A Quantitative Evaluation of Dissemination-Time Preservation Metadata

Joan A. Smith and Michael L. Nelson

Old Dominion University, C.S. Dept, Norfolk VA 23529

Abstract. One of many challenges facing web preservation efforts is the lack of metadata available for web resources. In prior work, we proposed a model that takes advantage of a site's own web server to prepare its resources for preservation. When responding to a request from an archiving repository, the server applies a series of metadata utilities, such as Jhove and Exif, to the requested resource. The output from each utility is included in the HTTP response along with the resource itself. This paper addresses the question of feasibility: Is it in fact practical to use the site's web server as a just-in-time metadata generator, or does the extra processing create an unacceptable deterioration in server responsiveness to quotidian events? Our tests indicate that (a) this approach can work effectively for both the crawler and the server; and that (b) utility selection is an important factor in overall performance.

1 Background

There are many on-going efforts aimed at web preservation. One problem shared by these efforts is the dearth of metadata available directly from websites themselves. For preservation, we need much more metadata than is customarily available from an HTTP request-response event. A common approach to this problem is to crawl the site then have the archivist store the resources for later analysis and repository ingestion. However, we believe that the best time to analyze a file is at the time of the request, when the server itself is more likely to be able to provide preservation-related information. We also believe that automated metadata utilities installed at the originating web server can contribute meaningfully to web preservation.

We demonstrated this as a proof-of-concept in prior work [1, 2], but the question remained whether it is *practical* to use the site's web server as a just-in-time metadata generator. Does performance suffer an unacceptable deterioration? Can an archival request be serviced simultaneously with quotidian web requests? To investigate the feasibility of this approach, we constructed a "typical" website for testing based on an analysis of published web site characteristics. We then subjected this test website to varying request (load) levels and harvested the contents to determine the performance impact of creating preservation metadata at dissemination time, i.e., at the time of the request. We found that for all metadata utilities but one, we could process the results without a significant impact on server performance overall. Our tests indicate that (a) this approach can work effectively for both the crawler and the server; and that (b) utility selection is an important factor in overall performance.

2 Related Work

2.1 Characterizing a Typical Website

Typical Website Content. Since the website's resources would be passed through the rigors of various metadata utilities, we wanted our test web to mimic a "typical" website in terms of content and structure. But what, exactly, is a typical website and what does a typical web page contain? An extensive survey of web content was published by Berkeley in 2003 [3]. At that point, surface web composition was roughly 23.2% images, 17.8% HTML, and 13% PHP, with the rest a collection of other formats ranging from PDFs to animations. More recent studies support this rough proportion, noting that most web pages have one or more images embedded in them thus contributing to a higher ratio of images to HTML resources but still supporting the intuitive impression that the web is largely HTML [4, 5, 6].

With regard to website size and content, a 2004 report on the composition of various national domains [4] showed a wide range of average number of pages per site, with a low of 52 (Spain) to a high of 549 (Indochina). That same study also indicated a preponderance of HTML over other document types, with PDF and plain text files accounting for up to 85% of the remainder (these figures do not include image files). Various studies on web content and configuration [5, 6] found that most HTML documents contain less than 300 words, with a per-page average of 281 HTML tags and a 221x221 pixel image (usually GIF or JPEG) that acted as a document header, much like the banner name of a newspaper. A 2004 examination of e-commerce sites at a large server farm [7] found an average object size of 9 KB and a much higher percentage of image use than seen in other studies, which the authors attribute to the nature of e-commerce sites. Other researchers [8, 9] have noted an increasing use of dynamic presentation technologies like Javascript, PHP, and Active Server pages.

Despite the many web studies available, no clear characterization of a "typical" website emerges, except perhaps at the extremes: single-page sites (often at "spam farms") and infinite sites, which use dynamic-generation to create infinite pages, such as a meeting-schedule site with a limitless value for future date. We are therefore left to "guesstimate" the composition of a small departmental or community website in terms of size and types of resources. The general tendency seems to be a small website of a few hundred files, with the HTML pages roughly 5 KB to 25 KB in size; having approximately 3 or more images embedded per HTML page; containing links to various internal resources distributed throughout the site, and a variety of external links on selected pages.

Typical Website Traffic Patterns. Many studies have been done on web traffic patterns, including some at large commercial sites [7, 9]. Data from these studies enable researchers to model request patterns realistically when simulating traffic at a test website. Key findings applicable to this project are the rate and distribution of requests, which show a Pareto-type distribution, i.e., the majority of the requests (80% to 90%) typically cover only 10% to 20% of the site's total resources. This aspect of web traffic has made it possible for webmasters to fine-tune their web server configurations. As a result, the server will typically have the majority of incoming requests already available in cache, improving overall response time. Other website traffic studies [6, 10] have focused on analyzing and improving search-engine-crawler efficiency. Because crawlers access all of a site's resources, server performance can suffer as it swaps seldom-used pages in and out of memory to satisfy the robot's requests (a locality of reference problem).

2.2 Resource Harvesting with OAI-PMH

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) was developed to facilitate interoperability among repositories. Most digital librarians know this protocol as a means to obtain information *about* objects in a repository, such as Dublin Core or MARC metadata. However, OAI-PMH can also be used to *harvest the resources* themselves, not just their metadata [11]. An Apache web server implementation of OAI-PMH ("mod_oai") brought the rich functionality of the protocol to the web server [1]. It overcomes inherent HTTP limitations by allowing, for example, date-range requests ("give me everything new since March 13, 2008") or MIME type requests ("give me all of the JPEG resources you have"). Here's an example of a request to a mod_oai-enabled web server that covers both of these criteria:
`http://www.foo.edu/modoai?verb=ListRecords&From=2008-03-13`
`&set=mime:image:jpeg&metadataPrefix=oai_didl`.

With OAI-PMH, web harvesting becomes much more efficient while still operating within the HTTP protocol. Instead of having to conduct the traditional link-by-link crawl of a site, a single OAI-PMH *ListIdentifiers* request can produce a full sitemap. OAI-PMH can also produce complex-object format responses. For example, the *ListRecords* request can return every resource, together with a set of minimal metadata, in MPEG-21 DIDL format [12]. Each resource — an image, PDF, text file, etc. — is encoded in Base64 and encapsulated with its metadata in the response. The MPEG-21 DIDL output contains plain ASCII in an XML-format; an abbreviated example is shown in Figure 1.

OAI-PMH and the MPEG-21 DIDL format are quite flexible: a response *could* contain more information, if it were available from the server. Experimenting with this concept, we expanded mod_oai to accept *plugins*, third-party metadata utilities which analyze the resource as it is being requested [2]. The final response contains the resource (as Base64), the minimal metadata *plus* the additional, expanded information from the metadata utilities, all packaged together in an MPEG-21 DIDL. We called this new aggregation of resource + metadata a "CRATE" and used mod_oai in an experimental CRATE prototype to demonstrate that it *can* be done [1, 2]. In the CRATE approach, metadata plugins are implemented on a per-resource basis depending on file type. They can be applied to a certain set of files (only images; only HTML; only plain text; etc.), or to every

```
<crateplugin>
<crateplugin:name>file</crateplugin:name>
<crateplugin:version>
<![CDATA[1.0.5]]>
</crateplugin:version>
<crateplugin:content>
<![CDATA[
/var/www/testWeb/group8/pdf120.pdf:
PDF document, version 1.3]]>
</crateplugin:content>
</crateplugin>
<crateplugin>
<crateplugin:name> md5sum </crateplugin:name>
<crateplugin:version>
<![CDATA[md5sum (GNU coreutils) 5.93
Copyright (C) 2005 Free Software
Foundation, Inc. This is free software.
You may redistribute copies of it under
the terms of the GNU General Public License
<http://www.gnu.org/licenses/gpl.html>.
There is NO WARRANTY, to the extent permitted
by law. Written by Ulrich Drepper and
Scott Miller.]]> </crateplugin:version>
<crateplugin:content>
<![CDATA[e1f66cd707c2df36dafe8557d82536a1]]>
</crateplugin:content>
</crateplugin>
```

Fig. 1. Partial OAI-PMH response (CRATE) shown in MPEG-21 DIDL format

resource ("*/*"), or to some specific combination of selected resources. Our proof-of-concept incorporated a variety of utilities on the web server to process each resource just as it is being disseminated (sent over HTTP). That is, when a web resource is requested ("GET x.html"), the web server calls the various utilities, gathers the output, then responds to the GET request by providing the aggregated utility data and the resource together in the response. An example of the output can be seen in Figure 1. Showing that it *can* be done does not say anything about *feasibility*. This paper takes a first look at the issues of feasibility and practicality from the quantitative perspective of web server performance.

3 Experiment Design

The goal of our experiment was to answer two key questions of concern to both webmasters and archivists, i.e., (1) Is it safe to generate metadata directly from the web server? (2) Is it safe to ask for such metadata? To answer the first question, we created a set of tests to see if a web server could reasonably provide CRATE-type responses (resource + metadata) without producing an unacceptable deterioration in the server's normal responsiveness to general users. To answer the second question, we monitored metadata utility response time, and the time and size of the output. Three components form the core of the experiment: (1) A test website; (2) a variety of common metadata utilities, and (3) a test environment (web traffic simulator).

For performing the tests, we used a commercial testing environment provided by a local software firm which develops web applications. The web server and "users" in the test environment are representative of mid-range web servers installed at the firm's clients. Users are simulated using Apache's JMeter v.2.3.1 software on a quad dual core AMD Opteron PC running Windows 2003 Server. This configuration allows many thousands of users to be simulated simultaneously. The web server has two 2 GHz AMD Opteron processors, 2 GB RAM, 6 GB of swap space, and an Ultra-320 SCSI hard disk operating at 10,000 RPM.

Server operating system is Red Hat Enterprise Linux version 4 (Linux 2.6.9-67.0.1 ELSMP), with Apache version 2.0.52 installed. Although the server is capable of having up to 100 Apache threads, we left it in the default "10 thread" configuration to represent a worst-case performance scenario.

3.1 Designing the Test Website

Research on the evolution of web page composition [5, 6] and on commercial web server traffic [7] provided guidelines for site design and traffic expectations. Average web page size in these studies ranged from 5 KB to 25 KB (a figure which includes the size of embedded images), with shopping sites often having a higher size because of a large number of embedded images. We based our overall website content on characteristics found in [3] and [5], allowing for a higher percentage of PDF, Microsoft Word ("DOC") and Powerpoint ("PPT") files to better reflect the content of our hypothetical site, where official forms, permits, and lecture notes are likely to be in printable PDF, DOC, or PPT formats.

HTML pages were built using a script we developed for other research projects. Content was extracted from Project Gutenberg e-text files, and images came from a variety of sources including Project Gutenberg and the authors' personal creations. The PDF files were created using a template which produced content similar to the HTML pages; each PDF file ranged from 1-3 pages in length. A collection of DOC and PPT files were created using Microsoft Office. These and the other files were randomly assigned to HTML pages throughout the site. If the random resource was a PNG, JPEG or GIF image, it was "embedded" in the page; otherwise, it was represented as a linked resource. Each resource was unique in content, and the site layout was a reasonable facsimile of a small, quotidian website. Table 1-(a) describes the overall content of the site by file type and hierarchy, and Table 1-(b) shows the resource distribution by type and size for the test website.

3.2 Metadata Utilities

Utility Selection Criteria. Our target environment is the small to mid-size website where there is interest in preservation but no budget to support it in terms of manpower or software investment. For example, a small-town citizen information website, or a university department-level website with perhaps only one professional webmaster and/or a group of students who act as webmaster support. With this is mind, four elements were defined as the primary selection factors for the metadata utilities to be included in our test: (1) Cost, (2) Operating System, (3) Invocation Method and (4) Ease of Installation.

Many, if not most, small departmental and community web servers operate under an extremely constrained budget. Cost, therefore, had to be a factor in selecting our test utilities. Each utility also had to be installable under our Linux OS but ideally under any Unix-like operating system (Sun OS, OS-X, etc.) where mod_oai could be installed. The utilities all provide a command-line invocation

Table 1. Resource organization and distribution on the test website. Each "Grp" directory has 4 directories below it, which contain the bulk of the website content. †Resources were randomly chosen from the list. ‡Other file types include SVG, MP3, WMV, and ASCII text.

(a) Organization

File Type	**Grp** *1-12*	**Dir** *1-4*	**Site Count**
"Home"	n/a	n/a	1 HTML
	n/a	n/a	3 GIF
HTML	1	10	492 HTML
Image	≥ 3	≥ 3	195 GIF
	$\leq 2^{\dagger}$	$\leq 1^{\dagger}$	51 JPEG
	$\leq 2^{\dagger}$	$\leq 1^{\dagger}$	51 PNG
App.	$\geq 3^{\dagger}$	$\leq 1^{\dagger}$	144 PDF
	$\leq 1^{\dagger}$	$\leq 1^{\dagger}$	48 DOC
	$\leq 1^{\dagger}$	$\leq 1^{\dagger}$	50 PPT
Other‡	$\leq 1^{\dagger}$	$\leq 1^{\dagger}$	49 (Total)
Total Files:			**1084**

(b) Distribution

Ext (MIME)	**Site Count**	**Avg.Bytes per File**
mp3	11	124165
png	51	10016
pdf	144	232579
ppt	50	744652
txt	14	7488
wmv	11	58922
html	493	2511
jpeg	51	6052
doc	47	32789
svg	14	24374
gif	198	6043
Total Bytes:		**77,979,284**

method. This is necessary because mod_oai issues the equivalent of a command-line request to each utility. It also enables us to automate the process of passing a single resource through each of the utilities via the Apache configuration file. Finally, ease-of-installation is important if we expect the average webmaster to be responsible for installing and configuring such utilities. External dependencies like software libraries should already be installed or should come packaged with the utility and be automatically included in the installation process.

Utilities Considered for Inclusion. There are many utilities that offer attractive analytical capabilities but which are not practical candidates. Some (e.g., Oxford's WordSmith tools [13]) are purely GUI-oriented, Windows-based products. Others such as Essence ([14]) are closer to frameworks than to utilities, with complicated installation and configuration requirements. Another popular utility we were not able to include is the keyphrase analyzer Kea ([15]), because it has to be "trained" for each document collection with a set of candidate texts and author-designated index terms. But others, like Jhove and Metadata Extractor, are both practical candidates and produce useful preservation metadata.

There is some duplication of analysis among the utilities considered. Exif, a utility for analyzing digital photo files, overlaps with Jhove's JPEG HULs, for example. Such duplication can be informative. Analysis results do not always agree between any two utilities, and input from multiple sources may help the archivist. For instance, the two sites: (a) `http://www.library.kr.ua/cgi-bin/lookatdce.cgi` and (b) `http://www.ukoln.ac.uk/cgi-bin/dcdot.pl` use different methods to extract and assign Dublin Core from HTML pages, and so their results often differ. Automated Dublin Core metadata extraction proved to be a bigger problem than we had expected. The two Dublin Core analysis utilities at the sites mentioned above are not designed for the automated, batch-style processing required by mod_oai. As a last resort, we wrote a short Perl script which simply extracts the <META> tags from the <HEAD> section of HTML documents. The result is not true Dublin Core, but the methodology is similar to the approach taken by other, GUI-based Dublin Core tools.

Utilities Selected for the Experiment. Several utilities were clear candidates for selection, easily meeting the criteria of (1)cost, (2)OS, and (3)batch-mode compatibility. A couple of utilities posed more installation issues than we would like to see (criterion 4), but they offer useful metadata and were included despite these difficulties. The eleven utilities used in the experiments were: (1) Jhove; (2) Exif; (3) Word Count, "WC"; (4) Open Text Summarizer, "OTS"; (5) File Magic, "file"; (6) Pronom-Droid, "droid"; (7) Metadata Extraction Tool, "MetaX"; (8) dcTag (our home-grown utility), and three hash functions, SHA, SHA-1, and MD5. There is some duplication of analysis; both Jhove and Exif are applied to JPEG resources, for example. The utilities represent a range of implementations, from tools like File Magic (Linux "file" command) and the hashes (MD5, SHA, SHA-1) which are installed by default with the operating system; to open source products written in C (Open Text Summarizer) which have to be compiled and installed on the target web server; to Perl-based scripts (dcTag) and Java utilities (Jhove, Metadata Extractor, and Pronom-Droid).

3.3 Site Setup

Configuring The Web Server. Like other Apache modules, mod_oai activity is controlled through the web server configuration file. A snippet from the mod_oai section is shown in Figure 2. Each *modoai_plugin* line specifies a label for the utility (ex: "md5sum"); the executable command path, with "%s" acting as a placeholder for the website resource to be processed; the command path to generate plugin version information (ex: "/usr/bin/md5sum -v"); and the range of MIME types to be processed by the plugin. For example, "*/*" indicates all resources are processed by that particular plugin, whereas "image/jpeg" indicates that only JPEG images will be processed.

Simulating Web Traffic. An important question to ask when evaluating the impact of metadata utilities is how it affects performance under normal server load, i.e., the traffic volume typically expected at the website. We configured our test server for the maximum possible traffic it would support which ranged from 88-93 requests per second. This number is significantly higher than that reported in [7] for the busiest commercial site, which experienced a maximum request rate of 25 per second. We modeled our request patterns to mimic the normal Pareto distribution seen in website traffic logs, i.e., the majority of the requests (80% to 90%) typically are for only 10% to 20% of the site's total resources.

```
Alias /modoai "/var/www/"
<Location /modoai>
  SetHandler                modoai-handler
  modoai_sitemap            /var/www/sitemap.xml
  modoai_admin              smith
  modoai_email              admin@foo.edu
  modoai_gateway_email      mail@foo.edu
  modoai_oai_active         ON
  modoai_encode_size        10000
  modoai_resumption_count   10000
  modoai_plugin wc '/usr/bin/wc %s' '/usr/bin/wc -v' text/*
  modoai_plugin file '/usr/bin/file %s' '/usr/bin/file -v' */*
  modoai_plugin md5sum '/usr/bin/md5sum %s' '/usr/bin/md5sum -v' */*
  modoai_plugin jhove "/opt/jhove/jhove -c /opt/jhove/conf/jhove.conf -m jpeg-hul -h xml %s"
                "/opt/jhove/jhove -c /opt/jhove/conf/jhove.conf -h xml -v" "image/jpeg"
  modoai_plugin pronom_droid "/opt/jdk1.5.0_07/bin/java -jar
                /opt/droid/DROID.jar -L%s -S/opt/droid/DROID_SignatureFile_V12.xml"
                "/opt/jdk1.5.0_07/bin/java -jar /opt/droid/DROID.jar -V" "*/*"
  modoai_plugin exifTool "/usr/bin/exiftool -a -u %s" "/usr/bin/exiftool -ver" "image/jp*"
</Location>
```

Fig. 2. Portion of the mod_oai section of an Apache configuration file

4 Test Data

We ran multiple "baseline" requests to establish the response range of the server without any CRATE requests active, using JMeter (an Apache web server performance analyzer). The general resource distribution as a portion of overall web traffic is shown in Table 2. HTML and GIF files formed the core 85% of the requests, as would be characteristic of normal web traffic distribution. For the remaining 15%, we used a random-selection factor that is configurable in the JMeter application, which chooses one of the non-core resources at random from a list. Because of this random-resource selection, the throughput during each test varied slightly, from a high of 92.7 requests per second to a low of 80.1 requests per second. If the random resource was a large video ("wmv" file), the request rate would drop to the lower value, for example.

The "Response Time" columns do not show a consistent growth rate from 0% through 100% across all rows. From a performance testing perspective, the variation is essentially "in the noise." Differences up to a few seconds between columns may be due to any number of factors other than load alone. For example, the server may have been doing swap clean up or flushing logs. In some ways,

Table 2. Average distribution of hits (requests) per test run

Type	Avg Hits	Type	Avg Hits	Type	Avg Hits
mp3	312	html	238085	png	24296
jpeg	24316	pdf	3479	doc	717
ppt	1648	svg	456	txt	307
gif	792618	wmv	240		
Average Total Hits (per test):					**1,086,474**

having a busier server is more efficient because it is more likely that a resource which is about to be put through the metadata utility “wringer” will already be available in cache. However, site crawls - archival and otherwise - pull the full range of resources from the server, inevitably forcing some “swap” activity.

Web servers are more likely to be I/O bound than CPU bound, unless the server is also acting as an application or database server (WebSphere or MySQL, for example); the throughput reflects this I/O limitation. Even when mod_oai was building a full CRATE using all utilities, the server was able to provide 90% of the responses to regular web requests within 16 milliseconds. Little impact was seen to normal request servicing because typically very little CPU time is needed to serve up a web page. I/O-bound and CPU-bound services *can* co-exist without serious collision. Web servers will often have many “spare” CPU cycles that can be utilized by the metadata utilities without disrupting the I/O process of serving up web pages. In other words, even if a metadata utility is demanding a lot of CPU time, the web server can continue to deliver resources at a rapid rate to other users since it is not waiting for the CPU to be free, but is instead dependent on I/O availability.

5 Findings and Discussion

The test results in Table 3 show that even a modest web server can provide CRATE-type output without significantly impacting responsiveness. Table 3 compares the performance of the server in building the CRATE response when the various utilities are turned on or off. The fastest are the “native” utilities such as File and the Hashes. All of these have been in wide use and heavily optimized over the years, so this result is to be expected. Two of the Java utilities also performed well, despite not being server-based programs (JVM startup adds significant overhead to such a utility). Utilities are essentially additive, with processing time and file size growing in proportion to the number of utilities called.

Performance under most utilities was acceptably fast. The CPU power of our test web server is not particularly remarkable, but it never bogged down during the tests, except during Pronom-Droid activity. Droid increased the harvest time over 1,000%, and frequently drew 100% of CPU. We attempted to compile the utility, and briefly looked at its source code to see if we could spot some obvious problem (we didn’t). We also ran the utility in several other environments and found it to be similarly time-consuming on other systems. Droid does not appear to make external calls; i.e., no traffic went out of the server to any other site during its operation. At this point, we are unable to explain this phenomenon.

To return to the questions we posed in Section 3, our data indicates that it is safe to generate the metadata on the web server. We recommend that the configuration be tested before deployment, since a utility might have overly high CPU demand. We would not recommend using utilities that dramatically increase the total harvest time when compared with the time of a simple harvest. Webmasters should configure and test the response time for each utility and

Table 3. Web server performance for full crawl using a standard crawler (wget) versus OAI-PMH. ListIdentifiers returns only a *list* of resources (i.e., a kind of sitemap), not the resources themselves; ListRecords returns the resources and metadata.

Request Parameters	Active Utilities	Response Time in Min:Sec By Server Load			Response Size (Bytes)
		0 %	50 %	100%	
wget (full crawl)	None	00:27.16s	00:28.55s	00:28.89s	77,982,064
ListIdentifiers:oai_dc	None	00:00.14s	00:00.46s	00:00.20s	130,357
ListRecords:oai_dc	None	00:00.34s	00:00.37s	00:00.37s	756,555
ListRecords:oai_crate	None	00:02.47s	00:08.34s	00:03.38s	106,148,676
ListRecords:oai_crate	File	00:09.56s	00:09.72s	00:09.50s	106,429,668
ListRecords:oai_crate	MD5	00:04.55s	00:04.52s	00:04.40s	106,278,907
ListRecords:oai_crate	SHA	00:19.36s	00:19.70s	00:19.96s	106,190,722
ListRecords:oai_crate	SHA-1	00:04.57s	00:04.49s	00:05.37s	106,316,236
ListRecords:oai_crate	WC	00:06.14s	00:06.11s	00:05.92s	106,419,750
ListRecords:oai_crate	Exif	00:04.60s	00:04.79s	00:04.51s	106,163,645
ListRecords:oai_crate	DC	00:31.13s	00:29.47s	00:28.66s	106,612,082
ListRecords:oai_crate	OTS	00:35.81s	00:36.43s	00:35.83s	106,285,422
ListRecords:oai_crate	MetaX	01:13.71s	01:15.99s	01:13.96s	106,257,162
ListRecords:oai_crate	Jhove	00:54.74s	00:54.99s	00:54.84s	106,297,738
ListRecords:oai_crate	Droid	44:14.01s	45:29.76s	47:23.29s	106,649,382
ListRecords:oai_crate	All *but Droid*	03:34.58s	03:38.84s	03:42.60s	107,906,032
ListRecords:oai_crate	All	47:42.45s	48:53.97s	50:09.76s	108,407,266

monitor system performance to ee if problems occur, just as they do for other aspects of the web server.

Is it safe to ask for the metadata? A full CRATE harvest of a site produces a large response. The final size of the CRATE , 108 MB, was nearly 50% larger than the site itself. Utilities which produce more descriptive output than those used in our tests would obviously produce a larger result (and take longer to build). The harvest method used in our experiment is termed "By Value" because it retrieves the resources and the metadata. As such, it represents a worst-case approach. An alternative approach is to harvest the information "By Reference" which returns only the URI to the resource, not the Base64 encoding of the resource; the preservation metadata is still included by value in the CRATE. The resulting file, using our example test website, will be only about 8 MB instead of 108 MB. The harvester can combine this response with the results of a standard crawl, which may be a more efficient solution for both sides.

6 Conclusion and Future Work

It appears safe to both generate such metadata and to ask the web server for it, within certain parameters. We tried several types of plugins with mod_oai: Jhove, Metadata-Extractor, Open Text Summarizer, Hashes, and others. Anything that can run automatically is likely to be *compatible*, although utility speed and CPU demands ultimately determine whether a given utility is *feasible* or not. Scripts that further customize plugin usage can simplify installation without adding significant overhead. For example, Jhove has a number of analysis or "HUL" modules (ASCII, TIFF, JPEG, etc.) targeted to specific file types. Rather than create a dozen `<modoai_plugin>` sections, a shell script can pass the filename and the correct HUL to the utility and yet add near-zero overhead to the process. The disadvantage is less transparency within the web server's configuration file where the utility/shell script is called.

There are two points we would like to emphasize. First, the CRATE process is *fully automated* – the metadata is not validated by the web server nor by any other administrative action. Second, the metadata is generated *at time of dissemination*; it is not pre-processed nor canned. The metadata thus reflects the best-information available at that point in time. This approach harnesses the web server itself to support preservation, moving the burden from a single web-wide preservation master to individual web servers, where detailed information about the resource is most likely to reside. It also moves preservation metadata from *strict validation at ingest* to *best-effort description at dissemination.* In other words, the web server acts as its own agent of preservation by providing the crawler with sufficient information to assist the preservation process at the time the site is crawled.

Several of the non-native utilities are Open Source, and we believe it would be worthwhile to experiment with modifications to these utilities to see if performance can be improved. This can sometimes be achieved by using more efficient libraries, for example. Other utilities have some awkward usage requirements which could be tweaked for used with mod_oai. Pronom-Droid, for instance, does not allow a space to appear between the "-L" and the filename being called. This forced us to use a shell script to pass the variables to that utility. The penalty for this approach is effectively invisible: Doing the same "shell" method with Jhove introduced no change in utility completion time. Still, it is an annoyance and makes the Apache configuration file less immediately interpretable, because the shell script has to be consulted to see what is being called and how it is being used.

Java-based utilities seem to be at a disadvantage in general, because the natural optimization of the JVM does not occur in this type of situation. Compiling such utilities could help. We tried compiling Jhove (after the tests reported above) and found that it ran significantly faster. Droid, however, proved resistant to compilation, having numerous dependencies we could not readily resolve. We would be interested in how much improvement could be gained by compiling Java-based utilities, and suggest that developers consider providing an alternative distribution package containing all necessary dependencies. We are also interested in whether a rule of thumb could be devised to guide webmasters in selecting or configuring metadata utilities, and if such utility performance tuning would produce enough improvement to warrant the effort. Finally, websites vary greatly in size, type of content, and hardware configuration. A large-scale series of tests across many different websites would provide useful data regarding parameters for integrating preservation metadata utilities into the web server.

Acknowledgments. This research has been supported in part by the Library of Congress. We would also like to thank Jim Gray of Kronos Corporation and John Owen of Owenworks, Inc. for providing a commercial test bed and a load-testing framework.

References

[1] Nelson, M.L., Smith, J.A., Van de Sompel, H., Liu, X., Garcia del Campo, I.: Efficient, automatic web resource harvesting. In: 7th ACM WIDM, pp. 43–50 (November 2006)

[2] Smith, J.A., Nelson, M.L.: CRATE: A simple model for self-describing web resources. In: IWAW 2007 (June 2007)

[3] Lyman, P., Varian, H.R., Charles, P., Good, N., Jordan, L.L., Pal, J.: How much information? 2003. Research Project Report, U.C. Berkeley School of Information Management and Systems (October 2003), `http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/`

[4] Baeza-Yates, R., Castillo, C., Efthimiadis, E.N.: Characterization of national web domains. ACM TOIT 7(2) (2007)

[5] Levering, R., Cutler, M.: The portrait of a common HTML web page. In: ACM DocEng 2006, pp. 198–204 (October 2006)

[6] Fetterly, D., Manasse, M., Najork, M., Wiener, J.L.: A large-scale study of the evolution of web pages. Software: Practice & Experience 34(2), 213–237 (2004)

[7] Bent, L., Rabinovich, M., Voelker, G.M., Xiao, Z.: Characterization of a large web site population with implications for content delivery. In: WWW 2004, pp. 522–533 (December 2004)

[8] Ntoulas, A., Cho, J., Olston, C.: What's new on the web?: The evolution of the web from a search engine perspective. In: WWW 2004, pp. 1–12 (December 2004)

[9] Cherkasova, L., Karlsson, M.: Dynamics and evolution of web sites: Analysis, metrics, and design issues. In: IEEE ISCC, pp. 64–71 (July 2001)

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

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

[12] Bekaert, J., De Kooning, E., Van de Sompel, H.: Representing digital assets using MPEG-21 Digital Item Declaration. Int. J. Digit. Libr. 6(2), 159–173 (2006)

[13] Scott, M.: Wordsmith software package. Oxford University Press, Oxford (2008), `http://www.lexically.net/wordsmith/`

[14] Hardy, D.R., Schwartz, M.F.: Customized information extraction as a basis for resource discovery. ACM Trans. Comput. Syst. 14(2), 171–199 (1996)

[15] Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: practical automatic keyphrase extraction. In: ACM DL 1999, pp. 254–255 (August 1999)