Efficient, Automatic Web Resource Harvesting

Michael L. Nelson, Joan A. Smith and Ignacio Garcia del Campo
Old Dominion University
Computer Science Dept
Norfolk VA 23529 USA
{mln, jsmit, dgarcia}@cs.odu.edu

Herbert Van de Sompel and Xiaoming Liu
Los Alamos National Laboratory
Research Library
Los Alamos NM 87545 USA
{herbertv, liu_x}@lanl.gov

ABSTRACT

There are two problems associated with conventional web crawling techniques: a crawler cannot know if all resources at a non-trivial web site have been discovered and crawled ("the counting problem") and the human-readable format of the resources are not always suitable for machine processing ("the representation problem"). We introduce an approach that solves these two problems by implementing support for both the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) and MPEG-21 Digital Item Declaration Language (DIDL) into the web server itself. We present the Apache module "`mod_oai`", which can be used to address the counting problem by listing all valid URIs at a web server and efficiently discovering updates and additions on subsequent crawls. Our experiments indicated comparable performance for initial crawls, and dramatic increases in update speed `mod_oai` can also be used to address the representation problem by providing "preservation ready" versions of web resources aggregated with their respective forensic metadata in MPEG-21 DIDL format.

Categories and Subject Descriptors: H.3.5 Information Storage and Retrieval: Online Information Services [Web-based services]

General Terms: Performance, Design, Experimentation.

Keywords: Web Crawling, OAI-PMH, mod_oai.

1. INTRODUCTION

Much of the web's usability depends on the efficiency of search engines and their crawlers. The indexable "surface" web has grown from about 200 million pages in 1997 to over 11 billion pages in 2005 [17], and the "deep web" is estimated to be 550 times larger [8]. Considerable attention has therefore been given to increasing the efficiency and scope of web crawlers. A number of techniques to more accurately estimate web page creation and updates [32, 13] and to improve crawling strategies [14, 12] have been proposed. Techniques such as probing search engines with keyword queries and extracting the results are used to increase the scope of web crawls and obtain more of the deep web [20, 33, 35, 27, 11]. Extending the scope of a web crawl has implications on the coverage of search engines and in web preservation [18, 28].

These approaches are necessary because web servers do not have the capability to answer questions of the form "what resources do you have?" and "what resources have changed since 2004-12-27?" A number of approaches have been suggested to add update semantics to http servers, including conventions about how to store indexes as well-known URLs for crawlers [9] and a combination of indexes and http extensions [46]. WebDAV [15] provides some update semantics through http extensions, but has yet to find wide-spread adoption. The RSS syndication formats [36] are widely implemented, but they are designed to expose "new" content rather than a complete set of site resources. Some search engines, notably Google and MSN [1, 3], have begun to work with the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), but they do not provide an open-source, broadly applicable solution. Google Sitemaps, which derive from the earlier research cited above [9], are a way to XML-encode the known URLs of a web site with optional support for specifying updates and update frequencies. MSN merely states it is committed to supporting industry standard protocols, of which OAI-PMH is one. Sites with OAI-PMH servers can register with MSN's "AcademicLive" search service for enhanced content harvesting.

To address the deficiencies of these approaches, we present `mod_oai`, an Apache module that implements OAI-PMH functionality and support for complex object metadata formats directly into the Apache web server. The goal of `mod_oai` is to bring more efficient update semantics to the general web crawling community. OAI-PMH [23, 42] is the de facto standard for metadata interchange within the digital library community, in part because of its rich semantics. Packages for implementing OAI-PMH repositories for XML files have been described [19, 37], but they are focused on highly constrained scenarios, not general web content and they do not integrate directly into the web server.

2. OAI-PMH

OAI-PMH 2.0 is a low-barrier, HTTP-based protocol designed to allow incremental harvesting of XML metadata. An OAI-PMH repository (or data provider) is a network accessible server that can process the six OAI-PMH protocol requests, and respond to them as specified by the protocol document. A harvester (or service provider) is an application that issues OAI-PMH protocol requests in order to har-

WIDM'06, November 10, 2006, Arlington, Virginia, USA.

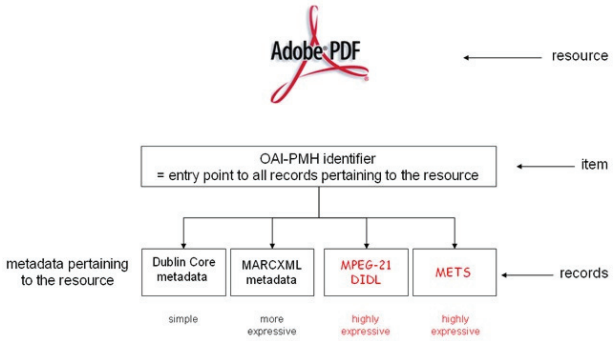


Figure 1: OAI-PMH Data Model.

vest XML formatted metadata. Scalability in OAI-PMH is achieved through building hierarchical harvesting networks with *aggregators* – services that are both a harvester and a repository [24]. A quantitative analysis of OAI-PMH is provided in [26].

The OAI-PMH is based on a simple data model primarily consisting of *resources, items and records*, as shown in Figure 1. Traditionally, the resource itself is not harvested. Instead, queries return records containing metadata - i.e., information about the resource, such as Dublin Core metadata. The entry point to all available metadata pertaining to a resource is the *item*, which is uniquely identified by an OAI-PMH *identifier*. Each item may have one or more metadata *records*, which have their own datestamps and identification type, so queries can access any combination of an item's metadata records by adding qualifiers. Another important aspect of the OAI-PMH data model is the *set*, enabling selective resource harvesting based not on a resource's location on the site, but on the resource's membership in that set.

OAI-PMH supports six simple, mnemonic, but powerful verbs or "protocol requests," as shown in Table 1. Three of the verbs are aimed at helping a harvester understand the nature of an OAI-PMH Repository - **Identify, ListMetadataFormats,** and **ListSets**. The ListSets verb can let a harvester know that a site maintains sets, and what those sets are. MIME type groupings (JPEG, PDF, etc.), and "interest group" sets ("USHistory," "animé") are typical examples of sets that a site might support. The other three protocol requests are used for the actual harvesting of XML metadata: **ListRecords** is used to harvest *records* from a repository. **ListIdentifiers** is an abbreviated form of ListRecords, retrieving only *identifiers, datestamps* and *set* information. **GetRecord** is used to retrieve an individual *record* from a repository. Required arguments specify the *identifier* and the *metadata format*.

For example, a request is issued to an OAI-PMH repository at baseURL `http://www.arxiv.org/oai2/`, to obtain metadata in Dublin Core format for all items in the set of physics that have changed since December 27th 2004:

```
http://www.arxiv.org/oai2?verb=ListRecords
&metadataPrefix=oai_dc&from=2004-12-27&set=physics
```

Since some OAI-PMH requests can result in a very long response, the repository uses a *resumptionToken* to separate the long responses into many shorter responses. A ListRecords response containing 1M records could be separated into 2000 incomplete lists of 500 records each. The fundamental, distinguishing characteristic that separates harvesting with OAI-PMH from regular web crawling is that the repository chooses the size of the resumptionToken, not the harvester. This allows repositories to dynamically throttle the load placed on them by harvesters. The format of the resumptionToken is not specified in the protocol and is left to individual repositories to define. Load-balancing, throttling and different strategies for resumptionToken implementation are discussed in the OAI-PMH Implementation Guidelines [24].

Another powerful feature of the OAI-PMH is that it can support any metadata format defined by means of an XML Schema. The minimum requirement is support for Dublin Core [47], but this metadata set is automatically derived by `mod_oai` from the http header information. This flexibility has generated considerable interest in liberal interpretations of the data model's elements - resource, *metadata, records, and items*. In some cases, this means using OAI-PMH for other than typical bibliographic scenarios [44]. In other cases, the interest is in transmitting the actual *resource* and not just the *metadata*.

3. COMPLEX OBJECT FORMATS AS METADATA

Web crawlers typically need access to more than just metadata. Even if the metadata included a full index of key terms, search engines will want to access the resource itself, in part to offer a "cached" copy to customers. At the same time, it is pointless for a crawler to request and process unchanged or duplicate resources. We recently reviewed a number of ad hoc resource harvesting strategies and described how they can lead to both missed updates and unnecessary downloads [43]. In the same article we introduced the concept of enabling accurate resource harvesting using XML-based complex object formats [43] such as the MPEG-21 Digital Item Declaration Language (MPEG-21 DIDL)[30]. Using OAI-PMH and complex object formats for resource harvesting in the repository synchronization project between Los Alamos National Laboratory and the American Physical Society is described in [5].

MPEG-21 DIDL is an XML-based instantiation of the data model defined by the MPEG-21 Digital Item Declaration (MPEG-21 DID) ISO Standard [45], which itself is representation independent. MPEG-21 DID introduces a set of abstract concepts that, together, form a well-defined abstract model for declaring Digital Objects. A simplified explanation of the MPEG-21 DID Abstract Model is given here; interested readers are referred to [6] for more detailed information. The MPEG-21 DID Abstract Model recognizes several MPEG-21 DID entities (written in *italic* font style), each of which has a corresponding XML element in the DIDL XML Schema [21]:

- An *item* is the declarative representation of a Digital Object. It is a grouping of *items* and/or *components*.
- A *component* is a grouping of *resources*. Multiple *resources* in the same component are considered bit-

Verb	Comment
Identify	returns a description of the repository (name, POC, etc.)
ListSets	returns a list of sets in use by the repository
ListMetadataFormats	returns a list of metadata formats used by the repository
ListIdentifiers	returns a list of ids (possibly matching some criteria)
GetRecord	given an id, returns that record
ListRecords	returns a list of records (possibly matching some criteria)

Table 1: OAI-PMH Verbs

equivalent and consequently it is left to an agent to select which one to use.

- A *resource* is an individual datastream.
- A *container* is a grouping of *containers* and/or *items*.
- Secondary information pertaining to a *container*, an *item*, or a *component* can be conveyed by means of a *descriptor/statement* construct.

When `mod_oai` exposes web resources via the OAI-PMH, it maps that resource to an XML-based representation of the resource that is compliant with MPEG-21 DIDL. Figure 2 presents a structural view of a web resource in MPEG-21 DIDL. The MPEG-21 DIDL XML elements have the same name as the corresponding entity of the MPEG-21 DID data model and, for clarity, are shown in `courier`:

The Web resource is considered a Digital Object, and hence is mapped to a top-level DIDL `Item` element. Two `Descriptor/Statement` constructs are attached to this `Item` to convey secondary information pertaining to the Web resource.

The Web resource URI is provided in a `Descriptor / Statement` construct, the content of which is compliant with the MPEG-21 Digital Item Identification Standard [7] that specifies how resources can be identified in the MPEG-21 framework. The http header information that would be provided if the resource was obtained through an http GET request is provided in a a separate `Descriptor / Statement` construct, the content of which is compliant with an XML Schema [25] specifically defined for the `mod_oai` project.

The Web resource itself - that is the datastream - is provided in a construct that contains one or two `Resources` in a `Component` that is a child element of the `Item`.

In all cases, the datastream is provided By-Reference by including the URI of the Web resource as the content of the *Ref* attribute of a `Resource` element. In cases where the file size of the datastream does not exceed a preset and configurable value, the datastream is also provided By-Value as the content of a `Resource` element. In this case the datastream is base64 encoded. In both cases, the MIME type of the Web resource is expressed by the *mimeType* attribute of the `Resource` element. The top-level `Item` is embedded in the `DIDL` root element to obtain an XML document that is compliant with MPEG-21 DIDL.

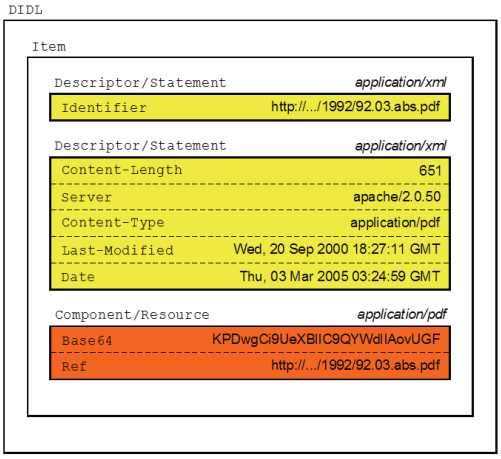


Figure 2: Resource expressed in MPEG-21 DIDL.

4. MOD_OAI

We have integrated OAI-PMH functionality into an Apache module, `mod_oai`, which responds to OAI-PMH requests on behalf of a web server. Adding the module to a web server results in a special purpose URI, called the "baseURL," which is defined from the site's root URL:
`http://www.foo.edu/` → `http://www.foo.edu/modoai`

Note that this does not represent a change to the underlying file system. It is a virtual path which instructs Apache to invoke the `mod_oai` handler. The original root URL is not affected; visitors and standard web crawlers continue to access the site as usual.

`mod_oai` is an instantiation of the general model of OAI-PMH resource harvesting. When recently discussing the advantages of OAI-PMH harvesting over web crawling with colleagues, they replied with "yes, OAI-PMH is nice, but who is going to do all that cataloging?" The answer, of course, is "no one" – OAI-PMH can be used outside of traditional catalogued resources. `mod_oai` has the advantage of *automatically* answering OAI-PMH requests for a web server. The module exposes the files on an Apache web server as MPEG-21 DIDL encoded resources through an OAI-PMH repository with the following characteristics in the OAI-PMH data model:

- OAI-PMH identifier: The URL of the resource serves as the OAI-PMH identifier. This choice facilitates a harvesting strategy whereby ListIdentifiers (with *from* and *until* parameters) is used to determine the URLs of web resources that have been updated since a previous harvest.
- OAI-PMH datestamp: The modification time of the resource is used as the OAI-PMH datestamp of all available metadata formats. This is because all metadata formats are dynamically derived from the resource itself. As a result, an update to a resource will result in new datestamps for all metadata formats.

- OAI-PMH sets: A set organization is based on the MIME type of resource. This choice facilitates MIME type specific resource harvesting, through the use of the set argument in protocol requests.

Three parallel metadata formats are supported. The most important of which is *oai_didl*. This metadata format is introduced to allow harvesting of the resource itself. In this metadata format, the web resource is represented by means of an XML wrapper document that is compliant with the MPEG-21 Digital Item Declaration Language (DIDL) [4], which has been devised to facilitate the representation of complex digital objects. This XML wrapper document includes the http_header metadata format (described below), as well as the web resource itself, provided using the By-Reference or By-Value approach, or both. Figure 2 shows a structural view of a web resource represented in the oai_didl (MPEG-21 DIDL) metadata format.

Dublin Core (*oai_dc*) is supported as mandated by the OAI-PMH, but only technical metadata that can be derived from http header information (file size, MIME type, etc.) is included. A new metadata format, *http_header*, is introduced. It contains all http response headers that would have been returned if a web resource were obtained by means of an http GET. This metadata format is included to ensure that no information would be lost as a result of choosing an OAI-PMH harvesting approach over a web crawling approach.

These design choices allow two main classes of `mod_oai` use: resource discovery and preservation, both of which offer selective harvesting semantics using datestamp and/or sets as selection criteria. The ListIdentifiers verb means an OAI-PMH harvester can be used as a URL discovery tool to identify web resources available from an Apache server, and using the resulting list of URLs as seeds for a regular web crawl. This fits the model of use for those with significant web crawling infrastructures (e.g., Google).

The ListRecords verb allows an OAI-PMH harvester to harvest the web resources represented by means of XML wrapper documents compliant with MPEG-21 DIDL. To ensure that harvesters that choose this approach instead of regular crawling obtain all the information they require, http header information represented using the http_header metadata format is also included in this XML wrapper. This use case corresponds to a preservation scenario where the harvester wishes to ensure that all resources are acquired from a web site.

Using our example site, `http://www.foo.edu/`, we can discover if the site's server supports `mod_oai` if we receive a valid OAI-PMH response to this request:

```
http://www.foo.edu/modoai?verb=Identify
```

The server administrator could advertise the existence of the baseURL more strongly by placing it in an "Allow:" field of the robots.txt file. The administrator could make `mod_oai` the exclusive access method for robots by adding the appropriate "Disallow:" fields in robots.txt.

To discover all PDFs from this site (ignoring all the navigational HTML pages) and feed the resulting URLs to a web crawler, a harvester could issue:

```
http://www.foo.edu/modoai?verb=ListIdentifiers
&metadataPrefix=oai_dc&set=mime:application:pdf
```

	Seed	
	index.html	"find . -type f"
# of files in baseline	709	5739
# of files in update (25%)	114	1318

Table 2: Files accessed by wget.

A ListSets response will contain the MIME types a particular server actually holds, not all known MIME types. Sets in OAI-PMH are recursive – requesting set "a" will get "a:b", "a:b:c", "a:d", etc. This can be exploited in `mod_oai` – to harvest all the 2004 videos encoded with MPEG-21 DIDL, a harvester could issue:

```
http://www.foo.edu/modoai?verb=ListRecords
&metadataPrefix=oai_didl&set=mime:video
&from=2004-01-01&until=2004-12-31
```

This request will return MPEG-21 encoded video-type files ("video:mpeg", "video:quicktime", "video:x-msvideo", etc.). Videos can be quite large, and whether or not the videos come back as By-Reference or By-Value is configurable by the `mod_oai` administrator.

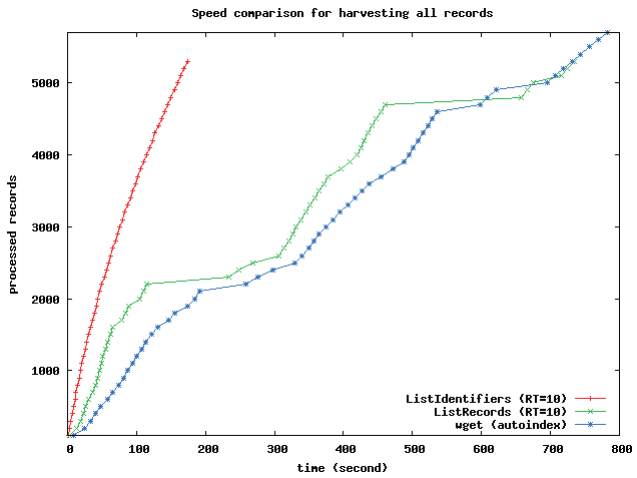
5. QUANTITATIVE EVALUATION

To examine the performance of `mod_oai`, we compared OAI-PMH harvesting using the OCLC Harvester [48] with the wget web crawling utility [2]. A copy of the Old Dominion University Computer Science Department web site (`http://www.cs.odu.edu/`) served as the testbed. We excluded a number of files, including user files (∼user), web mail files and data files from a survey utility. Overall, the testbed included 5268 files that used 292MB disk space. The server was at ODU and the client was at LANL.

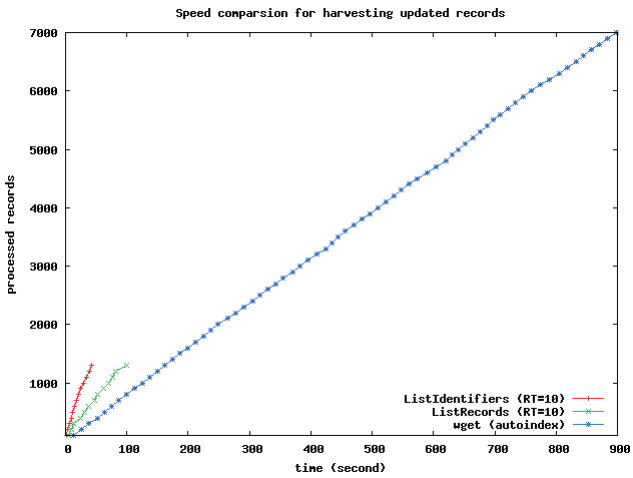
We performed two experiments. The first used the departmental homepage ("index.html") as a seed, and the 709 files crawled are those accessible via transitive closure from the departmental homepage. In the second experiment we used as a seed a web page with the list of all files found with "find . -type f". The significant difference between the numbers of files detected in both experiments (first row of Table 2) is because only a portion of valid URLs at a site are linked from web pages hosted on that site. The other URLs could be linked from pages on other sites or not linked at all. The time-stamp comparison feature in wget is turned on using the '-N' option. This will cause wget to check if a local file of the same name exists and only download the remote file if it is "newer" than the local file. Table 2 shows the number of files accessed by wget in both scenarios. Using the "find" seed, wget downloads more URLs (5739) than there are files (5268). This is because it finds additional URLs that the seed points to, including directories and broken links. The full wget command is:

```
wget -r --no-parent
--exclude-directories=/modoai,
-N $INDEX -o $INDEXLOG -P $INDEXMIRROR
--dns-timeout=1 --connect-timeout=1 --tries=1
```

Using the seed generated with "find", we baselined both wget and `mod_oai` with all file modification dates set to "2000-01-01" For our second test, we touched 25% of the files



(a) Baseline wget & mod_oai



(b) wget & mod_oai after 25% file updates

Figure 3: Comparison of crawling performance

(1335 files) to make their modification dates "2002-01-01". This simulated the monthly update rate expected for ".edu" sites [13]. For `mod_oai`, we issued two different request types (ListRecords, ListIdentifers), and two different "from" values (1900-01-01, 2001-01-01). We restarted Apache after each round of harvesting. Figure 3(a) shows the time required for the baseline for all files and Figure 3(b) shows the time required for just the updated files. It is surprising that wget takes more time in accessing only the updated files. The Apache log file shows that wget uses both the http HEAD and GET methods to check the time. Thus, in checking for updates, wget will use more http requests (5739 HEAD + 1318 GET).

We also tested the performance of `mod_oai` with resumptionToken sizes of 5, 10, 20, 50 and 100 (Figure 4). With ListRecords, the performance increase leveled off at a resumptionToken size of 50. With ListIdentifiers, performance continued to increase with increasing the resumptionToken size. This is due to the fact that ListRecords returns the base64-encoded file, and ListIdentifiers returns just the resource identifiers. These results imply that we should have different resumptionToken sizes for ListRecords and ListIdentifiers.

In fact, this has suggested an altogether new approach to resumptionToken size configuration. In bibliographic oriented OAI-PMH instantiations there is an implicit assumption that all records are roughly the same size. However, when complex object formats are used to base64 encode the resource, the size of each record will be highly variable. The resumptionToken should therefore be a function of the size of the response (e.g., 1MB) instead of the number of records (e.g., 100).

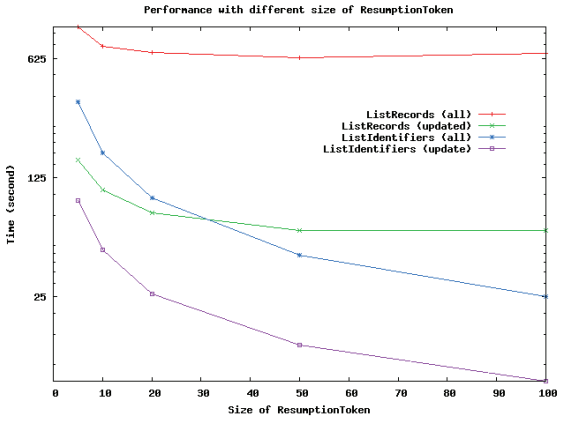


Figure 4: Impact of resumptionToken size.

6. DISCUSSION & FUTURE WORK

During the development of `mod_oai`, we encountered a number of subtle but important issues that illustrate the fundamental differences between web crawling and OAI-PMH harvesting. We are actively exploring the best way to resolve these mismatches. One issue is the "counting problem:" what constitutes a complete list of the site's crawlable resources? Another mismatch is the "representation problem" - the resource as sent to a browser is not necessarily an optimal representation for the crawler. `mod_oai` can help solve both problems.

6.1 The Representation Problem

`mod_oai` does not generate descriptive metadata. The values it populates in Dublin Core are strictly technical metadata derived from the file (bytes, MIME type, last modified date and URL). This is a conscious decision: we are not positioning `mod_oai` as a replacement for existing repository systems (e.g., Fedora[34], DSpace[39]) with extensive descriptive metadata. Rather, we aim to improve the efficiency of web crawlers which typically do not have access to extensive descriptive metadata. Some crawlers actively ignore the potentially dishonest metadata they receive (cf. [29]).

We are developing a plug-in architecture for mod_oai that will specify rules for extracting descriptive metadata from various MIME types and/or URL patterns. One benefit of the plug-in approach is the ability to integrate third-party metadata extraction tools without requiring a rewrite of the mod_oai module. Even though the web crawling community has become adept at extracting metadata directly from web resources, this plug-in architecture would enable a site to create complex objects as preservation packages for long-term archiving.

We are working on including technical and structural metadata at dissemination time as well. Capitalizing on our work of capturing "forensic metadata" in MPEG-21 DIDL [31], mod_oai administrators can automatically include the output of provenance and fixity utilities in ListRecords responses. (For a discussion of OAIS preservation terminology including "provenance" and "fixity", see [10]). Lastly, web servers will frequently transform files before serving them to the client (e.g, .cgi, .php, .shtml, .jsp, .shtml) and it is a potential security-hole to export the unprocessed file. For example, a .php file might have database passwords in the source file. What is the "correct" representation: the .php source, or the processed HTML sent to the client? Although this is primarily a representation problem, dynamic files producing infinite output also impact the counting problem. Currently, mod_oai will ignore any file that requires server-side processing by checking to see if there is a handler registered for the particular file type. We are currently working on techniques to securely support both modes of exporting dynamic files.

6.2 The Counting Problem

The first problem we encountered was simply determining how many files were on a web server. Traditional OAI-PMH applications are deterministic with respect to the number of records that the repository holds. Most OAI-PMH repository implementations are accessing a database in which all possible records are knowable. However, web harvesting is different. We define U as the set of all possible URLs for a particular web server, and F as the set of files that the web server can see. Apache maps U → F, and mod_oai maps F → U. Neither function is 1-1 nor onto. We can easily check if a single URL maps to F, but given F we cannot (easily) generate U. One problem is that Apache can "cover up" legitimate files. Consider two files, A and B, on a web server. Now consider an httpd.conf file with these directives:

```
Alias /A /usr/local/web/htdocs/B
Alias /B /usr/local/web/htdocs/A
```

The URLs obtained by web crawling and the URLs obtained by OAI-PMH harvesting will be in contradiction. That is, a user or crawler requesting http://www.foo.edu/B will actually receive the resource from htdocs/A, and vice-versa. mod_oai, on the other hand, is unaware of the alias and returns metadata from htdocs/B as if the directive did not exist.

Files can also be covered up by Apache's Location directive which is used with (among other things) Apache modules. For example, a file named "server-status" would be exported by mod_oai, but would not be accessible directly from the web server if the "mod_status" module is installed. Automounting of Network File System (NFS) directories is another example of a complication to the counting problem. NFS mounted directories shared between many departmental machines is a common deployment scenario and makes it extremely difficult to include UserDir files (e.g., http://www.cs.odu.edu/~mln/) in mod_oai responses since there is no (easy) way to know in advance all possible users. However, these files constitute the majority of files accessible from a web server in a shared-user environment such as a university department. mod_oai currently ignores files impacted by Alias, Location and UserDir, although future versions will resolve these conflicts.

6.2.1 Security

Users and administrators have developed bad habits based on "security through obscurity" – if there is no URL to a resource, a search engine cannot find it. Unfortunately, mod_oai is just one of many services (or people, in shared filesystem environments) that can unwittingly reveal previously unknown URLs to search engines. This is similar to the problem of search engines revealing files and metadata that users did not realize were available (e.g., [38, 16, 22]). It is important to stress that mod_oai will not export any files that are not accessible through normal http access (see the discussion in section 6.2.2); mod_oai provides configuration facilities for regular expressions to skip, as well as entire directories or URL patterns to avoid. Reasonable defaults are supplied in the mod_oai configuration, but it cannot enforce good habits by users and administrators.

mod_oai currently handles files protected by Apache by checking each file before it is included in a response to see if the necessary credentials in the current http connection are sufficient to meet the requirements specified in the .htaccess file. As a result, harvesters with different credentials will see a different number of records for the same server. Since harvesting is not interactive, passwords have to be included in the http environment when the harvest is begun.

6.2.2 Hidden Files

mod_oai will not advertise any file that the request does not have the credentials to retrieve. This prevents OAI-PMH errors from being generated when a harvester tries to access a protected file, but it also means that harvesters with different credentials will see a different "view" of the same mod_oai baseURL. Somewhat related is the problem that Apache will advertise files that it cannot read. For example, a file can be seen in a directory listing, but if the permissions on the file are "000" then no one can actually read the file. The file is listed by Apache since its existence is actually a property of the parent directory, not the file itself. To preserve OAI-PMH semantics, mod_oai will not include such files in responses.

Apache also uses the IndexIgnore directive to specify patterns for filenames that should not be included in a directory listing (e.g., "foo~" "foo#" and other file version conventions). However, if requested directly, Apache will serve it: http://www.foo.edu/index2.html~, for example. The Apache semantics in this scenario are similar to "hidden" files in the Unix filesystem. This has serious implications for OAI-PMH – it would be equivalent to the undesirable scenario where more files are available via GetRecord than ListRecords. To preserve OAI-PMH semantics, mod_oai ignores any files of the type specified in IndexIgnore.

6.2.3 Google Sitemaps

Google's Sitemap protocol [1] is a static XML file that lists the URL and a number of optional descriptors of the URL, such as the estimated change rate (chosen from a controlled vocabulary of values such as "always", "weekly", "yearly", etc.), date of last modification and a local crawling priority (*not* necessarily Google's priority).

For comparison, we ran sitemap_gen.py in mode 3 which is similar to `mod_oai`'s method for generating valid URLs. The Sitemap script generated a list of 5843 URLs – note that this does not match any of the numbers in Table 2. Compared to our methods the Sitemap script was overly optimistic, returning some URLs that did not exist (we are not sure why), and several URLs that were protected with a .htaccess file (http status code 403).

Google's Sitemap is designed to be an extremely lightweight mechanism for informing Google's web crawlers of new and updated URLs at a web site, similar to using `mod_oai` with only "ListIdentifiers" and no datestamps, sets or metadata formats. There is a trade-off between the dynamic access of `mod_oai` and the static access of Sitemap. A `mod_oai` response will always be up to date, but at the cost of computation. Similarly, Sitemap will be only as up to date as the local refresh policy, but each crawler access to Sitemap will not impose a computational cost on the server.

7. CONCLUSIONS

`mod_oai` is a demonstration of combining complex object metadata formats with OAI-PMH for efficient web resource harvesting. It is an Apache 2.0 module that exposes a general web server as an OAI-PMH repository. Our initial tests have shown that simply generating a list of valid URLs for a web site is not easily done. We have also presented initial performance results of `mod_oai` and wget on a typical university department web site. We have shown that `mod_oai` offers comparable performance to wget for baseline harvests and outperforms wget when file updates are considered. We have described how the OAI-PMH semantics are interpreted in the context of an Apache server, including some of the mismatches that occur between the filesystem, Apache web server and the `mod_oai` module that complicate the semantic mapping at each level.

`mod_oai` can be used in two scenarios: *resource discovery*, in which a harvester can generate a list of URLs to be fed to regular web crawlers (using ListIdentifiers), and *preservation*, in which `mod_oai` will export MPEG-21 DIDL encoded versions of the content (using ListRecords). `mod_oai` is not intended to replace existing OAI-PMH implementations. Rather, it is intended make web crawling more efficient by embedding support for OAI-PMH semantics of incremental harvesting based on datestamps and sets directly in an Apache web server using Apache's module infrastructure. An added advantage to this approach is that Apache modules are well understood by the webmaster community.

To date, much of the research in the web community has focused on efficiently estimating updates and additions of remote, uncooperative web servers. Now, there is interest in shifting some of the responsibility for resource discovery to the web servers themselves. `mod_oai` can help this effort by utilizing the local knowledge of web servers for more efficient web crawling.

8. ACKNOWLEDGMENTS

`mod_oai` is supported by the Andrew Mellon Foundation. Aravind Elango and Terry L. Harrison contributed to the `mod_oai` source code. Jeroen Bekaert contributed to the MPEG-21 DIDL profile of `mod_oai`. The `mod_oai` web site is www.modoai.org.

9. REFERENCES

[1] Creating google sitemaps files. http://www.google.com/support/webmasters/bin/topic.py?topic=8467.

[2] GNU wget GNU Project Free Software Foundation (FSF). http://www.gnu.org/software/wget/wget.html.

[3] Windows live search academic. http://academic.live.com/Publishers_Faq.htm.

[4] J. Bekaert, P. Hochstenbach, and H. Van de Sompel. Using MPEG-21 DIDL to represent complex digital objects in the Los Alamos National Laboratory digital library. *D-Lib Magazine*, 9(11), 2003.

[5] J. Bekaert and H. Van de Sompel. A standards-based solution for the accurate transfer of digital assets. *D-Lib Magazine*, 11(6), 2005.

[6] J. Bekaert, E. De Kooning, and H. Van de Sompel. Representing digital assets using MPEG-21 Digital Item Declaration. *International Journal on Digital Libraries*, 6(2), 2006.

[7] J. Bekaert and N. Rump. MPEG-21 DII (Output Document of the 71st MPEG Meeting, Hong Kong, China, ISO/IEC JTC1/SC29/WG11/N6928). Technical report, 2005.

[8] M. K. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), 2001.

[9] O. Brandman, J. Cho, H. Garcia-Molina, and N. Shivakumar. Crawler-friendly web servers. *SIGMETRICS Performance Evaluation Review*, 28(2):9–14, 2000.

[10] Consultative Committee for Space Data Systems. Reference Model for an Open Archival Information System (OAIS). Tech Report CCSDS 650.0-B-1, 2002.

[11] J. Caverlee, L. Liu, and D. Buttler. Probe, cluster, and discover: Focused extraction of qa-pagelets from the deep web. In *ICDE*, pages 103–115, 2004.

[12] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, 2003.

[13] J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3):256–290, 2003.

[14] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. *Computer Networks and ISDN Systems*, 30(1-7):161–172, 1998.

[15] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. HTTP extensions for distributed authoring – WEBDAV. Tech Report Internet RFC-2518, 1999.

[16] S. Granneman. The perils of googling. http://www.theregister.co.uk/2004/03/10/the_perils_of_googling/, 2004.

[17] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. *WWW 2005*, 5, May 2005.

[18] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: a repository of Web pages. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):277–293, 2000.

[19] P. Hochstenbach, H. Jerez, and H. Van de Sompel. The OAI-PMH static repository and static repository gateway. In *Proceedings of JCDL '03*, pages 210–217, 2003.

[20] P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: Categorizing hidden web databases. In *SIGMOD Conference*, pages 100–109, 2001.

[21] ISO/IEC. ISO/IEC 21000-2:2005 information technology - multimedia framework (MPEG-21) - part 2: Digital item declaration - schema for derived DIDL types. http://purl.lanl.gov/STB-RL/schemas /2004-11/DIDL.xsd.

[22] A. Klein. The insecure indexing vulnerability. http://www.webappsec.org/projects /articles/022805.shtml, 2005.

[23] C. Lagoze and H. Van de Sompel. The Open Archives Initiative: building a low-barrier interoperability framework. In *Proceedings of JCDL '01*, pages 54–62, 2001.

[24] C. Lagoze, H. Van de Sompel, M. L. Nelson, and S. Warner. Implementation guidelines for the Open Archives Initiative Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/2.0/guidelines.htm, 2005.

[25] X. Liu. XML schema defining a subset of HTTP headers used by mod_oai project. http://purl.lanl.gov/STB-RL/schemas/2004-08 /HTTP-HEADER.xsd.

[26] X. Liu, K. Maly, M. Zubair, and M. L. Nelson. Repository synchronization in the OAI framework. In *Proceedings of JCDL '03*, pages 191–198, 2003.

[27] Z. Liu, C. Luo, J. Cho, and W. W. Chu. A probabilistic approach to metasearching with adaptive probing. In *ICDE*, pages 547–559, 2004.

[28] P. Lyman. Archiving the world wide web. In *Building a National Strategy for Preservation: Issues in Digital Media Archiving*. Council on Library and Information Resources, 2002.

[29] C. A. Lynch. When documents deceive: trust and provenance as new factors for information retrieval in a tangled web. *Journal of the American Society for Information Science and Technology*, 52(1):12–17, 2001.

[30] J. P. McDonough. METS: Standardized encoding for digital library objects. *International Journal on Digital Libraries*, 6(2):148–158, 2006.

[31] M. L. Nelson, J. Bollen, G. Manepalli, and R. Haq. Archive ingest and handling test: The Old Dominion University Approach. *D-Lib Magazine*, 11(12), 2005.

[32] A. Ntoulas, J. Cho, and C. Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proceedings of WWW '04*, pages 1–12, 2004.

[33] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden web content through keyword queries. In *Proceedings of JCDL '05*, pages 100–109, 2005.

[34] S. Payette and C. Lagoze. Flexible and extensible digital object and repository architecture (FEDORA). In *ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 41–59.

[35] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, pages 129–138, 2001.

[36] J. Reagle. Web RSS (syndication) history. http://goatee.net/2003/rss-history.html, 2003.

[37] H. Suleman. OAI-PMH2 XMLFile file-based data provider. http://www.dlib.vt.edu/projects/OAI/ software/xmlfile/xmlfile.html, 2002.

[38] D. Sullivan. A closer look at privacy & desktop search. http://searchenginewatch.com/sereport/article.php /3421621, 2004.

[39] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, and M. Smith. The DSpace institutional digital repository system: current functionality. In *Proceedings of JCDL '03*, pages 87–97, 2003.

[40] H. Van de Sompel, T. Krichel, M. L. Nelson, P. Hochstenbach, V. M. Lyapunov, K. Maly, M. Zubair, M. Kholief, X. Liu, and H. O'Connell. The UPS prototype: An experimental end-user service across e-print archives. *D-Lib Magazine*, 6(2), 2000.

[41] H. Van de Sompel and C. Lagoze. The Santa Fe Convention of the Open Archives Initiative. *D-Lib Magazine*, 6(2), 2000.

[42] H. Van de Sompel and C. Lagoze. Notes from the interoperability front: A progress report on the Open Archives Initiative. In *Proceedings of ECDL '02*, pages 144–157, 2002.

[43] H. Van de Sompel, M. L. Nelson, C. Lagoze, and S. Warner. Resource harvesting within the OAI-PMH framework. *D-Lib Magazine*, 10(12), 2004.

[44] H. Van de Sompel, J. A. Young, and T. B. Hickey. Using the OAI-PMH ... differently. *D-Lib Magazine*, 9(7/8), 2003.

[45] R. Van de Walle, I. Burnett, and G. Dury. ISO/IEC 21000-2 Digital Item Declaration (Output Document of the 70th MPEG Meeting, Palma De Mallorca, Spain, No. ISO/IEC JTC1/SC29/WG11/N6770), 2004.

[46] A. van Hoff, J. Giannandrea, M. Hapner, S. Carter, and M. Medin. The HTTP distribution and replication protocol. W3C Technical Report http://www.w3.org/TR/NOTE-drp, 1997.

[47] S. Weibel. Metadata: The foundations of resource description. *D-Lib Magazine*, 1(1), 1995.

[48] J. Young. OAIHarvester2. http://www.oclc.org/research/software/oai /harvester2.htm, 2005.