, the Preload subsystem checks for content duplicates using an MD5 check sum of the content. Thus, a copy of the content is stored only once however many pages have that content. In order to guarantee fast access to the stored content, each page's content is compressed individually.

The architecture of the Page Store allows decisions to be made about which pages to store online at any given time. For example, the library might decide not to store large audio and video files online. While all metadata will be online at all times, an individual Web page could be accessed from the online Page Store, the off-line tape archive, or over the Internet from the Internet Archive.

2.2 Equipment

The library is housed at the Cornell Theory Center, which is the university's high-performance computing center. The choice of equipment and the use of a relational database were closely related decisions. The Theory Center has expertise in distributed cluster computing, but because of the very high data rates, a symmetric multi-processor configuration was chosen instead. Figure 1 shows part of the configuration of the central computer.

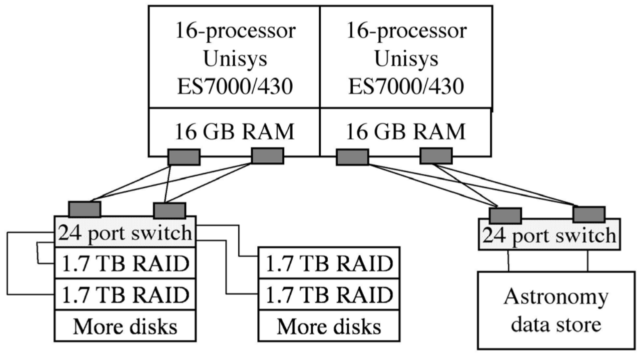

Figure 1. Configuration of the main computer system

[5] The Wayback Machine is accessible at http://www.archive.org/.

The system is shared with another data-intensive program of research, the analysis of data from the Arecibo radio telescope in Puerto Rico. Each group has use of a dedicated Unisys ES7000/430 server, with 16 Itanium2 processors running at 1.5 Gigahertz. The memory can be shared between the servers, but in practice each project has sole use of 32 GB. Each server has a RAID disk subsystem attached via a dual-ported fiber-channel. The operating system is Microsoft Windows Server 2003.

For the Web library the disk subsystem provides an initial 45 TB of disk storage. We plan to extend the capacity to 240 TB by 2007. There are no technical barriers to adding additional fiber channels and disk capacity. In the longer-term, disk prices are falling faster than the growth in the size of Web crawls, which gives confidence that the library will be able to keep up with the growth of the Web.

By using a symmetric multi-processor configuration with a high performance disk subsystem, we are able to balance processing and disk access requirements. Since the data sets are local to the system on which the database is located, the system can perform bulk-loading tasks without incurring any networking penalties.

The large real memory is an added attraction of this configuration. It allows researchers to carry out substantial computation entirely in memory. For instance, it is possible to process a Web graph of one billion pages within memory.

2.3 The Human Interface

The design of the human interface is perhaps the most challenging aspect of developing the library. The social science research groups that we are working with have the technical skills to write scripts and simple programs. Many are experts in statistical calculations. But they should not be expected to write large or complex computer programs. The current design supports three categories of users.

- The *Basic Access Service* provides a Web Services API that allows a client to access pages in the collection by any metadata that is indexed in the database, e.g., by URL and date. The Retro Browser, which is described below, uses this API to allow a user to browse the collection as it was at a certain date.
- The *Subset Extraction Service* supports users who wish to download sets of partially analyzed data to their own computers for further analysis. A Web form is provided to define a subset of the data (e.g., by date, URL, domain, etc.), extract subsets of the collection, and store them as virtual views in the database. Sets of analysis tools, many of which are already under development, can be applied to the subset and the results downloaded to a client computer.
- Technically advanced users can be authorized to run their own programs on the central computer.

To support the Basic Access Service and the Subset Extraction Service, we provide a dedicated Web server, which is housed next to the main system.

3. SCALABILITY EXPERIMENTS

Although the library has been generously funded by the National Science Foundation, we do not yet have sufficient capacity to download and mount online the entire Web collection of the Internet Archive. This is the long term goal, but during the initial phase, care has been taken to balance the several parts of the system: online storage, network bandwidth, processing of the incoming data, database, performance, and the need to archive, back-up, and restore the data. In spring 2005, several undergraduate and masters students carried out independent projects to estimate sizes and processing requirements[6].

To test database performance before large amounts of actual data were available, we used the R-MAT algorithm to generate a synthetic graph with properties similar to the Web graph [5]. This test graph has one billion nodes with more than seven billion links, and domain names generated according to their distribution on the real Web [11].

Based on these benchmarks, the decision was made to install a 100 Mb/sec network connection to the Internet Archive and to load data at a sustained rate of 250 GB/day, beginning January 2006. This rate will enable the library to acquire and mount online by the end of 2007 a complete crawl of the Web for each year since 1996. This phase will require approximately 240 TB of disk storage. Note that the disk requirement differs from the estimates of raw data shown in Table 1. The database with its indexes is less highly compressed than the raw data, but savings are made in the storage of duplicate data, both in the database and the Page Store.

During fall 2005, first generation software was written for (a) the data flow system that brings data to the library, and (b) the user API and tool sets. They are described in the next two sections.

4. DATA FLOW

Figure 2 shows the flow of data into the library. When ARC and DAT files are received from the Internet Archive, the first step is to store them in the tape archive. The Preload system then unpacks the raw data, extracts metadata, and prepares batch files for loading into the database and Page Store.

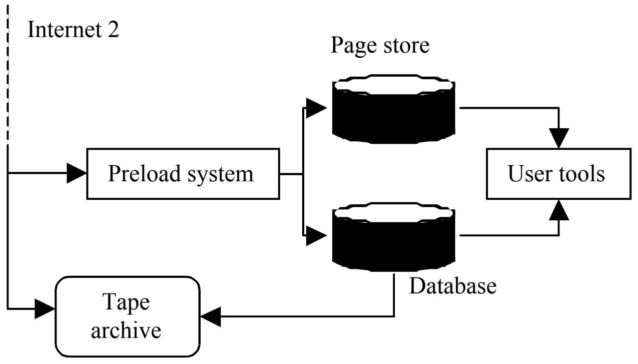

Figure 2. Flow of data into the library

Figure 2 does not show the data tracking system. This is a major subsystem that manages the data transfers, monitors all errors, and tracks the tens of millions of files within the library.

4.1 Networking

Internet2 is used to transfer data from the Internet Archive in San Francisco, California to Cornell University in Ithaca, New York. For this purpose, a 100Mbit/sec link has been established from the Internet Archive to Internet2. Both Cornell and the Internet Archive have internal networks with 1 Gbit/sec or greater performance.

In the future, the National LambdaRail and the TeraGrid are intriguing possibilities. These new networks have the capacity to go

[6] These student reports are available at http://www.infosci.cornell.edu/SIN/WebLib/papers.html.

beyond bulk data transfer and support genuine distributed processing between the Web library and the Internet Archive. For example, if large audio and video files are not stored online, an application could use the TeraGrid to retrieve individual large files from the Internet Archive on demand.

At the end of December 2005, a series of experiments were run to measure the sustained throughput of multi-threaded FTP transfers over Internet2, using Transport Layer Security. These measurements showed transfer rates of 280 GB per day before system tuning, or rather better than 30 percent of the theoretical maximum throughput of the link to the Internet Archive. This is sufficient for the planned rate of 250 GB per day. If greater bandwidth proves necessary, the link from the Internet Archive to Internet2 can be upgraded to 500Mbps inexpensively, while the Cornell Theory Center will soon have enormous bandwidth available via the TeraGrid.

4.2 Preload Subsystem

The Preload subsystem takes incoming ARC and DAT files, uncompresses them, parses them to extract metadata, and generates two types of output files: metadata for loading into the database and the actual content of the Web pages to be stored in the Page Store. Metadata for loading into the database is output in the form of 40GB text files, a separate file for every database table.

To satisfy the speed and flexibility requirements, the Preload system is designed to run as a set of independent single-thread processes, avoiding all inter-process communication and locking over input or output data. Likewise, each process writes its own output files. This design allows for easy configuration of the system to run a required number of processes on a given number of processors, on one or more machines. To determine each process's input, input files are partitioned by the first k bits of the MD5 hash sum of the filename, where 2k is the total number of processes in the system. The design of the subsystem does not require the corresponding ARC and DAT files to be processed together.

A series of experiments were run to test the performance of the Preload system, using the metadata from the synthetic Web graph. The experiments used 1, 2, 4 and 8 processors, with the data partitioned into 16 parts, according to the first 4 bits of the hash sum. Separate experiments were made for ARC and DAT files. Figure 3 shows results for the ARC files. The x-axis shows the number of CPUs used, and the y-axis shows the throughput in KB/sec. The white bar shows throughput per processor and the shaded bar shows total throughput. Adding more processors slightly decreases the throughput per processor due to contention for random disk accesses. The total throughput increases steadily up to four processors. After that, disk contention becomes too high, and the throughput actually declines. The results for DAT files are similar, with the total throughput flattening after four processors.

From these experiments, we conclude that four processors are optimal. The corresponding throughputs are 73 MB/sec (about 6 TB/day) for ARC files, and 12 MB/sec (about 1 TB/day) for DAT files.

When metadata from the DAT files is uncompressed its size increases by a factor of about 11:1. Fortunately, much of this data is duplicated. For example, a given URL may occur many times in a crawl and be replicated in many crawls. Therefore duplicate elimination has been a major consideration in refining the database design.

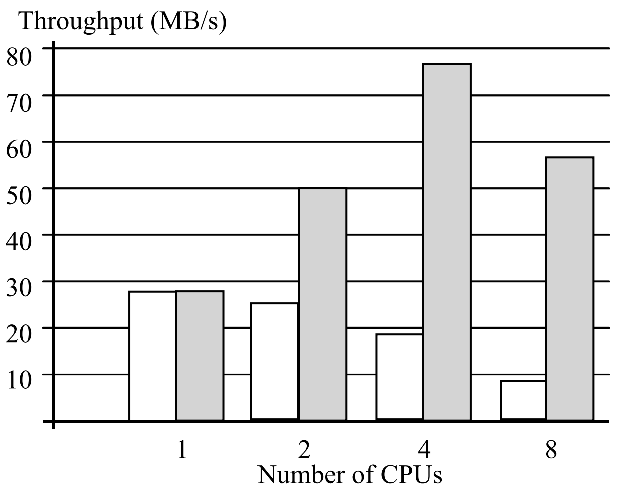

Figure 3. Performance of Preload system (ARC files)

Note that during the above experiments no other processes were running on the system. The overall performance will be lower when the Preload system is run in production mode at the same time as other subsystems. Also, during the preliminary phase, only text and HTML files were fully analyzed to extract links and anchor text. Processing of other file formats will be added in the future. Some of these formats will be computationally intensive, e.g., PDF files.

4.3 Database Design

The relational database uses Microsoft SQL Server 2000. Three important goals when designing the database schema and deciding how to load the data to the database were: (a) minimize the storage requirements, (b) maximize the load throughput, and (c) support efficient logging, backup, and restore.

Conceptually, for each crawl, the database stores metadata about each page (e.g., information about the content of the page, URL of the page) and about the links between them (including anchor text and text surrounding the anchor text). However, to avoid storing redundant data, the schema is denormalized. The denormalized schema is shown below in Figure 4. Information about URL and domain names is stored in a look-up table, since the same URLs and domain names appear many times in a single crawl and across crawls. For similar reasons, anchor text, text surrounding the anchor text, and information about page content are stored in the look-up tables Dictionary and Page Content respectively, as shown in the schema in Figure 4. To make the loading of the data faster, separate tables for each of Page, Page Content and Link are created for each crawl while the other tables (e.g., the URL table) are shared among crawls.

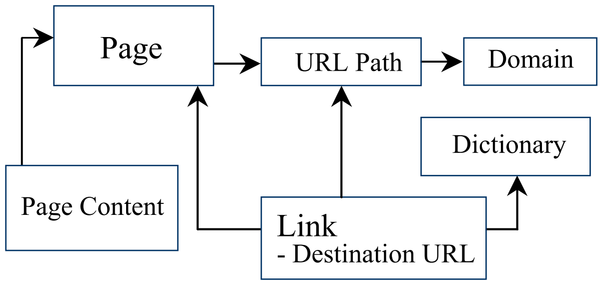

Figure 4. Database design: the de-normalized schema

The Preload subsystem outputs separate files conforming to the schema described above and these files are bulk loaded into the database. There are many parameters that affect the bulk load performance. These parameters include: batch size, file size, degree of parallelism, and interaction with the logging, backup and recovery system. The synthetic Web data was used to understand how these parameters affect the loading performance and to tune them. The first sets of experiments were used to determine the optimal file size for bulk loading and the number of CPUs used. In these experiments, default, recovery and backup mechanisms were used [6].

The results of the first set of experiments indicate that it is optimal to load each file as a single batch; a file size of 40GB and 4 CPUs gave the best performance, and around 800GB could be loaded in one day. However, the experiments to determine the file size and degree of parallelism showed significant variability. Using the default logging provided by MS SQL Server 2000, checkpointing and nightly backups were consuming enormous resources. They interfered with the bulk loading process and are a probable cause of the variance seen in the performance benchmarks.

Two observations were made to overcome the performance penalty. First, data is append-only while being bulk loaded and is read-only after the bulk load is complete; logging, recovery and backup mechanisms can be customized to increase the performance and decrease the variance in loading times. Second, tables in the schema can reside on different disks and thus can be written in parallel. Following these two observations, in the current design each table can be put onto a separate disk as shown in Figure 5. Moreover, Page, Page Content and Links information for each crawl are put into separate files. This partitioning according to crawls is easy in MS SQL Server as separate tables for each of Page, Page Content and Link are created for each crawl.

The database load subsystem is divided into two programs: a high-level program that organizes the processes and a low level program that runs separate loads in parallel. The workflow for loading each table in each crawl consists of five major steps: (a) Get the files produced by the Preload subsystem for the current table in the current crawl and write the relevant log information to an administrative database; commit the transaction writing the log information. (b) Write files of the table to the disk corresponding to the current table via the low level program; files corresponding to different tables can be written in parallel. (c) Create the necessary indexes. (d) Back-up the newly written data. (e) Write to the log the relevant information to indicate that processing of the files for the current table in the current crawl is complete and commit the transaction writing the log information. In MS SQL Server 2000, backups and index creation are all atomic.

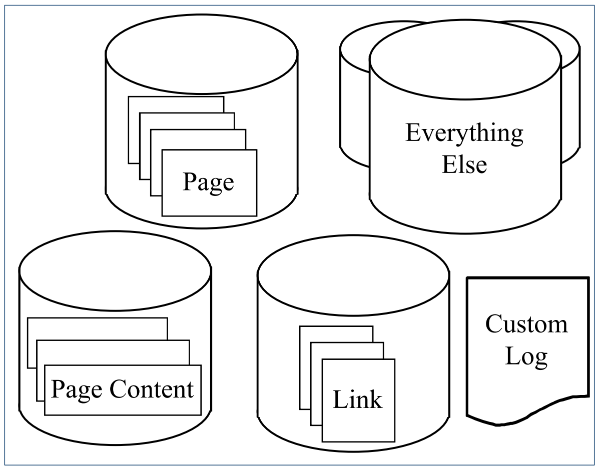

Figure 5. Database design: organization of file system

This new design is being implemented and tested, as of January 2006. First indications are that the performance will comfortably meet the required performance goals. Extensive benchmarking is required to tune many parameters, such as batch size, file size, degree of parallelism, and the index management.

4.4 Archiving

Archiving and back-up are expensive operations with complex trade-offs. Without care, the networking bandwidth and disk throughput used in logging, back-up, and writing to the tape library could have a major impact on the system throughput.

As described above, the database design allows the database files to be backed up incrementally. This provides two options for restoring the database, by reprocessing the raw data or from the back-up. The Page Store is not backed-up. If parts of it were ever corrupted, they would have to be restored by reprocessing the raw data. A current design project is to reorganize the Page Store to permit efficient restoration.

The library uses a robotic tape library with LTO3 tape drives. This is shared with other systems at the center. All unprocessed ARC and DAT files are copied to the tape library to be stored indefinitely. This preserves another copy of the Internet Archive's data for the long-term. This data is unique and could never be replaced. The industry standard life of these tapes is thirty years but our expectation is that six years is a more probable time before the tape library is replaced and all the data will have to be copied onto fresh media.

5. SUPPORT FOR THE USER

Figure 6 shows the architecture of the interface that the library offers to users. This is a three-tier architecture. The data tier consists of the relational database of metadata and the Page Store; the middleware tier provides services to access the data, tools to analyze it, and a choice of Web Services APIs; clients interact with the middleware tools either through the APIs, or directly.

5.1 Clients

The user tools system was designed to be extensible and scalable. Specifically, it supports two categories of users: (a) users who analyze the data remotely from their own computers, perhaps at another institution or even in another country, and (b) computationally intensive users, who may wish to run very heavy analyses of the data using the Cornell Theory Center computing environment. Corresponding to these two categories of users the architecture supports two types of clients: Web services clients and clients that execute their own programs on the library's servers. The first category requires little technical sophistication from the user, while the second category trades complexity against the flexibility of being able to write custom programs and to use the full processing power available.

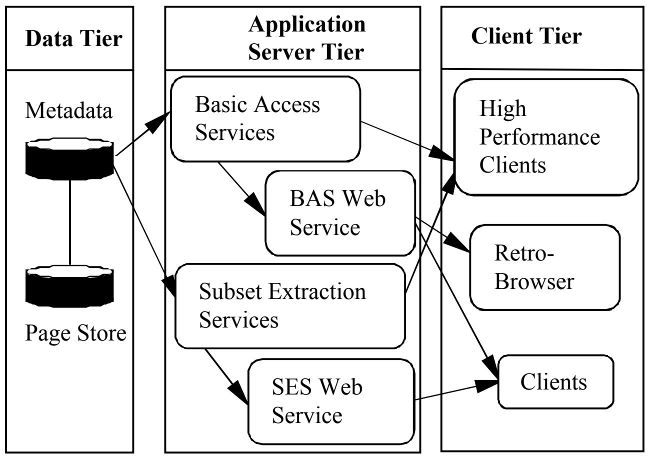

Figure 6. Architecture of the interfaces provided for users of the library

Web services clients are intended to be used remotely, with moderate demands on their access to the data. Examples of these clients include running queries using a full text index, or fetching specific pages using existing indexes. Web service clients may also be used to start, control, and retrieve results from experiments run by high-performance clients. Web services are implemented by using Microsoft's ATL server libraries. They run on a dedicated Web server. The clients themselves can be implemented in any language or environment that supports Web services standards. Users of these clients do not need to know how the data is stored. They are provided with forms that are automatically converted to SQL commands by the middleware.

The high-performance clients will for the most part run within the Cornell Theory Center. They will usually have a high bandwidth connection to the database. Clients of this form may carry out research that need lots of computation power, e.g., experiments that process very large subsets or analyze how the structure of the Web changes over time. These clients are implemented by linking against dynamic link libraries (DLLs) provided by the application server tier.

5.2 Access to the Database

The application server tier accesses the database using Microsoft database tools. Two main areas of functionality have been implemented in this tier: Basic Access Services (BAS), and the Subset Extraction Services (SES). Each consists of two parts: a set of services, implemented as a series of dynamic link libraries (DLLs) written in C++, and a Web Services API.

Basic Access Services are for clients that interact directly with the database. They allow a client to fetch pages given a combination of URL and date of crawl. They also allow a client to check within which crawls a given page is available. For example, a focused Web crawler can retrieve pages from a specified crawl, using a simple client script that interfaces to the BAS Web services API.

Subset Extraction Services allow a client to select a part of the data as a subset. Once created, this subset is stored in the database as a view. Such subsets are useful for running experiments over a smaller, perhaps random, sample of the Web, as well as selecting relevant pages for a particular experiments, such as those from a given domain. For example, a researcher studying government Web sites might extract textual pages from the .gov domain for a selected range of dates.

5.2.1 Users Beyond Cornell

This digital library is intended for the use of all academic researchers, not only those based at Cornell. Technically, this is straightforward. The library is connected to the Internet, including Internet2, and will soon be available via the TeraGrid.

We are currently developing a code of use policies for researchers. Mining this data has potential for abuses of privacy. Cornell researchers need to follow the university's procedures for such research and we need to find convenient ways to extend the code of practice to all users of the library.

5.3 User Tools

5.3.1 The Retro Browser

The Retro Browser is an example of a Web services client that is already implemented and running in a test environment [12]. To a user, it appears to be a regular Web browser, except that it browses an historical snapshot of the Web.

The Retro Browser is designed with the non-technical user in mind. The design assumes that the user may not be technically proficient and should not be expected to install new software or run special scripts. After the user has made an initial choice of a date in Web history, the Retro Bowser behaves like any other Web browser. The user uses a standard browser to carry out all the standard Web tasks, such as download an applet, run a script, submit a forms, etc. The only difference is that every URL is resolved to the record in the Web library for the specified date.

The major component of the Retro Browser is a Web server configured to be used as a proxy server. To obtain the data from the database, the proxy server utilizes the Basic Access Web Service API.

The Retro Browser client interacts with the Retro Browser proxy in a standard HTTP client-server fashion. To fetch the appropriate page from the database requires a URL and the date of the crawl, which is represented by a crawl ID. The proxy server expects a session cookie specifying a crawl ID with every request. If such a cookie is not found with the request, the user is asked to specify the crawl ID. Further requests may or may not contain a cookie since cookies are tied to a domain. However, the Retro Browser proxy ensures that the cookie is replicated for all domains using a series of redirects. In this manner, the architecture ensures that the user is asked to specify the crawl date for the first request only.

5.3.2 Analysis of the Web Graph

A set of analysis tools is under development that will provide more complex access and analysis functions. These tools are part of the application server tier. They are applied to subsets of the data and accessed either directly or through the Subset Extraction Services API.

One group of tools operates on the Web graph of a subset. Hyperlinks from each Web page are stored in the database. Representation of the graph is by its adjacency matrix using a compressed sparse row representation. Preliminary software has been written to read all the links from a given subset of the data and construct the adjacency matrix. The matrix is then stored in the file system in a compressed form, which allows performing the basic operations, such as matrix addition and multiplication. The Cuthill-McKee algorithm is used to reorder the nodes to create dense blocks within the matrix to increase the compression ratio and allow in-memory processing [8].

5.3.3 Full Text Indexes

Full text indexes are a vital tool for many researchers. For instance a social science researcher may wish to track the Web pages that refer to a named individual or may identify trends by burst analysis of terms used on the Web.

Our initial approach is to provide an indexing service for data subsets, using the Nutch search engine. It is straightforward to extract a subset, which is represented by a database view, and create a full text index of all textual pages in the subset. The only problem is the processing necessary to index a very large subset.

We are in discussions with Cutting, the principal developer of the Lucene and Nutch search engines[7]. He has been working with the Internet Archive to build indexes of very large collections of Web pages in ARC format. For this purpose, they are developing a modified version of Nutch, known as Nutch WAX (Web Archive eXtensions). Rather than duplicate this effort, we are exploring the possibility of providing access to these indexes through the Basic Access Service.

6. ACKNOWLEDGEMENTS

We wish to thank Brewster Kahle, Tracey Jaquith, John Berry and their colleagues at the Internet Archive for their support of this work.

The following Cornell students have contributed to the development of the library described in this paper: Mayank Gandhi, Nicholas Gerner, Min-Daou Gu, Wei Guo, Parul Jain, Karthik Jeyabalan, Jerrin Kallukalam, Serena Kohli, Ari Rabkin, Patrick Colin Reilly, Lipi Sanghi, Shantanu Shah, Dmitriy Shtokman, Chris Sosa, Samuel Benzaquen Stern, Jimmy Yanbo Sun, Harsh Tiwari, Nurwati Widodo, Yu Richard Wang.

This work has been funded in part by the National Science Foundation, grants CNS-0403340, DUE-0127308, and SES-0537606, with equipment support from Unisys and by an E-Science grant and a gift from Microsoft Corporation.

7. REFERENCES

[1] Arms, W., Aya, S., Dmitriev, P., Kot, B., Mitchell, R., Walle, L., A Research Library for the Web based on the Historical Collections of the Internet Archive. *D-Lib Magazine*. February 2006. http://www.dlib.org/dlib/february06/arms/02arms.html

[2] Bergmark, D., Collection synthesis. *ACM/IEEE-CS Joint Conference on Digital Libraries*, 2002.

[3] Brin, S., and Page. L., The anatomy of a large-scale hypertextual Web search engine. *Seventh International World Wide Web Conference*. Brisbane, Australia, 1998.

[4] Burner, M., and Kahle, B., *Internet Archive ARC File Format*, 1996. http://archive.org/web/researcher/ArcFileFormat.php

[5] Chakrabarti, D., Zhan, Y., and Faloutsos, C., R-MAT: recursive model for graph mining. *SIAM International Conference on Data Mining,* 2004.

[6] Gerner, N., Sosa, C., *Fall 2005 Semester Report for Web Lab Database Load Group.* M.Eng. report, Computer Science Department, Cornell University, 2005. http://www.infosci.cornell.edu/SIN/WebLib/papers/Gerner2005.doc.

[7] Ghemawat, S., Gobioff, H. and Leung, S., The Google File System. *19th ACM Symposium on Operating Systems Principles,* October 2003.

[8] Jeyabalan, K., Kallukalam, J., *Representation of Web Graph for in Memory Computation*. M.Eng. report, Computer Science Department, Cornell University, 2005. http://www.infosci.cornell.edu/SIN/WebLib/papers/JeyabalanKallukalam2005.doc.

[9] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Ninth ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[10] Mitchell, S., Mooney, M., Mason, J., Paynter, G., Ruscheinski, J., Kedzierski, A., Humphreys, K., iVia Open Source Virtual Library System. *D-Lib Magazine,* 9 (1), January 2003. http://www.dlib.org/dlib/january03/mitchell/01mitchell.html

[11] Shah, S., *Generating a web graph*. M.Eng. report, Computer Science Department, Cornell University, 2005. http://www.infosci.cornell.edu/SIN/WebLib/papers/Shah2005a.doc.

[12] Shah, S., *Retro Browser*. M.Eng. report, Computer Science Department, Cornell University, 2005. http://www.infosci.cornell.edu/SIN/WebLib/papers/Shah2005b.pdf.

[7] The Lucene search engine is described at http://lucene.apache.org/. Nutch is described at http://lucene.apache.org/nutch/.