Coping with very large digital collections using Greenstone

Stefan Boddie,† John Thompson,† David Bainbridge‡ and Ian H. Witten‡

†DL Consulting Ltd
Innovation Park
Hamilton, New Zealand
{Stefan, john}@dlconsulting.com

‡Department of Computer Science
University of Waikato
Hamilton, New Zealand
{davidb, ihw}@cs.waikato.ac.nz

Abstract. The Greenstone digital library software is widely used for small to medium digital library collections, but its reputation for creating very large collections is less well established. This paper describes how Greenstone is being used to produce large newspaper collections for the National Libraries of New Zealand and Singapore, respectively. It also describes current developments that integrate IBM's DB2 database system into Greenstone as an optional search engine and metadata database, which allows the runtime server to be deployed in a federated configuration.

1 Introduction

Greenstone is a suite of software for building and distributing digital library collections. It is not a digital library but a tool for building digital libraries. Developed and distributed in cooperation with UNESCO, it runs on all popular operating systems (even the iPod!). For more details see [1] and the project website [2].

The Greenstone Digital Library Software has been distributed under the GNU General Public License for a dozen years, during which time it has developed almost beyond recognition. Today the user base hails from 70 countries and the reader's interface has been translated into over 45 languages. Downloads have exceeded 4,500 times a month for many years. It is fair to say that this software has materially helped spread the practical impact of digital library technology throughout the world. Training courses have been offered throughout both the developing and developed worlds [3].

Because of the emphasis on training, most Greenstone collections are rather small, because educational exercises use a few documents to a few hundred. Some trainees return to their home institutions to work on large-scale projects, but many never progress beyond toy collections. Unfortunately this wide accessibility has earned Greenstone a reputation of being suitable for small collections only. This is incorrect.

It is probably unrealistic to expect a general-purpose piece of software, suitable for a wide variety of users and applications, to vie with a purpose-built system designed specifically for a very large information collection (like, say, a web search engine). However, Greenstone can be pushed reasonably far in the size of collections that can be built. This article reports on work by DL Consulting, a company that specializes in digital library solutions using Greenstone.

DL Consulting recently built a large Greenstone-based digital library for the National Library of New Zealand containing over 1 million pages of historic newspapers. Over half (60%) of these have been OCR'd, comprising nearly 20 GB of raw text: 2 billion words, with 60 million unique terms that is full-text searchable. When the project is complete, all images will be fully-searchable. The collection consists of over 6.5 million newspaper articles, each with its own metadata (much of it automatically generated); and the total volume of metadata is 50 GB—three times as much as the raw text! Before being built into a digital library collection the metadata is stored in XML format, which occupies around 600 GB, slightly less than 1 MB per newspaper page. A similar system is underway for Singapore's National Library Board, and this is projected to grow to four times the size of the fully realized NZ collection.

Currently the National Library of Singapore implementation runs on a single machine. For full scalability it will be necessary to distribute the system over multiple servers. This is already possible to a certain extent by structuring text and image capabilities as separate servers, but further growth will require distributing the search index and metadata database. Greenstone already includes the possibility of incorporating different indexers: two indexers (MP and MPGG) are included as standard, and a third (Lucene) is available but experimental. Through this work Lucene has now been upgraded to become a standard option. Additionally we are evaluating the performance of a fourth, IBM's DB2 (which is able to serve as a metadata database as well) and has the ability to be distributed over multiple servers.

This article reports on these two developments. We begin by describing the historic newspaper project. Then we discuss the issues involved in incorporating DB2 into Greenstone as an optional indexer and metadata database, and present the results of preliminary experiments.

2 Building the *Papers Past* collection

The *Papers Past* collection of the NZ National Library, which can be viewed at http://paperspast.natlib.govt.nz, contains approximately 1.1 million pages of historic newspapers. A similar collection has been built for the National Library Board of Singapore. This currently contains approximately 600,000 newspaper pages, and will grow to more than two million. This section describes *Papers Past*, but except where noted the structure of the Singapore version is essentially identical. The chief difference is the expected growth.

Text. About 600,000 pages have been processed by optical character recognition (OCR) software to date for the *Papers Past* collection. These contain 2.1 billion words; 17.25 GB of raw text. The images were digitized from microfilm, but were of poor quality—as were the paper originals—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 [4] 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 [5]. Moreover, the high incidence of errors makes it desirable to offer an approximate search capability. Unfortunately neither MG nor MGPP provide approximate searching. Instead we 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 experimental support for Lucene available through Greenstone.

Metadata. *Papers Past* involves a massive amount of metadata. Its specification demands that newspaper articles be viewable individually as well as in their original context on the page. The 600,000 OCR'd pages comprise 6.5 million individual articles, each with its own metadata. The physical coordinates that specify an article's position on the page are stored as metadata to allow article-level images to be clipped from pages.

A further requirement is that search terms be highlighted directly within 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—nearly three times as large as the collection's full text. Putting together all the article, page, and issue information yields a total of 52.3 GB of metadata.

The collection's source files are bi-tonal digital images in TIFF format. From these images, the OCR process generates METS/ALTO XML representation [6]. 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 include 91,545 METS files, one per newspaper issue, and 601,516 ALTO files, one per newspaper page. Together, these amount to a total of 570 GB of XML, slightly under 1 MB per page. All this XML is imported into Greenstone, the text is indexed with Lucene, and the metadata and bounding-box coordinates are stored by Greenstone in a database.

From the very beginning, Greenstone has used the GNU database management system (GDBM) for storing and retrieving metadata. It 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. 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.

A goal of both projects is that no special viewer software is required other than a modern web browser: users should not need browser plug-ins or downloads. However, neither TIFF nor JPEG 2000 are 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 image file size for download, 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 6.5 million articles and save them as pre-prepared web images would take a further 15 days, 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 to convert archival source images to web accessible versions on demand, and maintain 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 and, on modern hardware, 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, as is deployed at the National Library of Singapore.

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, canonical 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 600,000 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. Currently, the second step cannot be executed incrementally, and the complete collection must be re-built (but not re-imported) whenever data is added.

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 [7]. 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.

A long-standing deficiency of Greenstone is the need to rebuild collections from scratch when documents are added, modified, or deleted. This limitation arose because the original MG indexer was non-incremental—it was optimized for maximum compression, which requires non-incremental operation to ensure that the word statistics on which compression is based reflect the collection as a whole. However, Lucene, which Greenstone now incorporates as an option, is capable of operating incrementally. Today, the only bar to incremental operation concerns the way that the GDBM metadata database is updated and accessed. We are working to allow this to be easily replaced by alternative databases, just as users can select MG, MGPP, or Lucene as indexers when building a collection. As part of this work we will make Greenstone collections updatable incrementally, avoiding the necessity to rebuild the search index and metadata database as the collection evolves.

This will improve scalability of the building process. However, the only way to obtain arbitrary scalability of the run-time system is to distribute it over multiple servers. It is already possible for the image server and the core Greenstone system to run on separate computers; indeed, multiple image servers can easily be used. But for true scalability the search index and metadata databases must also be distributed.

3 Distributed operation with IBM's DB2

The research version of Greenstone [8] includes a sub-collection facility that allows users to search in a seamless way across sub-collections running on different servers. However, this facility is limited and does not fully support distributed operation; in particular it does not allow distributed metadata databases to be accessed in a simple and uniform way. Buried deep in the code are structures that could be extended to improve support for this. However, we are taking a different approach.

We are evaluating the performance of IBM's DB2 platform as both indexer and metadata database for Greenstone. When designing their collections, users will have the option of using this third-party software instead of the default document index and

metadata database. DB2 has known scalability and distributed functionality: built into it is the ability to distribute a single database over multiple servers.

The DB2 database. DB2 is a mature, enterprise-scale, database application developed by IBM [9,10]. Implemented as a client/server architecture, a single instance of the DB2 server can scale to support 3.5 terabytes of database with little loss in performance [11]. More importantly, multiple DB2 servers can be federated, with a single front-end providing transparent access to a farm of databases [12]. By augmenting Greenstone with an underlying DB2 database we have been able to create distributed indexes using this built-in federation technology.

We needed to rationalize the use of a commercial module in an open source system intended for wide distribution. An important factor is the availability of a free version of this software. The DB2 C-Express database has the same basic functionality as the commercial version, but with a restriction on the number of servers [13]. It also allows several key text search components to be integrated, including the Net Search Extender module, which allows full text, wildcard and fuzzy searching [14]. We cannot redistribute this software because it is not issued under the GPL license, but we can make it easy for users to download it themselves and hook it into Greenstone.

Implementation. Greenstone is written in a modular fashion. As noted earlier, it is already possible to switch between different full-text indexers. We needed to extend this modularity to encompass the metadata database too. In order to do this, modifications were made to both the index-building and server modules. The build-time code, written in PERL, is responsible for extracting text and metadata from source documents and building them indexes. The server code, written in C++, retrieves information from the index—possibly based upon some search criteria—and presents it to the user as HTML web pages. Both stages were integrated with DB2 through abstraction layers for the two programming languages; Class:DBI [15] for PERL and the DBConnect API [16] for C++. Finally, a sample collection was built and tested for functionality—using a DB2 installation that was itself a front-end to a federated database configuration—and the problems that arose were corrected as testing proceeded.

This configuration demonstrated proof-of-concept of a distributed version of Greenstone using the DB2 distributed database. However, further work is required to make it run smoothly in a production environment.

Evaluation. The new module was benchmarked against the MG, MGPP and Lucene indexers on large test collections. Three kinds of documents were used: regular electronic documents from the well-known Reuters Corpus, machine-generated Lorem Ipsum, and newspaper text. The first two were short documents of about 200–250 words, and exhibited a rapidly decreasing incidence of new terms typical of standard English. The third type of document contained about 5 pages of newspaper text, or 15,000–20,000 words. Being produced by an imperfect OCR process, the number of new terms continues to grow rapidly no matter how many documents are considered.

Table 1 shows information about building indexers with the four tools. The first column gives the number of documents (the newspaper documents contained an

	Documents	Indexable text (MB)	New terms	Build time (hh:mm:ss)				Processor load (%)			
				MG	MGPP	Lucene	DB2	MG	MGPP	Lucene	DB2
Reuters	1,000	0.4	13.6	00:10	00:13	00:10	00:43	73	88	109	13
	10,000	3.7	5.7	01:10	01:29	00:51	02:15	99	113	130	32
	20,000	7.2	4.5	02:17	02:51	01:35	03:53	102	115	130	48
Lorem	50,000	84.2	2.6	06:02	15:29	06:59	08:58	112	110	103	55
	100,000	154.8	2.5	12:14	32:42	19:02	47:01	111	109	77	22
	500,000	773.4	2.2	2:02:30	3:51:56	1:38:38	–	60	86	78	–
	1,000,000	1510.0	2.0	4:03:59	7:02:58	2:22:59	–	26	78	67	–
Newspapers	500	16.0	1132	02:38	07:55	01:44	04:46	98	103	113	77
	1,000	31.5	896	05:19	15:10	03:25	09:40	99	104	114	77
	1,500	44.6	708	07:04	20:24	04:38	13:21	99	105	118	76
	5,000	247.0	1369	38:10	2:21:43	28:27	1:41:09	95	98	100	53
	15,000	745.0	1110	1:54:54	7:36:57	1:27:57	5:07:29	95	94	99	53

Table 1. Using DB2 to build collections of various sizes

average of 5 pages), and the second the amount of indexable text. The third shows the average number of terms per document that were new—i.e. did not appear in the text corresponding to the previous row of the table. As noted above, an extraordinary rate of growth is apparent for newspaper text.

The next four columns show the time taken by the four indexers to build full-text indexes. Unfortunately the DB2 code was not yet stable enough to reliably handle the largest of these collections, which accounts for the missing figures. The three existing indexers typically outperform DB2. However, the final columns show processor load (we used a dual-core machine, which is why some percentages exceed 100%), and indicate that DB2 under-utilizes the processor—probably due to some database or file IO bottleneck. We find these results encouraging: DB2 is generally comparable with the other indexers despite this evident problem.

Informal experiments suggest that the run-time performance of DB2 remains unchanged as the size of the underlying collection grows. However, the DB2 version generally took longer to generate a page of results than its three peers, suggesting that optimization is once again required.

7 Summary and Conclusions

In this article we have described how the open source Greenstone digital library software has been utilized to support two substantial national newspaper projects: one in New Zealand and the other in Singapore. Each currently contain over half a million OCR'd pages and are on track to be scaled into the multi-million range.

Full-text indexing is included in both. Despite OCR software advertising accuracy rates of "99.9%" this is typically quoted at the character level—the error rate at the

word level is much higher—and assumes high quality images. The images in our collections are on average over a century years old and restored from microfilm and microfiche, consequently the image quality is significantly poorer, and compounds the problem of error rates. Working with OCR text we have found that the vocabulary size essentially grows linearly with collection size, giving it the potential to become a stress point in the digital library system, however for the current (and projected) scale of operation the indexing software coped admirably.

To operate at this scale, Greenstone software developments were actually modest. First, the existing multi-indexer framework it provides was fine-tuned in its support for Lucene, and second, the standard flat-file database used (GDBM) was extended to support multiple instances to surpass a 2GB per-file limit. Testing showed this arrangement more than adequately satisfied the needs of these newspaper projects.

With an eye for future development, we then took things one step further, and analyzed Greenstone using DB2, an alternative database solution that in principle is better geared to scale well in a distributed fashion. While initial results in terms of performance were disappointing in comparison to the status quo, the work was exploratory (precisely to find out how well it performed "out of the box") and further work is schedule to look at ways to enhance performance for our given application. Not withstanding the DB2 outcome, in doing this work, Greenstone now supports a multi-database framework that is complementary to its indexer framework permitting the addition of other database systems such as Oracle and Mysql.

References

1. Witten, I.H. and Bainbridge, D. (2003) *How to build a digital library*. Morgan Kaufmann.
2. http://www.greenstone.org
3. http://www.greenstone.org/map
4. The Google n-gram collection is available on six DVDs from http://www.ldc.upenn.edu/
5. Witten, I.H., Moffat, A. and Bell, T. (1999) *Managing gigabytes: compressing and indexing documents and images (second edition)*. Morgan Kaufmann.
6. Littman, J.A (2006) "Technical Approach and Distributed Model for Validation of Digital Objects." In D-Lib Magazine, 12(5).
7. Reynaert, M. (2008) "Non-interactive OCR post-correction for Giga-scale digitization projects." In *Computational Linguistics and Intelligent Text Processing*, LNCS Vol. 4919, Springer.
8. http://www.greenstone.org/greenstone3-home
9. http://www-306.ibm.com/software/data/db2/
10. Ahuja, R. (2006) "DB2 9 unveiled: overview and new enhancements." IBM White Paper.
11. http://searchoracle.techtarget.com/sDefinition/0,,sid41_gci214145,00.html
12. Josifovski, V., Schwarz, P., Haas, L. and lin, E. (2002) "Garlic: a new flavor of federated query processing for DB2." *Proc ACM SIGMOD Int Conf on Management of Data*, Madison, Wisconsin; pp. 524-532.
13. http://www-306.ibm.com/software/data/db2/express/
14. http://www-306.ibm.com/software/data/db2/extenders/netsearch/
15. http://search.cpan.org/~mferris/Class-DBI-DB2-0.16/
16. http://dbconnect.sourceforge.net/