The WWW Prototype of the Alexandria Digital Library

D.Andresen, L.Carver, R.Dolin, C.Fischer, J.Frew, M.Goodchild, O.Ibarra, R. Kothuri, M.Larsgaard, B.Manjunath, D.Nebert, J.Simpson, T.Smith, T.Yang, Q.Zheng
Alexandria Digital Library, University of California at Santa Barbara
Santa Barbara, CA 93106, USA
Phone: 805-893-7665; Fax: 805-893-3045; E-mail: smithtr@cs.ucsb.edu

Abstract

The Alexandria Digital Library (ADL) is focussed on providing broad access to distributed collections of spatially-indexed information. ADL has a four-component architecture involving *collections*, *catalog*, *interfaces*, and *ingest facilities*. The first stage in the construction of ADL resulted in the design and implementation of a rapid prototype (RP) system. The second stage, which is described in this paper, involves an expansion of the functionality of the RP and its extension to the World-Wide-Web (WWW). We describe issues arising in each of the components of the architecture in extending the library to WWW as well as our current resolution of these issues. We also discuss an extension of the class of supportable queries to include simple, content-based queries involving geographic "features" and image textures. The metadata of ADL has been extended to include *gazetteer* information supporting the first class of extended queries. We discuss image processing and parallel computing support for ADL.

Keywords: digital library, spatial information, metadata, prototype, World-Wide-Web

1 Introduction

The primary goal of the Alexandria Project is to construct a distributed digital library (DL) for spatially-indexed materials [1]. The Alexandria Digital Library (ADL) will comprise a set of nodes distributed over the Internet, with each node supporting library components that include collections, catalogs, interfaces, and ingest facilities [2].

Key aspects of the ADL development strategy include 1) a design focused on *digitally supportable extensions* to traditional library functionality and consistent with requirements of the library community; 2) a distributed library that is accessible by many classes of users from the Internet; 3) user access to both the explicit and implicit information available in DL collections; 4) an evolutionary, reusable, and incremental approach to design and implementation; 5) particular focus on developing the user interface (UI) and catalog components of the DL architecture; 6) initial focus on building collections of *spatially-indexed* materials. In particular, we believe that basing ADL on the Internet and following an evolutionary and incremental path of development is the most appropriate given the current rate of technical development.

A critical goal for the Alexandria Project is the design and implementation of UI's that support general DL functionality and a catalog component that supports the full range of user queries entered into the UI. An important subgoal relating to the UI is the provision of library services to users with a variety of backgrounds and needs. A school child, for example, may wish to use ADL to find a map that shows nearby rivers and trails for a camping trip; a scientist may wish to find datasets of elevation and rainfall for a study area in order to develop a vegetation model. Different users need different ways to formulate search queries and have different expectations of, and requirements for, search results.

An initial, and now completed, increment in the development of ADL involved the design and construction of a stand-alone "rapid prototype" (RP) system [5] [3]. The second increment involves providing an augmented version of the functionality of the RP over the World-Wide-Web (WWW). We term this the "WWW prototype" (WP) [4]. The third increment will focus on developing a catalog component that is based on a general model of metadata.

In the current paper, we focus on the resolution of issues that arise both in expanding the functionality of the RP and in extending it to the WWW. The paper is structured as follows. We first describe the basic four component architecture of ADL, as adapted for the WP version. We next describe our resolution of key issues that arise for these various components in a WWW en-

[1] Work reported in this paper is supported by funding from NSF, ARPA, and NASA under NSF IRI94-11330 and by product contributions from the Digital Equipment Corporation.

[2] The ADL Project involves a consortium of universities, public institutions, and private corporations headed by the University of California at Santa Barbara and jointly sponsored under the Digital Libraries Initiative of NSF, ARPA, and NASA.

[3] A limited number of RPCD's are available.

[4] The WP will be available to users with access to WWW by late 1995.

vironment. Finally, we describe the application of two key technologies that have implications for each of the components, namely *wavelet transformations* and *parallel processing* [5].

2 Architecture of WP

In Fig. 1, we illustrate the four main components of the architecture for the WP. This architecture is derived from the general architecture of ADL by specifying the languages and protocols at the interfaces between components. A few important aspects of this architecture include: 1) in the *storage component*, the use of "handles" and Internet protocols for transfer; 2) in the *catalog component*, an extensible metadata model and SQL/Z39.50 query engines; 3) in the *interface component*, the use of HTTP/HTML, browsers, and external viewers for "vector" data; 4) for the *ingest component*, the creation of catalog entries. Unlike the RP version of ADL, the storage and catalog components are distributed.

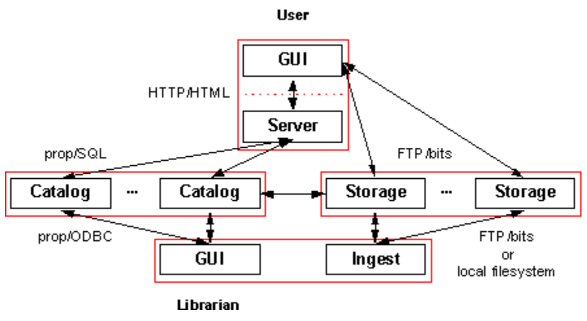


Figure 1: The main components of the WP.

3 The Storage Component

The storage component of ADL contains the collection of *digital objects* (DOBJ's). Apart from physical support for storage of DOBJ's, key aspects of DOBJ's are the binary representation of the information of interest ("data"); a universal object identifier (*oid*); and procedures for interpreting/retrieving the data. In relation to the WP, we focus largely on the issue of *oid*'s, although we discuss the ADL collections and their physical storage.

The initial collections of ADL (i.e. the "data" of the DOBJ's) are focused on spatially-referenced materials such as digitized maps, digitized air photos, and images from many domains of application [5]. Such items are typically characterized by a *spatial footprint*, which is a set of points characterizing the spatial extent of the item in the space over which the item is defined [6]. While the set of points comprising a footprint may be of arbitrary shape, it is frequently represented by a polygon (especially a rectangle) or by a union of non-intersecting polygons [7]. We note that many "features" represented in maps and images, such as *towns* and *valleys*, also have well-defined footprints.

We are extending the basic collections to include textual materials that reference objects possessing spatial footprints.

3.1 Storage Issues

The DOBJ's in ADL are typically very large. For satellite images, a size of 150 MB is not uncommon and may exceed 2 GB. The collections are also large in size. The UCSB Map and Imagery Library has, for example, a collection of over two million air photos in analog form which, when scanned at 600dpi, require between 25MB and 100MB of storage space, resulting in a collection whose size exceeds 100 TB. Finally, there are a number of reasons why collections of such items must inevitably be distributed.

Three issues arising from these considerations include the difficulties faced by users in finding an appropriate item; the cost of examining or downloading large items over bandwidth-limited channels; and the provision of access to distributed sets of storage locations. The last issue is also related to questions of interoperability between DLs.

An important issue in distributed Internet applications is that there is currently no accepted standard for *oid*'s. There are a number of alternative suggestions relating, for example, to URx's of various forms (where the "x" is an identifier, locator, or name) or handles [6].

3.2 Storage Implementation

ADL is employing distributed hierarchical storage to store its own collections. The storage and management of the data is simplified by the fact that it is predominantly "write-once". Initial storage facilities are small, comprising 150 GBytes of disk storage, both magnetic and optical, at UCSB and approximately 500 GBytes of robotic tape storage at San Diego Supercomputer Center. Other storage sites are currently being added, while access is being provided to collections that are not owned by ADL.

The issue of delivering large items to users is partly resolved with the use of a wavelet-based progressive delivery of items (see below). In cases in which a user does not require information at all levels of resolution in an image, the user need only transmit information down to the levels of resolution required.

We are employing an initial resolution of the issue of *oid*'s for DOBJ's, based on suggestions of [7], in a test of interoperability between ADL and the UC Berkeley DL [8]. Since a goal of the interoperability test is to operate without knowing

[5] Owing to restrictions in space, there are many important aspects of ADL that cannot be discussed here, such as the issues of security, privacy, and copyright.

[6] We use the term *geographic footprint* if the space over which the item is defined is the surface of the Earth.

[7] A point may be viewed as a degenerate class of footprint.

[8] We note that the issue of object registration will prob-

the details of each other's storage model, storage interoperability involves the implementation of a server that accepts the handles and returns objects. In this model, a DOBJ is a data structure whose main components are *data* and key metadata. The essential component of the key metadata is a globally unique identifier known as a *handle*. Handles are provided by authorized handle generators and, if the handle of a DOBJ is made known to a system of handle servers, the *registered* DOBJ may be stored in a repository. Repositories are also named by a global naming authority and possess mechanisms for adding new DOBJ's to their collections and for making the objects available with the use of a *repository access protocol*. Hence, to retrieve a registered DOBJ a user must, at the very least, present a handle to a handle server to learn the network names or addresses of repositories in which the corresponding DOBJ is stored.

4 The Catalog Component

The catalog component of ADL permits users to make a mapping from their requirements for information to the most appropriate set of information accessible from the total collection of items. The standard cataloging system of an analog library, including the usual author and subject catalogs, provides an initial model for the catalog component of DLs. It is necessary, however, to generalize this model for DLs, such as ADL, that contain spatially-indexed materials.

The catalog component of the RP incorporated a generalization based on a new metadata schema for spatially-indexed information that combines the Federal Geographic Data Committee (FGDC) and USMARC standards [5] [9]. The design and implementation of the WP catalog represents a further step in this process of generalization in which we introduce metadata that supports new classes of content-based queries.

4.1 Catalog Issues

Important issues concerning the catalog component of the WP include: 1) extending the classes of content-based queries supportable by ADL; 2) extending the metadata in the catalog component and the associated search procedures to support the extended classes of queries; 3) user access to the catalog system; 4) support for a distributed catalog.

An important class of queries in a library of spatially-indexed items involves requests for items of a given class in terms of the footprint of the item. In the RP, such queries were answered by intersecting a query footprint (entered into a map browser) with footprints of collection items (stored in metadata records). A significant extension to this class of queries supported in WP involves the search for items in which specific features, with or without their own spatial footprint, are represented. We also support the inverse query in which the user wishes to know which features are represented in specific items.

4.2 A General Model of Metadata

Extending the class of queries answerable in ADL generally entails extending the metadatabase. While the traditional library community has established well-defined sets of terms and procedures relating to cataloging issues [1], there is no generally accepted conceptualization of metadata for DL's. Hence the current design of metadatabases and catalogs for DL's is essentially an *ad hoc* procedure. This is a critical issue because digital technology is providing tools that are relaxing the limits on the extraction, storage, maintenance, and search of metadata. It is likely that metadatabases in the catalogs of DL's will grow to be extremely large and that there will be a blurring of the distinction between items in the "main" collection of a library and representations of these items in the metadata.

One strategy for avoiding an *ad hoc* growth in the metadata and catalogs of DL's, is to develop a general model of metadata that accommodates likely extensions to catalog information. We therefore provide a sketch of a simple model of metadata for the ADL catalog component [10]. We believe that this model provides sufficient generality to cover many of the conceivable extensions to the metadatabase and the catalog that are supportable with digital technology, while still providing a useful conceptual tool for designing and implementing these components in an evolving DL.

We view the assignment of "meaning" to a DOBJ as being provided by a set of *interpretive mappings* or *interpretations* that assign to the handles of the DOBJ information characterizing any aspect of the DOBJ that may be of value to a user. The information may or may not be inferable from the data of the DOBJ. Examples of such information for the case of a document include an HTML representations of the document, textual representations of the lineage of the document (which may not be inferable from the document itself), symbolic representations of entities existing in the real world that are referred to in the document, and a "reduced" graphical representation of the illustrations represented in the document.

We distinguish two main classes of such interpretive mappings: 1) *formal* interpretations that take as input the handle of a DOBJ and return some "displayable" representation (e.g. *ascii, html, sgml, postscript*) of the "data" associated with the DOBJ; 2) *informal* interpretations that *either* take as input the output of some formal interpretation and return representations of charac-

ably not be resolved by the DL community alone, particularly in the case of spatial data.

[9] USMARC now contains the full FGDC standard.

[10] A more detailed discussion of this model is currently in preparation.

teristics of the item (e.g. the creator of an image or the edges computable from the image) *or* that map the output of some other informal interpretation into the representations of other entities (e.g. a set of edges into faces). Formal interpretations are invertible mappings from the data of the DOBJ, while informal interpretations are generally neither invertible nor computable solely from the data of the DOBJ. One may construct an unlimited number of informal interpretations, and it may be possible to form compositions of such mappings.

As noted by many other researchers (see, for example, [3]), it is useful to distinguish *content-independent* and *content-dependent* informal interpretations. Examples of the former include the *author catalog* and more general "lineage" information about items. While it is also useful to define many other subclasses of metadata, for purposes of exposition we only discuss a classification of content-dependent interpretations into *domain-independent* and *domain-dependent* interpretations. Examples of domain-independent interpretations include the *LoC subject headings* and general classes of image processing procedures that extract domain independent features such as the sizes, colors, shapes, and textures of regions from gray scale images. Examples of the domain-dependent interpretations include footprints of library items, gazetteers, and representations of such features as trees or valleys.

We define the *metadata* of a catalog to be the total set of instantiated interpretations of DOBJ's. It is sometimes possible to view such metadata in terms of a set of *metadata records* that each contains the handle of a DOBJ and a set of "fields". The fields contain representations of the output of all valid compositions of interpretative mappings applied to that DOBJ.

We argue, however that it is more useful to view metadata as a set of mappings rather than a set of "records". At one extreme, the application of all compositions of interpretations to all DOBJ's would generally require impossible amounts of precomputation. It would also lead to the need for continual record updates as new interpretive mappings are defined (i.e. as the catalog is extended). At the other extreme, the application of all interpretations at search time is also computationally infeasible. Some combination of precomputed interpretations and interpretations computed on the fly will generally be optimal for a given set of interpretations, as well as permitting the easy addition of new interpretations. The concept of interpretive mappings therefore provides a uniform conceptualization of metadata.

It is straightforward to conceptualize retrieval in terms of this model. For *simple* cases of retrieval based entirely on a precomputed set of interpretations 1) a user query is translated into a *query record*, which takes the form of a metadata record but without a DOBJ handle, with the query record employing only those representations used to characterize the output of interpretive mappings; 2) the query record is matched against precomputed metadata records and additional interpretations are computed as required; 3) an appropriate set of DOBJ handles is returned; 4) additional transformations are applied to the retrieved set of DOBJ's in order to extract the requested information.

In general, this procedure is likely to be complex. First, additional informal interpretations may be computed as required. These may take the form, for example, of image processing procedures that are applicable by the user or the form of a *browse process* in which users apply the informal interpretations that arise in their visual and cognitive processing of items. Second, it is generally necessary to match query records and metadata records based on *similarity values*, such that a *set* of DOBJ's will be returned. Third, the search for appropriate metadata records must be aided by the application of indexing strategies.

The metadata and retrieval process of the Alexandria RP is clearly a special case of the preceding model. The metadata corresponding to a single DOBJ may be viewed as taking the form of a single "record" in some "relation", whose "field" values are drawn from USMARC (incorporating FGDC) standards. In terms of the preceding model, we may view this as a set of precomputed informal interpretations of various classes, including *originator, publication* (content-independent), *theme keywords* (domain-independent, content-dependent), and *bounding coordinates* (domain-dependent, content-dependent.)

During search in the RP, the user's query is essentially translated into a query record involving a relatively small number of key pieces of information such as: the "theme" of the item requested, the spatial footprint of the item, and the "time" at which the item was created. This query record is then matched against the metadata records. In particular, the class and time values are matched exactly, while a truth value is determined for an intersection predicate defined on the footprint values in the query and metadata records. A formal interpretation of the DOBJ's is displayed to the user, who may then request delivery of the data in the DOBJ or the application of transformations to this data.

4.3 Catalog Implementation

The WP metadata model incorporates two extensions to the RP metadata model, including the addition of interpretations involving: 1) image "texture"; 2) features found in maps and images. The first extension permits searching for images containing textures matching either preselected sets of textures or specific images selected by the user. We discuss this case further in the section on wavelet transformations.

The second extension permits retrieving ADL holdings based on overlaps between the footprints of collection items and footprints of named instances of various classes of features, such towns or rivers. The WP employs precomputed representations of footprints that may be found in available *gazetteers* [11]. A gazetteer is basically a list of feature classes (sometimes hierarchically organized), a set of their named instances, and a footprint of each instance [12].

We are using both the Geographic Names Information System (GNIS) digital gazetteer of US features and the Board of Geographic Names (BGN) digital gazetteer of worldwide features. The GNIS gazetteer contains about 1.8M names of US features, organized hierarchically into 15 classes of features, while the BGN gazetteer contains approximately 4.5M names of land and undersea features. Specific issues that we are addressing include: 1) ingesting non-digital gazetteers (e.g. historic place names); 2) merging gazetteers of different feature classes, formats, and accuracies; 3) constructing meaningful footprints for entities; 4) organizing the feature classes into meaningful hierarchies.

The issue of footprints is the most troubling since those of existing gazetteers are often point locations, rather than sets of points that define an area. When a gazetteer employs a single point to represent the footprint of a feature possessing non-zero extent, it is not always clear how the points were chosen. For example, they may be centroids, corners, or arbitrary points.

In constructing a gazetteer for the WP, we are having to choose appropriate footprints for features and to extract these footprints. The definition of an appropriate footprint for many classes of features is difficult. One must in general decide whether a single point, a simple polygon, a complex polygon, or some other representation is the most appropriate. Furthermore, there may be no unambiguous definition of the footprint of some feature (where does a mountain begin and end?). This ambiguity and fuzziness is inherent in a person's notion of the spatial extent of a feature, and is particularly difficult to specify. Finally, extracting and entering footprints into a gazetteer is expensive. Footprints may be generated manually, or from existing digital data, or from other ancillary information. An associated problem is finding and correcting errors in existing gazetteers.

Database support for the gazetteer information is currently provided by the ConQuest text-retrieval engine. A significant feature of ConQuest is its ability to handle fuzziness in the feature specifications. While the WP metadata are currently stored in the Sybase RDBMS, we are also implementing our metadata in the O2 object database, since many of the range values in informal interpretive mappings of interest are best represented as structured objects.

As the size of the metadatabase grows, it is critical to provide efficient support for different types of queries over the footprints with the use of appropriate spatial indexing methods. We have been exploring new methods of indexing multiply-nested spatial data [9] such as footprints. In particular, we have extended B-trees to "IB-trees" for handling data objects that span a range of values (intervals) rather than single-valued points in the data space. This allows two distinct approaches for indexing multidimensional hierarchical data. The first decomposes the d-dimensional data objects into d intervals, one per dimension, and indexes the intervals in each dimension separately. The second organizes all data objects at the same level, using standard spatial indexing methods.

4.4 User Access to the Catalog

Data described within the ADL domain are made accessible via HTTP and Z39.50 servers. Primary access to ADL will be from WWW browsers connecting to the Alexandria Web server. Z39.50 clients will also be able to contact the Alexandria Z39.50 service. Under the direction of the FGDC, federal and state governments are setting up servers that support a well-known set of metadata elements from the FGDC metadata standard. Through these standard data elements, broadcast queries can be made from the ADL server, using Z39.50, to other servers and a composite set of relevant documents from many sites can be built and served to a client.

In addition to relying on Z39.50 to access other servers we envision that existing Z39.50 clients, primarily on-line systems in the library community, will benefit from gaining direct access to ADL's metadata and data stores. Z39.50's standard queries, presentation fields, and search and retrieve protocols provide time-tested interoperability elements that we cannot ignore.

5 The User Interface

The WP UI is currently intended to provide easy access to a core set of functionality for a heterogeneous user population. This core functionality is focused on 1) composing spatial search queries; 2) displaying spatially-indexed materials, in both raster and vector format; 3) browsing search results; 4) allowing user-configurable defaults and options; 5) efficiently retrieving data items in various native formats.

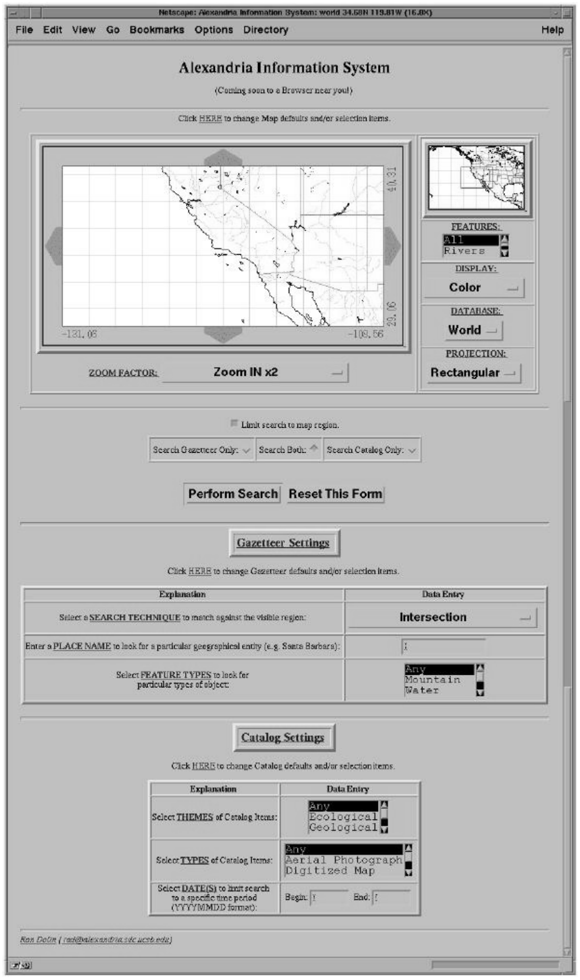
The RP supported the first three of these functions, using an implementation based on the GIS software package Arcview [4]. In adapting the system to the WWW environment, however, and in augmenting the functionality of the RP, a number of significant issues arise.

5.1 User Interface Issues

Many of the mechanisms employed in the RP interface are not well supported in the (currently available) standard WWW/HTTP/HTML envi-

[11] The WP does not yet support automated procedures that extract the footprints of features from maps or images.

[12] A footprint in a gazetteer is frequently a single point represented in terms of a latitude/longitude pair.



Alexandria Information System

(Coming soon to a Browser near you!)

Click HERE to change Map defaults and/or selection items.

Limit search to map region.

Search Gazetteer Only: Search Both: Search Catalog Only:

Click HERE to change Gazetteer defaults and/or selection items.

Explanation	Data Entry
Select a SEARCH TECHNIQUE to match against the visible region:	Intersection
Enter a PLACE NAME to look for a particular geographical entity (e.g. Santa Barbara):	
Select FEATURE TYPES to look for particular types of object:	Any Mountain Water

Click HERE to change Catalog defaults and/or selection items.

Explanation	Data Entry
Select THEMES of Catalog Items:	Any Ecological Geological
Select TYPES of Catalog Items:	Any Aerial Photograph Digitized Map
Select DATE(S) to limit search to a specific time period (YYYYMMDD format):	Begin: End:

Ron Dolin (rad@alexandria.sdc.ucsb.edu)

Figure 2: Web user interface "mockup".

ronment. In particular, the ADL WP must operate within the following WWW limitations: 1) current implementations of HTML lack mechanisms for presenting vector data, and provide weak support for entering spatially-indexed information; 2) HTTP is stateless, and is designed for small, fast transactions; 3) current Web browsers do not allow a sufficiently high degree of interactivity. Helper applications are only a short-term substitute for better browser-helper communications and/or programmable browsers.

The limitations of HTML are apparent when attempting to support a user in defining a spatial search region. A natural procedure for such a specification is to draw a polygon on a *base map* by either clicking on multiple points or clicking and dragging over a desired region. Such actions are not currently supported by Web browsers, which immediately send an HTTP request after a single mouse click.

The stateless nature of HTTP makes browsing difficult. By default, after a user completes an HTTP request from a WWW client to an HTTP server, neither the client nor the server maintains any state or "memory" of the transaction (other than perhaps logging the URL involved). Each request appears to the server to be completely new. This statelessness prevents needed activities such as a user-defined configuration, environment parameters, a query history, and iteratively refined searches. In order to support these activities, information must be kept at either the client site (through parameters stored in the URL and "hidden" form variables) or at the server site (through unique user identifiers and a session database).

Finally, no WWW browser that we know of supports vector data display. This is a serious issue, since a significant and important portion of spatially-indexed information collections involve items represented in vector format.

In addition to the three major UI functionalities supported in the RP, the UI of the WP is designed to be user-configurable, and to permit delivering data to a remote user. Most WWW information systems have a simple static UI and a static representation for retrieved information. A static UI is insufficient for systems that must provide a wide range of services to heterogeneous library user communities. To provide various services to different levels of users, a UI should be user-customizable and provide a mechanism for saving tailorable options for future sessions. Additionally a user must be able to retrieve a particular data item or metadata record. Since the WWW is part of the Internet, simple retrieval via FTP is straightforward to implement. As noted above, however, the items in ADL are often extremely large, so methods that allow for the extraction and transfer of smaller pieces of a data item (portions of images or lower resolution versions for initial browsing) are needed. Such a requirement is in part supported by the use of wavelet transformations, as described below.

5.2 User Interface Implementation

We briefly discuss our design for the ADL WP interface, focusing on the issues raised above. The major part of the WP UI is based on an HTML form, as shown in Fig. 2, which contains sections for interacting with a map browser, a catalog, and a gazetteer, as well as a set of control/configuration and help/glossary links. Rather than basing this design for the UI on a particular set of user scenarios, it was designed around a state transition model. Each state represents a Web form or page, some of which include partial or complete query results.

The preceding model supports standard methods for querying the catalog, such as starting with the base map or gazetteer to select an area and then querying the catalog for information within this area. However, other paths through the state model extend the functionality of the library to novel uses. For example, a user could bypass the base map and gazetteer, leaving the default area defined as the entire world, and query the catalog for air photos from 1960 to 1970. Then, based on the resulting footprints denoting the catalog entries, the user could select the region of the world corresponding to the highest density of air photos within this time interval. This allows the user to select geographical areas of interest based on the library store, rather than always starting with a given geographical area. Although past usage patterns (observed at existing map libraries) generally start with an area specification, it is difficult to know if this is due to the non-digital interface through which most geographical data are now acquired.

Within the map browser section of the main form, each base map image is dynamically generated by a Common Gateway Interface (CGI) application based on user requests. We currently use a modified version of the Xerox PARC Map Viewer [http://www.parc.xerox.com/map/]. The Map Viewer supports zooming, panning, and feature selections. We have added several new functions to the Map Viewer, including generic labeling, fast panning, and overlaying graphics objects on a map. The region of the base map which is visible in the map browser is called the *display window*. As explained below, this window is not necessarily the same as a *query window*, which consists of the Earth coordinate boundaries used to limit a query. The base map also serves as the background on which we overlay footprints of the results of various queries. The catalog section allows users to enter thematic and temporal search terms for the display and browsing of objects that are in the ADL catalog.

The UI supports the primary function of the base map and gazetteer, which is to allow the user to define spatial extents or regions for catalog searches. The user can either zoom and pan the base map to an area of interest or submit a gazetteer query with a given place name and feature type. For example, a gazetteer query with the place name set to "Ohio" and the feature type set to "hydrography" will yield a query window whose Earth coordinates bound the Ohio River, but not the state of Ohio.

By default, a gazetteer query is spatially bounded by the current display window. Thus, if the current display window shows the USA and does not contain Europe, a gazetteer query with the place name set to "Paris" will yield a query window around Paris, Texas and will not include Paris, France. The display window will then be changed to show an area slightly beyond the query window around Paris, Texas. If the result of a gazetteer query is several unconnected regions (all of which, by default, must lie within the original display window), we contract the display window to show the smallest bounding box that encompasses all query windows returned by the gazetteer. So, for example, if the display window is set to the entire world and one queries the gazetteer with "Paris", both Paris, Texas and Paris, France are returned, and our display window will be a fairly large portion of the world map, as described below.

Catalog queries are bound by the query window(s) shown in the display window, rather than by the entire display window as a whole. Since our UI allows for automatically piping the query windows resulting from the gazetteer search directly through to the catalog search, we do not, by default, enlarge the query window to include the entire display window. Using the above example, one would probably not want to include footprints of satellite images of Maine after asking the gazetteer for "Paris". So instead we allow for several query windows within a single display window, and thus pass to the catalog a separate query for each bounding region within the display window which was returned by the gazetteer.

After a user submits a query to the ADL query engine, the query engine converts the user query to different catalog-dependent internal query syntax (e.g. SQL, ConQuest, and Z39.50) and performs the search on local and remote catalogs. The query engine returns the search results to the UI for display as an HTML table. The format and content of this table is also user configurable. Users can then select a subset of items returned by the query to get the full metadata documentation, to project selected footprints onto the base map, to view "browse" or "thumbnail" images (see below), or to down-load data objects.

The search result set contains a set of metadata information for each data item (library holding) that matches the user query. The metadata contains both text and images. In particular, two types of metadata, footprints and GIF browse graphics, can be displayed as images in an HTML document. A browse graphic is a low-resolution image of a data item, similar to the abstract of a textual article.

Frequently many more search results are returned from a catalog query than can be shown intelligibly on a relatively small display map. When footprints of multiple data items are displayed on the same map, it is difficult to distinguish which footprint is associated with which item. We are currently experimenting with many heuristic approaches and visual aids (such as clustering and labeling) to find a reasonable way of displaying footprints.

In particular, a heuristic is required for deciding which footprints returned by a query are to be displayed on the base map. This selection criteria must be configurable by the user, who may want the decision to be based on chronological order, total area covered, or other attributes. Our current default heuristic is based on the footprint weighting $w = p_1 \times p_2$, where p_1 is the percentage of the query window that is covered by the footprint and p_2 is the percentage of the footprint that is contained in the query window. Thus, if the query window is defined by, say, California, less weight is given to world maps ($w = 1.0 \times \sim 0.001 =\sim 0.001$) and maps of Santa Barbara ($w =\sim 0.001 \times 1.0 =\sim 0.001$) than is given to, say, a map of the California/Nevada area ($w = 1.0 \times \sim 0.5 = \sim 0.5$).

Some of the "best" footprints are often those that lie just outside, but very close to, the query window. Thus within the UI and corresponding CGI's, we maintain two sets of bounding coordinates. The first set defines the display window, and the second set keeps track of one or more query windows, as previously discussed with the "Paris" example. Once we know the coordinates of the query windows returned by the gazetteer, the dimensions of the display window are calculated such that approximately 75% of the visible area envelopes the query windows. This allows footprints that contain the query windows to be displayed in the display window. The user can change the ratio of the two areas.

The query form is dynamically generated based on user preferences and selections. The initial UI presents a set of default query forms and parameter settings suitable for the general user community. Experienced users can click on a configuration button to invoke the CGI script "Query Form Generator" (QFG). The QFG creates a query form configuration page with which users can customize the query form to fit their particular needs. The QFG lets a user choose a set of attributes, logical operators, and heuristics to be used within the query form.

The Alexandria WWW system maintains all user configuration parameters, query statements, and current query result sets with a combination of client side and server side state parameters. On the client side, "hidden" form variables within the HTML document and the actual URL are used to keep track of state information. Part of the URL contains a "cookie" which is a unique session identifier. Thus every request coming from a particular user is associated with session information stored in a server-side session database. Upon request, this session information can be written out as hidden variables to an HTML document, allowing a user to save all customization and history parameters to a local file. This page can be opened later to return a user to their previous state. Keeping limited session information allows a user to re-use their customized query/display environment without reconfiguration, and browse through a result set without a redundant database search. Maintaining user state also allows progressive delivery of large data objects. In particular, network traffic is reduced by displaying wavelet browse images and using multi-step image enhancement.

Displaying vector or wavelet data is another important issue in our implementation. It is unlikely that any existing publicly-available Web browser will support vector or wavelet data types in the near future. This problem can be solved by helper-applications (either stand-alone or with NCSA Mosaic Common Client Interface (CCI)), or by programmable browsers such as HotJava. Due to HotJava's current availability only on SUN Solaris and Windows NT, we have decided to develop helper-applications for vector and wavelet data display. The helper-application approach creates a minor distribution problem and a major porting and support problem, since every ADL client site must install our programs in order to view vector and wavelet data contained in the ADL catalog. We are therefore tracking the availability of systems such as HotJava.

6 The Ingest Component

The ingest activities in the RP focused on 1) modeling the metadata schema for ADL; 2) designing the ingest interface to the Sybase RDBMS, using Tcl/Tk; 3) entering records into the RP database, using 80 metadata fields.

The ingest activities in the WP are similar, with the following additions: 1) extending the metadata schema to include *all* FGDC fields, plus as many additional USMARC fields as are needed; 2) ingesting existing metadata sets, including approximately 450K frame-level records for a NASA air photo database, approximately 350K sheet-level records for map series (Geodex), and approximately 100K USMARC map records from MELVYL (the University of California online union catalog); 3) creating new catalog records for selected items, including Web sites for digital spatial data, and air photos for four local California counties; 4) moving the ingest interface from Sybase to Microsoft Access.

7 Image Processing Issues

Image-processing has implications for each of the four components of ADL. In particular, 1) for storage, wavelet transforms support hierarchical storage management, with fast retrieval appropriate for low-resolution components, and slower retrieval appropriate for high-resolution components; 2) for the catalog, metadata in the form of texture "badges", extracted using Gabor filtering of the images, provide additional indexing methods; 3) for the interface, wavelet transformations support the delivery of holdings at appropriate resolutions, with rapid delivery of low-resolution ("browse") images, as well as image-match search; 4) for the ingest component, image pre-processing permits automatic extraction of some metadata, such as texture badges.

7.1 Wavelets and Efficient Access

Even with access to very high speed networks, it is often impractical to transmit a large image as a single item, particularly if the user is in a browsing mode, trying to find items of interest. A simple solution to this problem is to maintain for each large image, a low-resolution (e.g., subsampled) "thumbnail" image for browsing purposes. While thumbnails consume storage space (or processing cycles, if generated on-demand), this overhead is typically insignificant compared to the advantages from their use. If the user finds a thumbnail of interest, additional high-resolution data may be downloaded.

It is clear from our experience with the RP and various other GIS systems that we should make thumbnails available as well as the original data. However, this only addresses the issue of browsing quickly through large numbers of images. In an interactive database system, users are likely to do much more than make binary "go/no-go" decisions based on simple thumbnail images. They may, for example, wish to zoom in on a given region. Such operations typically cannot be supported with low-resolution thumbnail images. Furthermore, different groups of users may have different requirements. A school teacher using a LANDSAT image for a certain demonstration may not need the same high-resolution image as a scientist trying to classify land cover types. The general solution is to have access to hierarchical, multiscale representations of image data.

An obvious solution to this requirement is the use of wavelet transforms, which provide multiscale decompositions of the image data [11]. Wavelets have been widely used in many image-processing applications, including compression, enhancement, reconstruction, and image analysis. Fast algorithms exist for computing the forward and inverse wavelet transforms, and desired intermediate levels can be easily reconstructed. The transformed images (wavelet coefficients) also map naturally into hierarchical storage structures. We can assume that low-resolution data are accessed more frequently than the finer information, and hence should be stored in faster devices for efficient browsing, while higher-resolution data may be placed in tertiary storage. A useful property of the decomposition is that the lowest-resolution, low-pass filtered and subsampled image (a by-product of the wavelet transform) may be used as a thumbnail for browsing.

Important issues related to wavelet-based storage include the choice of decompositions (i.e. choice of filters) that are appropriate for the different image databases. Image compression is important in storing large amounts of data. Many GIS and medical imaging applications often require lossless compression and this continues to be an active research problem in image-processing. Although the total number of wavelet coefficients equals the number of pixels in the images, their storage requirements differ. The original intensity data, in most cases, consists only of integer numbers. Wavelet coefficients are real numbers, thus requiring more memory. Even for the case of no compression, these coefficients need to be quantized and encoded appropriately to ensure that they do not take more space than the original image data. How to quantize these coefficients without losing near perfect reconstruction is an important research issue.

7.2 Content-based Retrieval

Research on content-based retrieval in image data bases has focussed on searching image properties such as color, texture, histogram, and shape. We have made considerable progress in developing algorithms for texture-based search [10]. We are currently investigating augmenting the ADL catalog with indices based on texture features.

The basic idea is to extract texture information from the images as they are ingested. This is done using Gabor filters, which are modulated Gaussians. Processing through a bank of these Gabor filters is approximately equivalent to extracting line edges and bars in the images, at different scales and orientations. Simple statistical moments, such as the mean and standard deviation of the filtered outputs, can then be used as indices to search the database. Figure 3 shows an application to browsing large air photos.

The figure shows a downsampled (reduced-resolution) version of the image on the left, and the retrieval results using the query pattern in the first column. The query pattern was selected from the region containing some buildings in the left center of the air photo. The retrieved subimages, ordered according to a similarity measure (top to bottom and left to right in the last two columns), are all from the same region.

For the WP, we plan to create a database of air photos which can be searched using texture templates. Texture information will be extracted when the images are ingested. A small set of texture templates will then be created which represent the different textures that may occur in the air photos. A user initiating a search can chose an image region (by pointing the cursor) and the region's texture will be used to retrieve matching texture templates. Each of these texture templates will have pointers to the air photos where they occur.

Instead of Gabor filters, one may also use the same orthogonal wavelet transform that was used for storing the image data. However, experiments on a large set of textured images have shown that the retrieval performance of conventional orthogonal wavelets is not as good as that of Gabor filters [11]. Unfortunately, Gabor transforms are awkward for storage applications. In particular, they do not form an orthogonal basis set. Many researchers have used Gabor transforms for image compression, mainly for lossy compression. However, no efficient algorithms exist for computing the forward and inverse transformations, which is important in a digital library context. While data ingest is off-line and can be computationally intensive, data retrieval should be both fast and performed in real time using existing hardware. Orthogonal wavelets are good for such implementations, whereas non-orthogonal Gabor wavelets are good for image analysis.

7.3 Wavelet Applications within the WP

Since current WWW browsers cannot display wavelet data directly, our short-term WP plan is to develop a helper-application, most likely communicating with a browser via CCI. This will allow the helper-application to communicate fur-

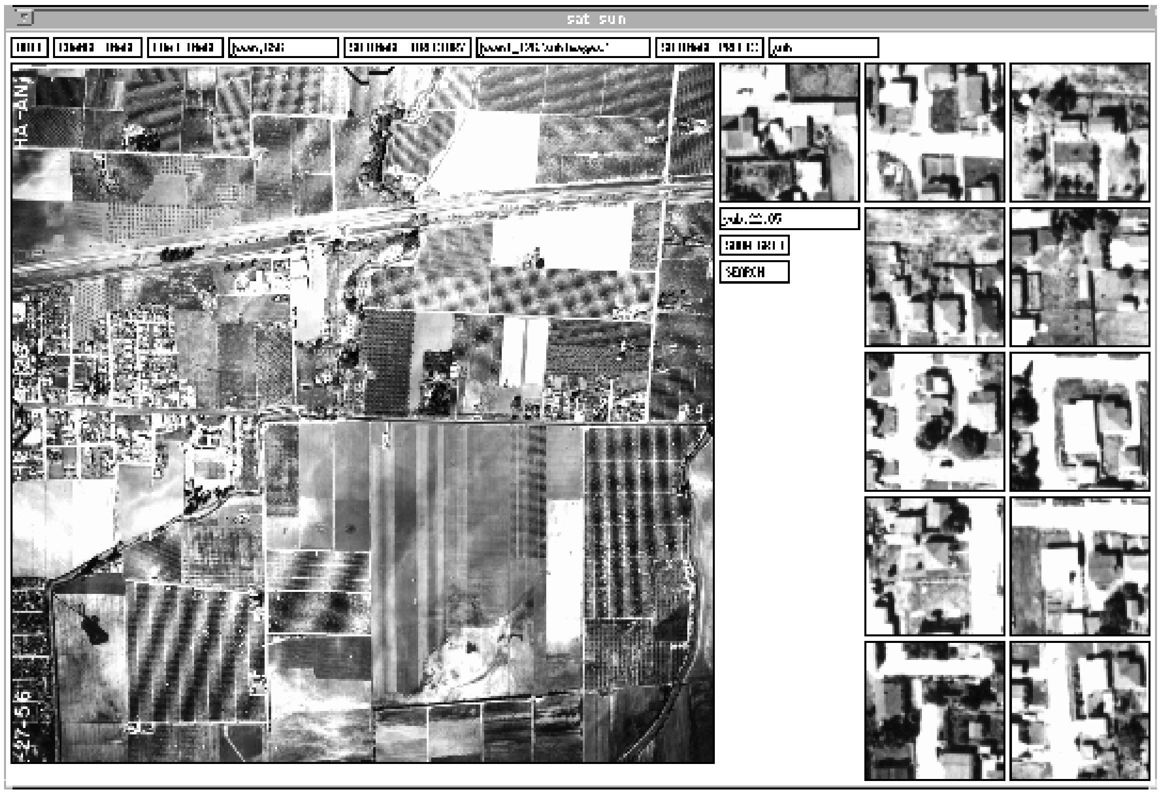


Figure 3: Browsing a large air photo using Gabor texture features.

ther image enhancement requests back to the browser. The browser can then relay that request (in the context of its session state) back to the image (HTTP) server, receive the update information, and pass it back to the helper-application, which then incorporates the new data into its image. While this is not as general a solution as having the application run directly in the browser, it allows us to make performance modifications to the helper-application which might not be possible in an interpreted language like HotJava.

8 Parallel Processing Support

Performance issues are critical to the success of ADL. Performance bottlenecks include both network bandwidth and server processing capability. While we expect that network communication technology will improve steadily, particularly with the advent of ATM and BISDN, improvements in the performance of Internet servers will have to match huge increases in expected access requests. For example, in 1993 alone, the weekly access rate for NCSA's HTTP server at UIUC increased from 91K to 1.5M. The peak request arrival rate at NCSA exceeds 20 requests/second, while current high-end workstations can handle only about 4 requests/second [8]. The requests to the NCSA server typically involve only simple file retrieval operations, whereas requests to the ADL server will involve much more intensive I/O and CPU activities with, for example, the use of run-time wavelet transforms and content-based searching on metadata and images.

The Alexandria Project, like many other projects, is investigating parallel computation [2] to address performance issues, for example, scheduling on multiprocessors and parallel I/O, parallel forward wavelet transform in image ingest, and parallel reverse wavelet transform for efficient browsing of multi-scale images. Although the sequential wavelet transform has a complexity proportional to the size of the image, the computing time is still slow for a large image (e.g. 2 minutes in a SPARC-10 workstation for a 1024x1024 image) and parallelization of wavelet transforms is beneficial. The main research is the efficient compression and retrieval of wavelet coefficients and image reconstruction with the support of parallel I/O. In this paper, we focus on improvements in the HTTP server performance.

A goal of the Alexandria Project is to develop high performance DL servers based on clusters of workstations, parallel machines, and parallel I/O facilities. The system we are developing includes networked Sun and DEC workstations and a Meiko CS-2 (distributed memory machine). As depicted in Fig. 4, the main component of the system is a distributed scheduler that executes information retrieval and processing operations in response to HTTP requests.

We have developed the preliminary version of SWEB, a parallel HTTP server which contains a set of collaborative processing units, each of which is capable of handling a user request. The distinguishing feature of SWEB is resource opti-

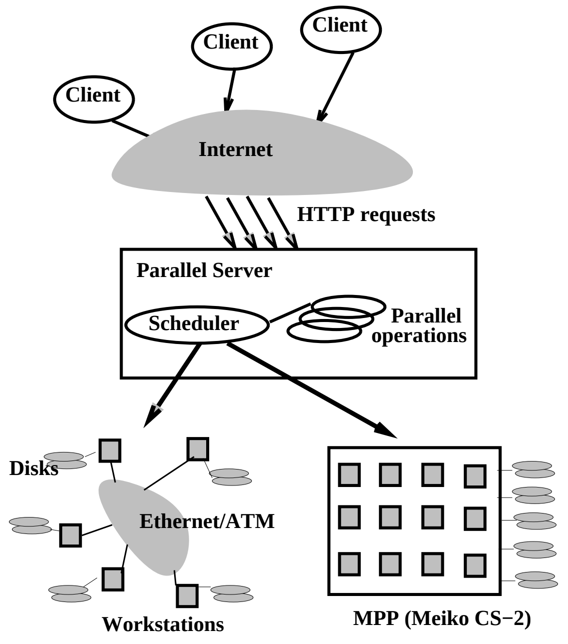


Figure 4: The parallel server.

mization by close collaboration of multiple processing units. Each processing unit is a workstation (e.g. SUN SPARC or a Meiko CS-2 node) linked to a local disk. The disks are NFS-mounted to all processing units. Resource constraints affecting the performance of the server are: CPU speed and memory size of one processing unit; the background load imposed by non-SWEB processes; I/O bandwidth between the processing unit and its local disk; network latency and bandwidth between a processing unit and a remote disk, when the accessed files are not stored in the local disk; and disk contention when multiple I/O requests are accessing the same disk. Scalability of the server is achieved by actively monitoring the CPU, disk I/O, and network loads of system resource units, and then dynamically scheduling user HTTP requests to a proper workstation for efficient processing.

Our experiments indicate that SWEB provides a sustained round-trip performance of response time when the number of requests reaches 5 to 30 million per week. The following table shows the performance of SWEB on 6 Meiko CS-2 nodes, compared with single-node server performance, in accessing image files of average size 1.5MB when the number of requests varies from 2 to 20 per second. The round-trip total response time in seconds is improved significantly by using multiple processing units, and the response time does not change significantly when the number of requests per second (rps) increases. We have observed similar speedups using a multi-node server when varying the size of image files.

#proc	2 rps	4 rps	8 rps	16 rps	20 rps
1	5.1	24.3	142.5	283.7	448.4
2	3.3	4.6	29.4	87.1	112.2
4	3.2	3.7	6.6	32.2	45.4
6	3.1	3.3	3.7	12.5	21.1

NCSA [8] has built a multi-workstation HTTP server based on round-robin domain name resolution to assign requests to workstations. This technique is effective when HTTP requests return relatively uniform-sized chunks of HTML. For ADL, however, the computational and I/O demands of requests may vary dramatically because of large images and variable-sized metadata. We have compared the round-robin approach to our load-balancing approach for different file sizes and have observed a 20% to 50% improvement in performance.

References

[1] American Library Association. Anglo-American cataloguing rules. 2nd ed., rev., Chicago, 1988.

[2] D. Andresen, O. Egecioglu, O. Ibarra, A. Poulakidas, A. Srinivasan, and T. Yang. Parallel processing support for high performance digital libraries, Technical Report, CS Department, UCSB, 1995.

[3] K. Bohm and T. C. Rakow. Metadata for Multimedia documents. Sigmod Record, Vol. 23, pp. 21-26, 1994.

[4] Environmental Systems Research Institute Inc. ArcView 2.0c software, Alpha/OSF1 version. Redlands, California, 1978.

[5] C. Fischer, J.Frew, M. Larsgaard, T.R. Smith and Q.Zheng. Alexandria Digital Library: Rapid Prototype and Metadata Schema. Proceedings of ADL95 Conference, 1995.

[6] Internet Engineering Task Force. http://www.ietf.cnri.reston.va.us/ids.by.wg/uri.html, 1995.

[7] R. Kahn and R. Wilensky. http://WWW.CNRI.Reston.VA.US/home/cstr/arch/k-w.html, 1995.

[8] E.D. Katz, M. Butler and R. McGrath. A Scalable HTTP Server: the NCSA Prototype. Computer Networks and ISDN Systems, Vol 27, pp. 155-164, 1994.

[9] R. Kothuri and A. K. Singh. Indexing Hierarchical Data. Technical Report TR95-14, CS Department, UCSB, 1995.

[10] B. S. Manjunath and W. Y. Ma. Texture Features for browsing and retrieval of image data. Technical Report CIPR-TR-95-06, ECE Department, UCSB, 1995.

[11] M. Vitterli and C. Herley. Wavelets and filter banks: theory and design. IEEE Transactions on Signal Processing, Vol. 40, pp. 2207-2232, 1992.