Stress-Testing General Purpose Digital Library Software

David Bainbridge[1], Ian H. Witten[1], Stefan Boddie[2], and John Thompson[2]

[1] Department of Computer Science
University of Waikato
Hamilton, New Zealand
{davidb,ihw}@cs.waikato.ac.nz
[2] DL Consulting Ltd
Innovation Park
Hamilton, NZ
{stefan,john}@dlconsulting.com

Abstract. DSpace, Fedora, and Greenstone are three widely used open source digital library systems. In this paper we report on scalability tests performed on these tools by ourselves and others. These range from repositories populated with synthetically produced data to real world deployment with content measured in millions of items. A case study is presented that details how one of the systems performed when used to produce fully-searchable newspaper collections containing in excess of 20 GB of raw text (2 billion words, with 60 million unique terms), 50 GB of metadata, and 570 GB of images.

1 Introduction

Today we are witnessing a great upsurge in nationally funded digital library projects putting content on the web. As the volume of data to be stored in these repositories increases, the question of the scalability of the software used becomes crucial. The focus of this paper is to detail and assess what is known about three widely used open source general purpose digital library systems: DSpace, Fedora and Greenstone.

The structure of the paper is as follows. First we briefly describe the three pieces of software under review, with emphasis on the core technologies they are based on, before discussing in turn known results (from a variety of sources) of scalability testing. Then we take, as a case study, the work undertaken in building newspaper-based digital library systems for the National Libraries of New Zealand and Singapore. These collections currently contain over half a million OCR'd items each, and are on track to be scaled up to twice and four times their size, respectively.

2 Background

DSpace is a collaborative venture between Hewlett Packard and MIT's Library, heavily optimized for institutional repository use [6]. Ready to use out of the box, it has been widely adopted for this purpose. It is written in Java, and is servlet based, making extensive use of JSP. Tomcat is recommended by its developers with either PostgreSQL or Oracle as the relational database management system. Other RDBMS can be used with it (such as MySQL) through JDBC.

It can be configured to support full-text indexing (utilizing Lucene) for Word, PDF, HTML and plain text. Documents are either ingested individually through the web (intended for author submission), or locally (i.e. on the server where DSpace is installed) through batch processing by command-line scripts into a collection. A DSpace repository is represented as a hierarchy of communities and sub-communities, with collections forming the leaves to this hierarchy.

Fedora is founded upon a powerful digital object model, and is extremely flexible and configurable [1,4]. A repository stores all kinds of objects, not just the documents placed in it for presentation to the end-user. For example a Service Definition (or SDef) object defines a set of abstract services and the arguments they take, which can then be associated with a document in the repository to augment its behavior; a Service Deployment (or SDep) object provides a concrete implementation of an SDef. Ingesting, modifying and presenting information from this rich repository of objects is accomplished through a set of web services. These are grouped by function into two APIs, one for access and the other for management. Both "lite" (RESTful) and a full (SOAP-based) versions of the APIs are available.

Documents are ingested either using command-line scripts or the Fedora Administration tool, which is a Java-based application. No matter which option is used, ultimately the data is passed through the management API (which includes authentication), and so these tools function equally well remotely as they do locally to the Fedora installation. Documents must be encoded in XML in either FOXML, the Fedora extension to METS, or ATOM.

Like DSpace, the core system is written in Java and makes use of Servlets. Again a relational database is part of the mix. Fedora ships with McKoi—a light-weight pure Java implementation—and it can also be set up to work with MySQL, DB Derby, Oracle 9 and PostgreSQL. Full-text indexing is possible through a third-party package (GSearch) into which Lucene, Zebra or Solr indexing packages can be plugged.

Unlike DSpace, Fedora is not a turn-key software solution, and software development is necessary to shape it into the desired end product. In the case where the shaping is of interest to others, it is of course possible to package this up and make it available to give a more ready-to-run solution. For the main part, these additions sit on top of the Fedora architecture are of secondary importance in terms scalability issues. Of primary importance is how well the core repository handles large volumes of data, typically in the form of documents and metadata, but also the expressed relationships between objects. In Fedora, such relationships are handled by a triple-store.

In terms of a design philosophy, Greenstone sits between DSpace and Fedora. The standard download is a turn-key solution, however it includes several alternative software components (with different strengths and weaknesses) that are the software equivalent of being "hot swappable." The ingest phase to Greenstone is written in Perl, with the delivery mechanism implemented in C++ and executed as a CGI program using a standard web server.

A system of document parsing plugins allows for a extensive range of file formats to be full-text indexed, along with the automatic extraction (and manual assignment) of metadata: plain text, e-mail messages, Postscript, PDF, HTML, Word, PowerPoint, Excel, and OpenOffice documents are just some of the formats supported. Records from metadata-only formats can be treated as documents or bound to their source document counterpart if present: Greenstone processes metadata in MARC, Bibtex, Refer (EndNote), Dublin Core and LOM (Learning Object Metadata) formats, amongst others.

As mentioned above, the database and indexing systems are swappable components. MG, MG++ and Lucene are available for full-text indexing; GDBM and SQL-lite for the metadata database. MG's strength is that compression is built into the indexing technique, but it is not incremental [8]. This contrasts with Lucene, which can support incremental building but does not include compression. In designing a collection, a digital library can choose (literally with the click of a button) to trade the amount of space needed to represent the built collection against speed of rebuilding, or *vice versa*, as the needs of the project dictate.

The principal requirement for the database system used by Greenstone is that it supports *get-record* and *set-record* operations. The most basic of flat-file database management systems provides this, and they are trivial operations for a relational database system to support. Greenstone ships with both the GDBM (flat-file) and SQL-lite (relational) database systems.

Greenstone also supports web-based submission of documents (aka DSpace) and a graphical tool called the Librarian Interface that can be configured for local or remote access (aka Fedora). All three systems provide the option of running command-line scripts.

2.1 Discussion

The different aims of these three software tools manifest themselves in the decisions made in developing the underlying software architecture, and how it is deployed. It is also the case that there are significant differences seen in the web technologies used by Greenstone and the other two. This stems from the fact that one of Greenstone's important design requirements is to run on early versions of Windows (right down to Windows 3.1) for deployment in developing countries [7].

Greenstone in fact comes in two flavors: Greenstone 2 is the production version that is described above, and is what is assessed here; Greenstone 3 is a more recent development that is structured as a research framework. It is written in Java, makes use of web services and servlets, is backwards compatible with Greenstone 2, and has optional support for a relational database system. In terms of applicability of this paper, scalability findings from DSpace and

Fedora that stress-testing these shared web technologies are broadly applicable to Greenstone 3; furthermore, despite the different web technologies used between Greenstone 2 and the other two DL systems, there are many parallels as to how the ingest process is performed, particularly with the command-line scripts. Many of the lessons learnt in the case study of building a newspaper DL using Greenstone are applicable. We return to this point in our conclusion.

3 Stress-Testing

We now review known results from testing the scalability of Fedora, DSpace and Greenstone.

3.1 Fedora

The first known stress-testing of Fedora was by the University of Virginia Library in 2001. Still in an embryonic form compared to the Fedora Project as we know it today, Virginia worked from the reference implementation Cornell had produced, and over a period of two years modified it for use in a web environment. For performance reasons, it was at this point that the decision to incorporate a relational database was made. A testbed of half a million heterogeneous digital objects—consisting of images, scientific data and a variety of XML encoded documents, such as e-texts—was established and performance of the digital library evaluated.

Using a Sun Ultra80 dual-processor as the server, a laptop in another location of the campus was used to simulate 20 simultaneous users accessing the repository. The average response time of the server was approximately half a second. By synthetically replicating objects the repository was grown to 1 million and then 10 million items, with the same experiment repeated at each stage. By the time there were 10 million objects stored in the repository the server response time was found to be between 1–2 seconds.

In 2004 The National Science Digital Library (NSDL) took the decision to base its content management system on Fedora. A factor in it being chosen was its established credentials with regard to scalability through the work of Cornell and Virginia. By 2007 NSDL had grown to the point where the repository stored over 4.7 million objects with 250 million triples representing the various relationships expressed between objects. The latter was causing problems with the component they had selected to support this aspect of the architecture, Kowari, and it motivated the development of MPTStore[1] as an alternative solution that had better scalability characteristics.

A recent and more comprehensive project to study the scalability of Fedora is underway at FIZ Karlsruhe in Germany. This project is assessing a variety of aspects of configuring a high-volume Fedora repository. Before scaling up to 14 million items, a fine-grained analysis was conducted on a test set of 50,000 documents. Parameters studied included:

[1] http://mptstore.sourceforge.net

– Different versions of Java, and tuning components of the JRE to control garbage collection, heap size, etc.
– Different database and triple-store backends (such as MPTStore and Mulgura), including examples of using them with default values and fine-tuning their settings.
– Remote versus local access, and internally versus externally managed content.

Ingesting 14 million items took three weeks, using a dual core 2.4 GHz Intel processor with 2 GB of RAM, and resulted in 750 million triple-store items. A small number of documents failed to be ingested; the researchers are still investigating why. Some noteworthy recommendations from the testing to date are:

– There are efficiency advantages to using Java 1.6 over 1.5.
– Mass ingest using a remote connection to the relational database is significantly faster (by a factor of 2) than the same operation performed locally.
– It is worth fine-tuning the configuration of the database and triple-store, but unfortunately most adjustments lower the ability to recover from errors.

It is perhaps initially surprising that a remote database was faster than a local one. This is attributed to the streamlining of disk IO when undertaken remotely. Database updates no longer have to compete with disk activity associated with reading the source documents.

3.2 DSpace

The U.S. National Library of Medicine has developed a digital repository called SPER (System for the Preservation of Electronic Resources), based on DSpace. It uses MySQL for the relational database and includes an enhanced command-line ingest facility. In 2007 they undertook a detailed study of DSpace designed to stress test it in this configuration [3].

To form a testbed, they combined two collections on Food and Drug-related Notices Of Judgment to form a new collection containing nearly 18,000 documents. Each document comprises TIFF image data, OCR'd text and descriptive metadata. The majority of documents (75%) contain 1–2 pages; the longest one has 100 pages. These files were then replicated to form a collection of over a million (1,041,790) items, and located in a hierarchy of 32 communities, consisting of 109 collections.

Ingesting the testbed took slightly over 10 days using a Sun Microsystems X4500 server with a dual core 2.8 GHz processor. The version of Java used was 1.4. The files were processed without error, and the experiment repeated a second time to gather more statistics. A prominent result found was that the ingest time per document increased roughly linearly with the size of the collection. Ingesting their sample documents into an empty repository took approximately 0.2 seconds on average. With 1 million items in it, the average ingest time had risen by a factor of 7 to around 1.4 sec. The overall conclusion was that performance was satisfactory at this scale of operation.

3.3 Greenstone

Initial scalability testing on Greenstone was conducted by its developers (also authors of this paper) during 1998–1999 under Linux. Using real-world data, in one case a collection was formed from 7 GB of text, and in another (a union catalog), 11 million short documents were used (about 3 GB text). Both examples used MG for indexing (this was the only indexing tool supported at that point) and built (first time) without any problems.

In these tests, we did not attempt to use a larger volume of data as there was nothing appropriate available at the time. Using 7 GB twice over to make 14 GB does not really test the scalability of the indexing tool because the vocabulary has not grown accordingly, as it would with a real collection. The issue of vocabulary size is particular acute when working with OCR'd text, a point we return to in the case study below.

Independent stress testing of Greenstone was carried out in 2004 by Archivo Digital, an office associated with the Archivo Nacional de la Memoria (National Memory Archive), in Argentina. Its test collection contained sequences of page images with associated OCR text. They used an early version of Greenstone (2.52) on an 1.8 GHz Pentium IV server with 512 MB RAM running Windows XP Professional. There were 17,655 indexed documents, totaling 3.2 GB of text files, along with 980,000 images in TIFF format. Full-text indexes were built of both the text and the titles.

They found that almost a week was spent collecting documents and importing them. (No image conversion was done.) Once the import phase was complete, it took almost 24 hours to build the collection. The archives and the indexes were kept on separate hard disks to reduce the overhead that reading and writing from the same disk would cause.

3.4 Discussion

Both Greenstone and Fedora undertook scalability testing early on, and it is fair to say that, given the available technology at the time, both performed well. Since DSpace is offered as a turn-key solution, there is evidence of greater expectations within the community that deploys this software that is should simply just continue to work as the volume grows.[2] This contrasts with Fedora's design philosophy, where an organization is expected to undertake their own IT development to get going with it, and therefore (as with the case of MPTStore) there is a greater willingness to adjust the installation when things do not work out as expected.

Greenstone sits somewhere between these two projects. It has a pluggable indexer and database infrastructure, and includes several options as part of the standard install. If one particular technology hits a limit it is easy to switch to an alternative. For example, MG's Huffman coding calculations overflow at around 16 GB of text, however if this poses a difficulty, it is trivial to switch the

[2] http://wiki.dspace.org/index.php/ArchReviewIssues

collection to use Lucene instead (which does not have this restriction). There are always trade-offs, of course: a Greenstone collection based on Lucene does not support case-sensitive matching, whereas ones based on MG and MG++ do.

3.5 Comparable Testing

Scalability tests have been conducted by different groups at different times on different content, and so it is difficult to compare the results produced by the different systems directly. Consequently, we installed DSpace (version 2.1), Fedora (version 3.1) and Greenstone (version 2.81) on the same computer and compared them ingesting the same set of files: 120,000 newspaper articles comprising 145 GB of images and OCR'd text. (The files in question are an excerpt that comprises about 2% of the scanned newspaper collection described in the next section.)

The workstation used was a dual-processor 3 GHz quad-core Xeon processor with 4 GB of memory. Rounding to the nearest minute, the processing time for the three systems was surprisingly close: Greenstone (using Perl 5.8) took 35 minutes; DSpace (using Java 1.5) took 34 minutes; and Fedora (same version of Java) took 78 minutes. In all cases the digital library server was run locally to where the source documents were located. With a server that was remote from the source content, the processing time for Fedora could perhaps be halved to around 37 minutes due to the effect noticed by FIZ Karlsruhe (noted above).

4 Case Study

We now present, as a case study, the construction of a large collection in Greenstone. This contains approximately 1.1 million digitized pages from national and regional newspaper and periodicals spanning the years 1839–1920, and was undertaken by DL Consulting for the National Library of New Zealand. Over half (69%) of the images in the collection have been OCR'd, comprising nearly 20 GB of raw text: 2 billion words, with 60 million unique terms that is full-text searchable. This number will increase over the years until all images are fully searchable.

Resulting from the OCR'd data, the collection consists of 8.3 million searchable newspaper articles, each with its own metadata (much of it automatically generated). The total volume of metadata is 50 GB—three times as much as the raw text! A large part of this is information that records the location of each word in the source image, along with coordinate information for each article. This is used in the digital library interface to highlight search terms in the images and to clip individual articles out of the newspaper pages. Before being built into a digital library collection the metadata is stored in an XML format, which occupies around 600 GB, slightly less than 1 MB per newspaper page. The result of this work, known as Papers Past, is available on-line and can be can be viewed at *http://paperspast.natlib.govt.nz*.

4.1 Building the Papers Past Collection

Text. About 820,000 pages have been processed by optical character recognition software to date (early 2009) for the Papers Past collection. These contain 2.1 billion words; 17.25 GB of raw text. The images were digitized from microfilm, but the paper originals from which the microfilm was obtained were of poor quality, which inevitably resulted in poor performance of the OCR software.

The high incidence of recognition errors yields a much larger number of unique terms than would normally be expected. The 2.1 billion words of running text include 59 million unique terms (2.8%). Our tests show that this will continue to increase linearly as further content is added. By contrast, clean English text typically contains no more than a few hundred thousand terms, and even dramatic increases in size add a relatively small number of new terms. As a point of comparison, the Google collection of n-grams on the English web[3] is drawn from 500 times as many words (1,025 billion) but contains less than a quarter the number of different words (13.6 million, or 0.0013%). However, words that appear less than 40 times were omitted from this collection, a luxury that we did not have with Papers Past because of the importance of rarely-used place and personal names for information retrieval.

Enormous vocabularies challenge search engine performance [8]. Moreover, the high incidence of errors makes it desirable to offer an approximate search capability. Neither MG nor MG++ provide approximate searching, so it was decided to use the Lucene indexer because of its fuzzy search feature and proven scalability—it has been tested on collections of more than 100 GB of raw text. Consequently we worked on improving and extending the support for Lucene that Greenstone provides.

Metadata. Papers Past involves a massive amount of metadata. Its specification demanded that newspaper articles be viewable individually as well as in their original context on the page. The 820,000 OCR'd pages comprise 8.3 million individual articles, each with its own metadata. To meet the specification, the physical coordinates that specify an article's position on the page were stored as metadata, which is used to clip the article-level images from pages at runtime.

A further requirement was that search terms be highlighted within page images. In order to do so, the bounding-box coordinates of each and every word in the collection must be stored. These word coordinates represent 49 GB of metadata. Putting together all the article, page, and issue information yielded a total of 52.3 GB of metadata.

The collection's source files are bi-tonal digital images in TIFF format. From these, the OCR process generates METS/ALTO XML representation [2]. This includes all the word and article bounding-box coordinates, as well as the full text, article-level metadata and issue-level metadata. The resulting source data includes 137,616 METS files, one per newspaper issue, and 8,381,923 ALTO files, one per newspaper page. Together these amount to a total of 570 GB of

[3] The Google n-gram collection is available on six DVDs from http://www.ldc.upenn.edu/

XML, slightly under 1 MB per page. All this XML is imported into Greenstone, which indexes the text with Lucene and stored the metadata and bounding-box coordinates in a database.

From the very beginning, Greenstone has used the GNU database management system (GDBM) for storing and retrieving metadata. Alternative database backends were added later. GDBM is fast and reliable, and does this simple job very well. Crucially, and of particular importance for librarian end-users, GDBM can be installed on Windows, Linux and Macintosh computers without requiring any special configuration. However, the design of Papers Past exposed limitations. GDBM files are restricted to 2 GB; moreover, look-up performance degrades noticeably once the database exceeds 500 MB.

A simple extension was to modify Greenstone to make it automatically spawn new databases as soon as the existing one exceeds 400 MB. Currently, Papers Past uses 188 of them. With this multiple database system Greenstone can retrieve word coordinates and other metadata very quickly, even when running on modest hardware.

Images. The archival master files for Papers Past are compressed bi-tonal TIFF images averaging 770 KB each. The full collection of 1.1 million images occupies 830 GB.

A goal of the project is that no special viewer software is required other than a modern web browser: users should not need browser plug-ins or downloads. However, TIFF is not supported natively by all contemporary browsers. Consequently the source images are converted to a web friendly format before being delivered. Also, processing is required to reduce the image file size for downloading, and in the case of individual articles, to clip the images from their surrounding context.

Greenstone normally pre-processes all images when the collection is built, stores the processed versions, and serves them to the user as required. However, it would take nearly two weeks to pre-process the 1.1 million pages of Papers Past, at a conservative estimate of 1 page/sec. To clip out all 8.3 million articles and save them as pre-prepared web images would take over 3 months, assuming the same rate. The preprocessed images would consume a great deal of additional storage. Furthermore, if in future it became necessary to change the size or resolution of the web images they would all need to be re-processed. Hence it was decided to build an image server that converts archival source images to web accessible versions on demand, and maintains a cache of these for frequently viewed items.

A major strength of Greenstone is its ability to perform well on modest hardware. The core software was designed run on everything from powerful servers to elderly Windows 95/98 and even obsolete Windows 3.1/3.11 machines, which were still prevalent in developing countries. This design philosophy has proven immensely valuable in supporting large collections under heavy load. Greenstone is fast and responsive. On modern hardware, it can service a large number of concurrent users. However, the image server is computation-intensive and consumes significant system resources, though the overall system can cope with moderately heavy loads on a single modern quad-core server. (At the New Zealand National Library it is run on a Sun cluster.)

Building the collection. It takes significant time to ingest these large collections into Greenstone, even without the need to pre-process the source images. Greenstone's ingest procedure consists of two phases. The first, called importing, converts all the METS/ALTO data into Greenstone's own internal XML format. The second, called building, parses the XML data and creates the Lucene search index and the GDBM metadata databases.

The import phase processes approximately 100 pages/min on a dual quad-core Xeon processor with 4 GB of main memory, taking just over four days to import 820,000 pages. This stage of the ingest procedure may be run in batches and spread over multiple servers if necessary. The building phase processes approximately 300 pages/min on the same kind of processor, taking around 33 hours to build the collection. The latter step is incremental, so when new content is added (or existing content is altered) only the affected items need processing.

Future improvements. The problem of large-scale searching is exacerbated by the presence of OCR errors. An obvious solution is to remove errors prior to indexing [5]. However, we decided not to attempt automatic correction at this stage, in order to avoid introducing yet more errors. In our environment this solution is workable so long as Lucene continues to perform adequately on uncorrected text. However, we do plan to investigate the possibility of eliminating the worst of the OCR errors.

Singapore National Library has embarked on a comparable newspaper project using Greenstone. This currently contains approximately 600,000 newspaper pages, and will grow to more than two million (twice the size of Papers Past) The Singapore collection uses grayscale JPEG-2000 master source files, which average 4.5 MB per page; the existing 600,000 pages consume 2.6 TB of storage.

4.2 Apache Solr

We have conducted some indicative experiments with Apache Solr. Solr is based on Lucene, crafted to optimize performance in a web environment. It features faceted search, caching and a mechanism for distributed indexes, along with convenient API access from Java, JavaScript, PHP, Ruby and other web-integration-friendly programming languages.

To trial this indexing technology we combined the OCR'd text and metadata from the New Zealand and Singapore National Libraries to produce a total of nearly 17 million articles (16,901,237). Running on a 2.33 GHz dual-core Xeon processor, it took 30 hours (wall-clock time) to index the files. The server is shared by others; however the build was done over the weekend and consequently the machine was lightly loaded.

Comparing query times between Lucene and Solr, on this size of collection Lucene took between 7–10 seconds (processor time) to produce the first 500 matching documents. This range factors in a caching effect of files through the operating system that was observed, which resulted in a saving of approximately

2 seconds after the initial query. For the equivalent queries using Solr, query times were in the range 0.24–0.33 seconds—over 20 times faster. This stark difference is all the more impressive when it is remembered that Solr is using essentially the same indexing technique, only better tuned. For example, one thing it does is run a daemon process to respond to queries, rather than perform each query in isolation.

5 Summary and Conclusions

This article has described a range of scalability tests applied to the general-purpose open source digital library software solutions DSpace, Fedora, and Greenstone. Indicative scalability tests where conducted by the developers of Greenstone and Fedora early in their development history. Other than taking a long time to ingest, these tests worked flawlessly on test sets that exceeded 10 million items.

The Fedora project synthesized a 10 million item collection from a starting set of documents that numbered 500,000. The test looked at the server response time from simulated user requests for content. As the size of the repository grew, response time slowed from 0.5 to 1–2 seconds, still within acceptable bounds. There was no mention of full-text indexing being tested at that point, but that is only to be expected, because—to the best of the authors' knowledge—support for this was not added until some years later. Eight years on, with the software in a considerably more mature form, FIZ Karlsruhe has undertaken a comprehensive program of stress testing this software. Again, replicated data was used to produce a high volume testbed (14 million items).

DSpace has been stress-tested with 1 million documents; again replication was used to produced the test set. With full-text indexing becoming a standard feature of digital library software, some caution is necessary when interpreting results from scalability tests when replicated data is used. Vocabulary size does not grow the way it would in a real collection, and consequently the indexing component is not placed under as much pressure as it would under normal conditions.

With Greenstone, which has supported full-text search from the outset, testing has focused on high-volume real-world data. As with Fedora, testing was undertaken early in the development process, and tests with 11 million items and 7 GB of text helped confirm the viability of the software infrastructure. Ten years on, Greenstone's scalability has recently been assessed through substantial national newspaper projects in New Zealand and Singapore. Each contains over half a million OCR'd pages (text and images) and are on track to be scaled into the multi-million range.

As a case study, this paper has presented details of building the Papers Past collection for the New Zealand National Library. The advertised accuracy of OCR software is around 99.9%, but this is at the character level and assumes high quality images. The error rate at the word level is much higher. The Papers Past images were a century old and were digitized from microfilm and microfiche. The image quality is poor, which compounds the problem of errors. Our experience with this OCR text is that vocabulary size grows linearly with collection size.

This is potentially a severe stress point in any digital library system that offers full-text indexing.. However, for the current (and projected) scale of operation the indexing software coped admirably.

To operate at this scale, Greenstone software developments were modest. First, the existing multi-indexer framework was fine-tuned to increase its support for Lucene. Second, the standard flat-file database (GDBM) was extended to support multiple instances, to overcome the performance degradation that occurred when database files exceeded 500 KB. Testing showed that this arrangement more than adequately satisfied the needs of these projects.

The two newspaper projects helped identify some bottlenecks in the building (and rebuilding) of collections, and these were addressed as part of this work. Greenstone's importing phase is particularly amenable to being distributed across several servers. Moreover, Greenstone supports a system of "plugouts" that allows this phase of the operation to output to other formats such as FedoraMETS and DSpace. Greenstone's importing can be fed directly into the ingest processes of DSpace and Fedora, thereby passing the benefits of the improvements made to Greenstone on to these projects as well.

References

1. Lagoze, C., Payette, S., Shin, E., Wilper, C.: Fedora: an architecture for complex objects and their relationships. International Journal on Digital Libraries 6(2), 124–138 (2006)
2. Littman, J.: Technical approach and distributed model for validation of digital objects. D-Lib Magazine 12(5) (2006)
3. Misr, D., Seamans, J., Thoma, G.R.: Testing the scalability of a DSpace-based archive. Technical report, National Library of Medicine, Bethesda, Maryland, USA (2007)
4. Payette, S., Lagoze, C.: Flexible and extensible digital object and repository architecture (FEDORA). In: Nikolaou, C., Stephanidis, C. (eds.) ECDL 1998. LNCS, vol. 1513, pp. 41–59. Springer, Heidelberg (1998)
5. Reynaert, M.: Non-interactive OCR post-correction for giga-scale digitization projects. In: Gelbukh, A. (ed.) CICLing 2008. LNCS, vol. 4919, pp. 617–630. Springer, Heidelberg (2008)
6. Smith, M., Bass, M., McClella, G., Tansley, R., Barton, M., Branschofsky, M., Stuve, D., Walker, J.: DSpace: An open source dynamic digital repository. D-Lib Magazine 9(1) (2003), doi:10.1045/january2003-smith
7. Witten, I.H., Bainbridge, D.: A retrospective look at greenstone: lessons from the first decade. In: JCDL 2007: Proceedings of the 2007 conference on Digital libraries, pp. 147–156. ACM Press, New York (2007)
8. Witten, I.H., Moffat, A., Bell, T.C.: Managing gigabytes: compressing and indexing documents and images, 2nd edn. Morgan Kaufmann, San Francisco (1999)