CORE
Metadata, citation and similar papers at core.ac.uk
Provided by UCT Computer Science Research Document Archive

Flexible Design for Simple Digital Library Tools and Services

Lighton Phiri
Department of Computer Science
University of Cape Town
Private Bag X3
Rondebosch 7701
Cape Town, South Africa
lphiri@cs.uct.ac.za

Hussein Suleman
Department of Computer Science
University of Cape Town
Private Bag X3
Rondebosch 7701
Cape Town, South Africa
hussein@cs.uct.ac.za

ABSTRACT

The design of Digital Library Systems (DLSes) has evolved over time, both in sophistication and complexity, to complement the complex nature and sheer size of digital content being curated. However, there is also a growing demand from content curators, with relatively small-size collections, for simpler and more manageable tools and services for managing content. The reasons for this particular need are driven by the assumption that simplicity and manageability might ultimately translate to lower costs of maintenance of such systems. This paper builds on previous work in order to assess the flexible nature of the proposed design approach —the explicit adoption of a minimalistic approach to the overall design of DLSes. A two-axis evaluation strategy was used to assess this proposed solution: a developer-oriented survey assessed the flexibility and simplicity; and a series of performance benchmarks were conducted to assess the scalability. In general, the study outlined some possible implications of simplifying DLS design; specifically the results from the developer-oriented user study indicate that simplicity in the design of the DLS repository sub-layer does not severely impact the interaction between the service sub-layer and the repository sub-layer. Furthermore, the scalability experiments indicate that desirable performance results for small- and medium-sized collections are attainable.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*Collection, Systems issues, User issues*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation*

General Terms

Human Factors, Measurement, Performance

Keywords

Digital Libraries, DSpace, Minimalism, Simplicity

1. INTRODUCTION

Preservation of digital content is a key concern of archivists because of the lack of strong evidence that content will, in fact, be accessible and usable in the near and distant future. Various solutions have been proposed but many of the comprehensive and generalisable content solutions (e.g., LOCKSS[1], DuraCloud[2], emulation) have a requirement for complex hardware/software/communication systems. At the other end of the spectrum are solutions based on simplicity, exemplified by the successful Project Gutenberg [1], where all manuscripts in its collection can be viewed in a universal plain text form.

Simplicity of system architectures also has been hailed as a potential solution to growing software service/system complexity. Many existing digital library (DL) tools are monolithic but there is a gradual shift towards a more Unix-like architecture, with interchangeable tools and common interfaces.

Thus, a simplification in DLS service/storage architecture may yield multiple benefits. However there may be adverse effects, such as reduced performance because of insufficient pre-processing and indexing.

This paper therefore reports on an investigation into the simplicity and performance constraints of simpler DLS architectures, specifically for a file-based storage architecture [2]. The core questions asked include: do simple architectures result in tools and services that are flexible and easy to use; is the performance acceptable in some circumstances; what are the bounds for acceptable use of file-based architectures; and what are the major issues that cause performance degradation, if any.

The remainder of this paper is structured as follows: Section 2 is a discussion of background information and related work associated with the study; Section 3 details the architecture of the prototype simple file-based repository; Section 4 describes a developer-oriented user study that was conducted to assess the flexibility and ease of use of the proposed architectural design, and additionally, the performance benchmarks that were conducted to assess the scalability of the architecture; and, finally, Section 5 concludes the paper.

2. RELATED WORK

Evaluation of DLs has been a subject of interest for DL research from the very early stages. This is evidenced by early initiatives such as the D-Lib Working Group on DL

[1] http://www.lockss.org
[2] http://www.duracloud.org

Metrics[3] that was established in the late 1990s. A series of related studies have since been conducted with the aim of outlining a systematic and viable way of evaluating the complex, multi-faceted nature of DLs that encompasses content, system and user-oriented aspects. For instance, the DELOS[4] Cluster on Evaluation [3, 4], which is perhaps the most current and comprehensive DL evaluation initiative, was initiated with the aim of addressing the different aspects of DL evaluation.

The DELOS DL evaluation activities have yielded some significant results. In an attempt to understand the broad view of DLs, Fuhr et al. [5] developed a classification and evaluation scheme using four dimensions: data/collection, system/technology, users and usage, and further produced a MetaLibrary comprising of test-beds to be used in DL evaluation. In a follow up paper, Fuhr et al. [6] proposed a new framework for evaluation of DLs with detailed guidelines for the evaluation process.

A number of user studies have been conducted to evaluate DLs from users' perspectives. For instance, Blandford et al [7] proposed PRET A Rapporter —a framework for structuring user-centred evaluation studies, in order to focus more on the design of interfaces, and user-system interaction. In addition, Xie [8] conducted a user study to investigate users' use, their criteria and their evaluation of DLs.

There have also been a number of specific performance evaluation experiments conducted on DLSes. In an attempt to study the ingest behaviour of a large-scale archive, Misra et al. [9] conducted ingest performance experiments to confirm the capability of DSpace[5] to serve as a large archive. Bainbridge et al. [10] provided a comprehensive report of stress-tests and scalability experiments conducted on three widely-used DL open source DLSes — DSpace, Fedora Commons[6] and Greenstone[7]— and further present a case study, detailing the construction of a large collection with 1.1 million digitised objects that was built using Greenstone. However, their experiments were more centred on the collection building process, which typically involves ingestion and importation of content. The scalability of Fedora Commons is also assessed on the Fedora Performance and Scalability Wiki[8], which gathers data and documents limits and constraints to help improve Fedora Commons.

While the performance-centred studies each address single aspects of DL performance, none of them provide all-encompassing results detailing the impact on typical DL operations as collections are scaled up in size. Furthermore, the majority of user studies that have been conducted predominantly focus on end users' perspectives of DLs. In Section 4 we present results from a developer study, and experimental results detailing collection sizes when response times exceed generally acceptable limits. First, however, the architecture of the experimental repository is presented.

3. REPOSITORY ARCHITECTURE

DLs are specifically designed to store, manage and preserve digital objects over long periods of times [11]. The digital objects are composed of bitstreams —the stored raw files e.g. documents and images, and corresponding metadata that provides a detailed description of the bitstreams. The high-level design of a typical DLS is composed of three main components/layers: a user interface layer through which end users access digital objects; a service layer that provides the necessary services for interacting with and manipulating the digital objects; and a repository layer for storage and management of digital objects [12, 13].

The architectural design of the prototype simple repository is based on a set of design principles presented in previous work [2], and makes use of a typical native operating system filesystem as the core storage infrastructure, with the goal of supporting preservation and enabling ease of use and management. The main components that make up the repository sub-layer, with all the components residing on the filesystem, arranged and organised as regular files and/or directories, are shown in Figure 1. A typical DLS repository is composed of two types of conceptual digital objects: Container Objects and Content Objects. Either may have an associated Metadata Object. Container Objects are repeatable and can be recursively created within other Container Objects. A Metadata Object associated with a Container Object contains: information that uniquely identifies the object; an optional description of the object, including relationships that might exist with other objects; and a detailed manifest of objects contained within it. Content Objects represent digital objects to be stored within the repository. The Metadata Objects associated with Content Objects are similar to those of Container Objects, with the exception of manifest information. This hierarchical structure of Containers and Content/Metadata maps directly onto directories and files, respectively, in a typical filesystem, as illustrated in Figure 1.

4. EVALUATION

In previous work conducted, the effectiveness of the proposed simpler design approach was assessed through the implementation of a real-world digital collection [14]. In order to extensively evaluate the proposed solution, a two-axis evaluation strategy was employed to assess the simplicity and easy of use, and the scalability.

4.1 Developer Survey

A developer-oriented user study was conducted to assess the simplicity of file-based repository implementations and the easy of interaction with such implementations. The study involved participants building layered services on top of a file-based repository, and subsequently participating in an optional post-study survey.

4.1.1 Target Group

The survey participants were recruited from a total of 34 Computer Science Honours (CSC4000) students, enrolled for the World Wide Web Technologies (WWW) elective course module at the University of Cape Town. The WWW module had a mandatory practical assignment, accounting for 20% of the overall assessment, in which the students were required to build generic Web applications, in groups, using the file-based repository store described in Section 3.

Incidentally, the student-centric target population was explicitly used due to the following reasons:

- The survey participants had the necessary background knowledge of Web technology principles

[3] http://www.dlib.org/metrics/public/index.html
[4] http://www.delos.info
[5] http://www.dspace.org
[6] http://fedora-commons.org
[7] http://www.greenstone.org
[8] http://fedora.fiz-karlsruhe.de/docs

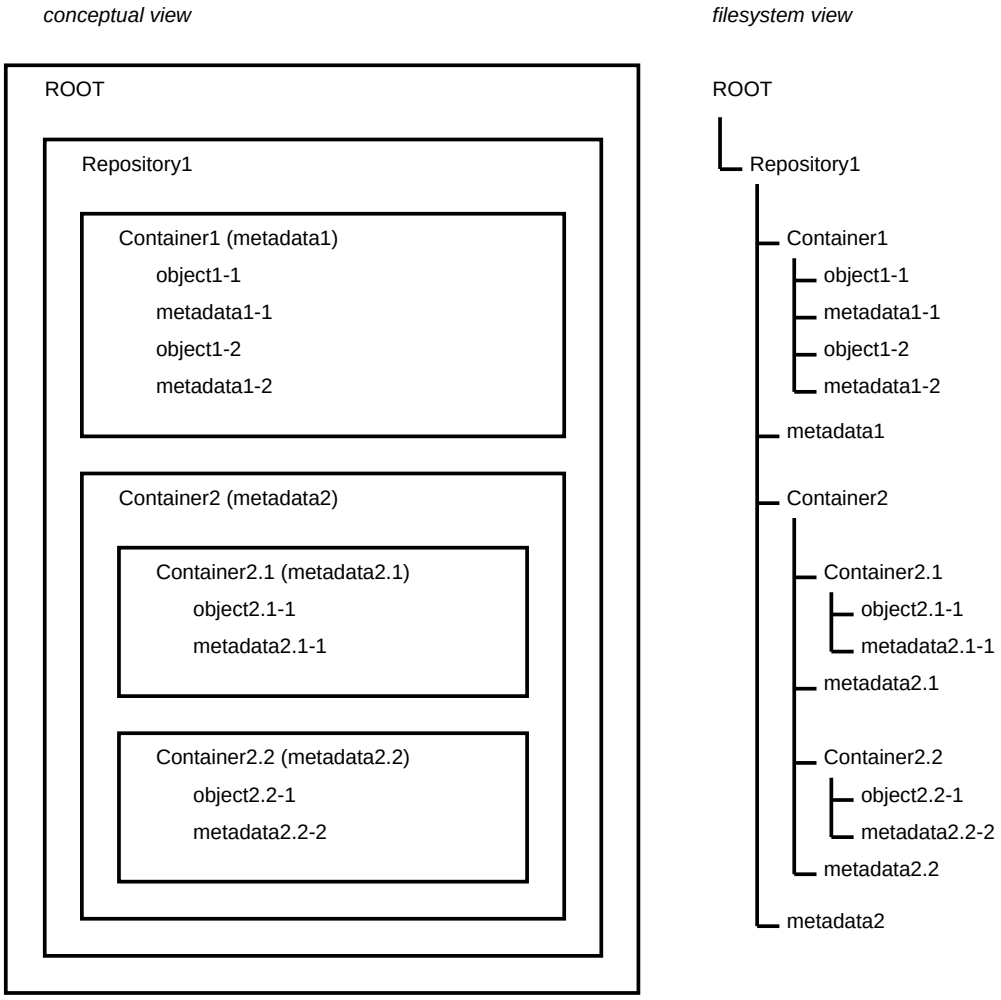


Figure 1: Repository architecture object structure.

- The survey participants had all been exposed to the same training
- The survey participants had a very strong motivate for taking part in the study, and thus spent a substantial amount of time engaging with the technology
- The cost implications of paying for professional developer, to participate in the survey, would have been substantial

A request for survey participation was emailed to the class mailing list after the assignment due date, in which 26 out of the 34 students responded, as shown in Table 1.

4.1.2 *Data Collection*

A post-experiment survey was conducted in the form of an online questionnaire, designed using LimeSurvey[9]. The questionnaire was aimed at eliciting participants' experience in working with a simple and minimalistic file-based collection, effectively establishing the impact of simplicity on the overall design.

4.1.3 *Results and Discussion*

The survey participants all met the basic requirement associated with the technical expertise required to undertake the study —their knowledge of Web technology principles are depicted in Figures 2; their knowledge of fundamental DL principles are depicted in 3; and their knowledge working with some popular storage solutions are depicted in 4. The implementation of the Web services was done using a variety of programming languages, as shown in Figure 5. In addition, the participants' views on the simplicity and ease of use of the repository is shown in Figure 6. Furthermore, when asked to rank preferred storage solutions for data management, 46% of the participants chose database management systems in preference to using file-base and cloud based solutions.

The survey results indicate that the target population generally had the necessary skill-set required for this study. The majority of respondents had some form of experience working with Web applications and associated technologies (see Figure 2). Furthermore, all respondents were familiar with fundamental concepts associated with DLs (see Figure 3).

The strong preference of using databases as storage structures is arguably as a result of the majority of participants' prior work with databases, as shown in Figure 4, and is further justified by the question that asked participants for reasons for their prior preferred storage solutions; the responses from some participants who ranked databases

[9] http://www.limesurvey.org

Table 1: Developer survey target population

Groups	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8	Group 9	Group 10	Group 11	Group 12	Grand Total
Candidates	3	3	3	3	3	2	3	3	3	3	3	2	34
Participants	3	3	1	2	2	2	3	3	2	2	1	2	26
Recruitment	100%	100%	33.3%	66.6%	66.6%	100%	100%	100%	66.6%	66.6%	33.3%	100%	76.5%

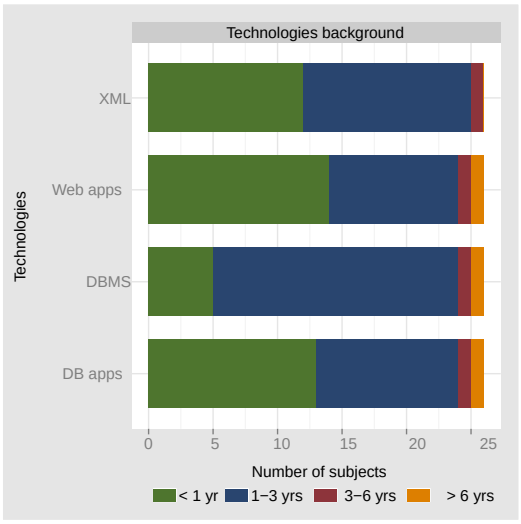


Figure 2: Survey participants' background knowledge working with technologies relevant to the study.

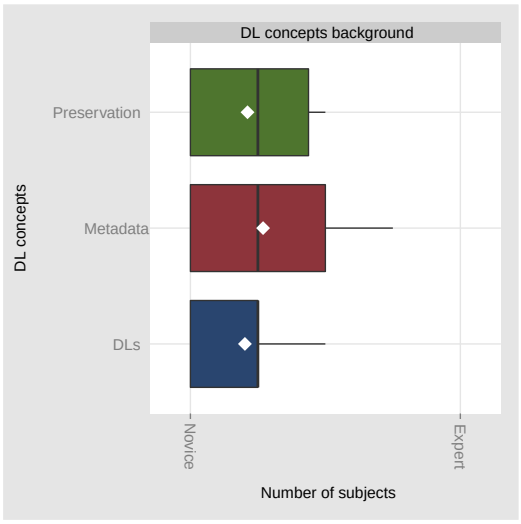


Figure 3: Survey participants' background knowledge working with fundamental DL concepts.

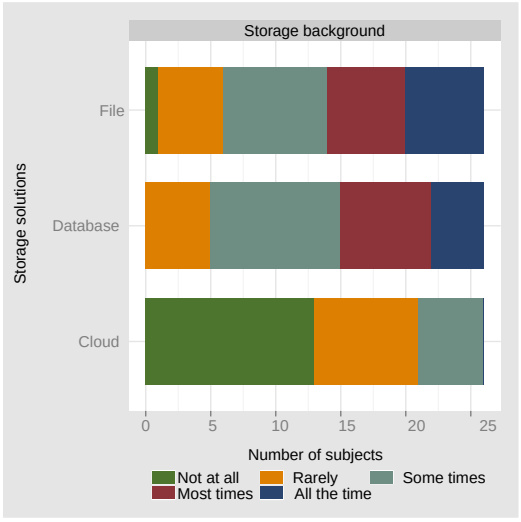


Figure 4: Survey participants' background knowledge working with some popular storage solutions.

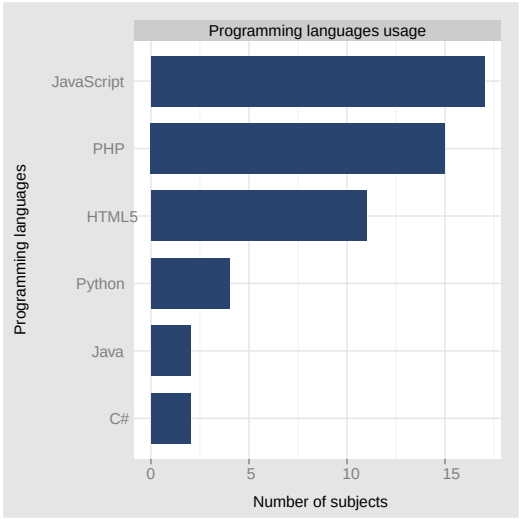


Figure 5: Survey participants' programming languages usage during service implementation.

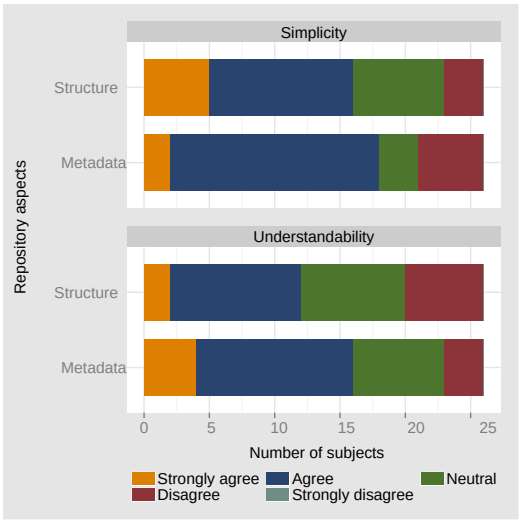


Figure 6: Survey participants' simplicity and understandability ratings of repository design.

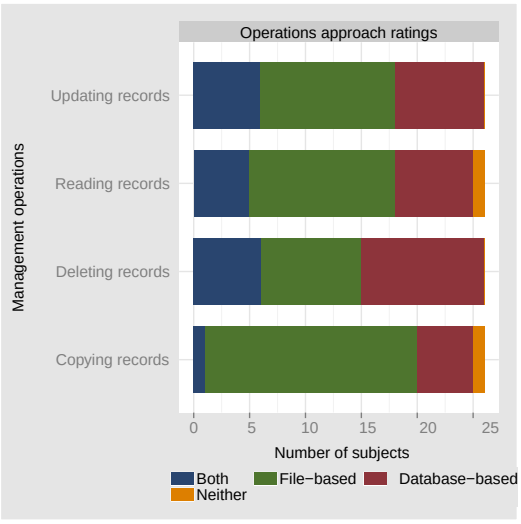


Figure 7: Survey participants' ratings of data management approaches for typical DL operations.

first are listed below.

- "I understand databases better."
- "Simple to set up and sheer control"
- "Easy setup and connection to MySQL database"
- "Speed of accessing data, and its free."
- "Ease of data manipulation and relations"
- "Easy to query"
- "Centralised management, ease of design, availability of support/literature"
- "The existing infrastructure for storing and retrieving data"
- "Querying a database table to retrieve a record is most useful for data."

In addition, as shown in Figure 6, 69% and 61% of the participants found the storage of metadata records in XML-encoded files simple and easy to use. Interestingly, 62% found the structure simple to work with, however, only 46% found the structure of records understandable. This significant reduction is attributed to the fact that the majority of participants had limited knowledge of the method used to model relationships between records; incidentally, Dublin Core 'hasPart' and 'isPartOf' [15] were used to model relationships between records. Interestingly, out of the total 12 participants whose preference was databases,the majority identified themselves as having little background information pertaining to metadata standards, DLs and digital preservation. It can be argued that their lack of knowledge of these fundamental concepts could have influenced their subjective views —this is supported by some of their general comments listed below.

- "Had some difficulty working the metadata, despite looking at how to process DC metadata online, it slowed us down considerably."
- "Good structure although confusing that each page has no metadata of its own(only the story)."
- "The hierarchy was not intuitive therefore took a while to understand however having crossed that hurdle was fairly easy to process."
- "I guess it was OK but took some getting used to"

Furthermore, the range of Web services implemented by the target population and the variety of programming languages, as shown in Figure 5, used to implement the services is indicative of the flexibility of the repository design, strongly indicating that the repository design did not significantly influence the choice of service and implementation language. This conclusion is further supported by an explicit survey question, that was aimed at eliciting respondents' views on whether the repository structure had a direct influence on their programming language(s) of choice, to which only 15% of the participants agreed.

In conclusion, the results from the developer survey suggest that developer interaction with resulting systems is not significantly affected, as the majority of participants found the collection simple and easy to work with. The use of a variety of programming languages also gives insight into the flexibility of the proposed design —a key attribute of extensibility in software design.

4.2 Scalability

A series of experiments (see Section 4.2.3) were performed to assess the scalability of specific DL services with different collection sizes/structures.

In order to compare scalability performance with existing solutions, the same experiment workloads described in Section 4.2.2 were used to conduct scalability experiments on DSpace (see Section 4.2.5).

4.2.1 Test Setup

The experiments were all conducted on a standalone Intel Pentium (E5200@ 2.50 GHz) with 4 GB of RAM running Ubuntu 12.04.1 LTS. ApacheBench 2.3[10] and Siege 2.70[11] were used to simulate user requests, with five-run averages taken for each request.

[10] http://httpd.apache.org/docs/2.2/programs/ab.html

[11] http://www.joedog.org/siege-home

Table 2: Performance experiment dataset workload design

Workload	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Records	100	200	400	800	1600	3200	6400	12800	25600	51200	102400	204800	409600	819200	1638400
Collections	19	25	42	57	67	83	100	112	116	119	127	129	128	131	131
Size (MB)	0.54	1.00	2.00	3.90	7.60	15.00	30.00	60.00	118.00	236.00	471.00	942.00	1945.00	3788.80	7680.00

The dataset used for the experiments is a collection of Dublin Core [15] XML records, organised into 131 subsets, which was harvested from the NDLTD Union Catalogue[12] using the OAI-PMH 2.0 protocol [16]. The harvested records were separated into individual XML files that were used to design the experiment workloads described in Section 3.

4.2.2 Workload Design

A random sampling technique was used to generate 15 workloads of exponentially-increasing sizes, with sampled records placed into their original containers. The structure of the dataset is depicted in Table 2.

A second dataset was then created, with a further level of containers based on record publication dates, resulting in a 2-level hierarchy. Finally, a third dataset was created, with one additional level of containers based on the initial letter of an author's last name, resulting in a 3-level hierarchy. Figure 8 illustrates the object organisation in one-level, two-level and three-level workload models.

As stated in Section 3, to leverage ease of use and management of resulting systems, storage of digital objects is exclusively done on the native operating system. In order to further ensure simplicity, repository features and functionalities that could potentially contribute to the overall complexity of resulting systems were eliminated. As such, the collection workloads were not indexed, and no third-party search services were used.

4.2.3 Performance Benchmarks

The performance benchmarks were aimed at evaluating the performance and scalability of services with varying collection sizes. Using Nielsen's three important limits for response times [17], a series of experiments were designed, with each experiment specifically focusing on determining the break-even point at which performance drastically degrades.

The performance experiments were carried out on the following list of services derived from a transaction log analysis of a production digital library system[13]:

- Item ingestion
- Full-text search
- OAI-PMH data provider operations
- Feed generation of most recently added items

4.2.4 Results and Discussion

Item Ingestion.

The ingestion process for a typical DLS in part involves importation of metadata associated with the bitstreams being ingested. The purpose of experiments conducted for this aspect was to determine the relative ingestion performance of metadata records, in terms of response time, with varying workload sizes. A single newer record was then used to simulate single item ingestion, through a script that read the record to be ingested and wrote the contents of the record to each of the 15 workload collections in the three datasets. The times taken to successfully write the record to disk were then noted. Figure 9 shows the results of the experiment. The ingestion response times generally remain constant, irrespective of the workload size. This is because the only overhead incurred results from disk write IO. The workload size does not substantially affect the ingestion response times.

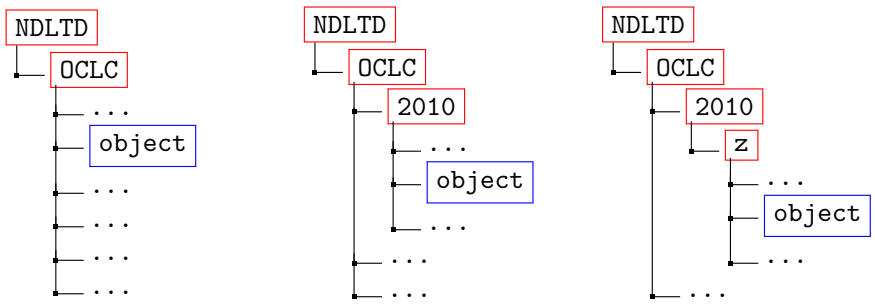
Search.

The most frequently occurring terms in the 15 workloads were identified using the Apache Solr Luke Request Handler[14] and search requests executed to determine response times for each workload. The search technique involved recursive traversal of workload containers, and successively parsing and querying each metadata file in the collection for the search phrase in question. The mean response times taken to generate search query resultsets are shown in Figure 10. There is a linear correlation between the workload size and the query response time, largely because all metadata records need to be analysed each time a search query is issued. In addition, a large amount of time is spent parsing and querying the record, with these tasks accounting for an average of 39% and 46% respectively. When the workload size exceeds 409600, the parsing phase becomes extremely expensive, accounting for 95% of the total search time.

OAI-PMH Data Provider.

The XMLFile Perl data provider module [18] was used in these experiments by integrating it with each of the workloads. The module was configured and deployed within a mod perl-enabled Apache 2.2.22 Web server. The resumptionToken size and container levels in datasets were varied. Figures 11 and 12 show: baseline results, for the four OAI-PMH verbs —GetRecord, ListIdentifiers, ListRecords and ListSets [19], conducted on the dataset with a one-level container and configured with a resumptionToken size of 1000; and results for the ListRecords verb when executed on the three datasets with varying container levels while keeping the resumptionToken size constant. The ListRecords and ListIdentifiers verbs are the most expen-

[12] http://union.ndltd.org/OAI-PMH
[13] http://pubs.cs.uct.ac.za
[14] http://wiki.apache.org/solr/LukeRequestHandler



(a) Dataset#1 structure (b) Dataset#2 structure (c) Dataset#3 structure

Figure 8: The workload hierarchical structures for the three experiment datasets. The setSpec, publication date and first character of creator name were used as first-, second and third-level container names respectively.

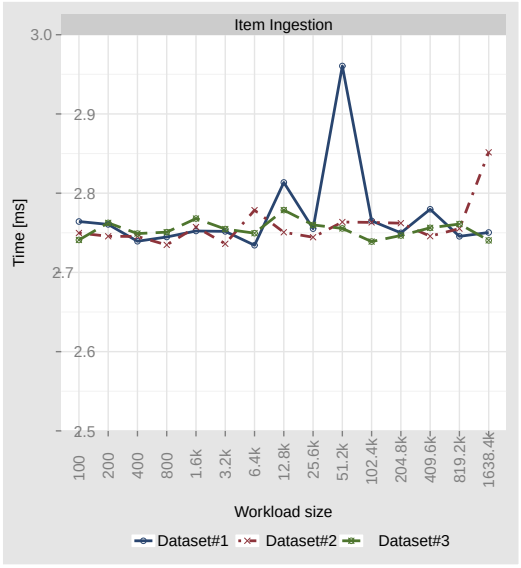


Figure 9: A plot showing the average time taken to ingest a single item into an existing collection.

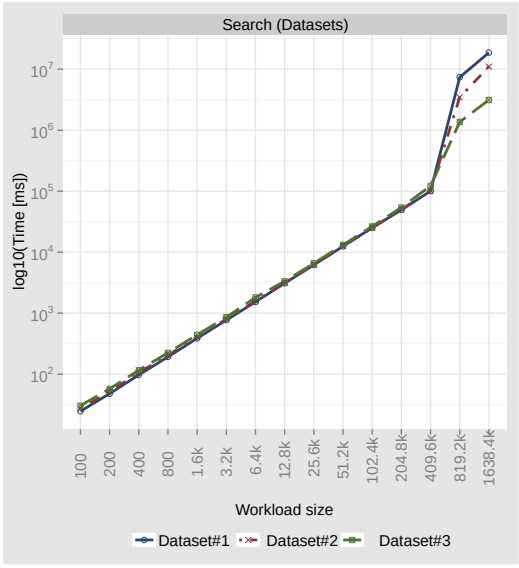


Figure 10: A plot showing the impact of collection size/structure on overall query performance.

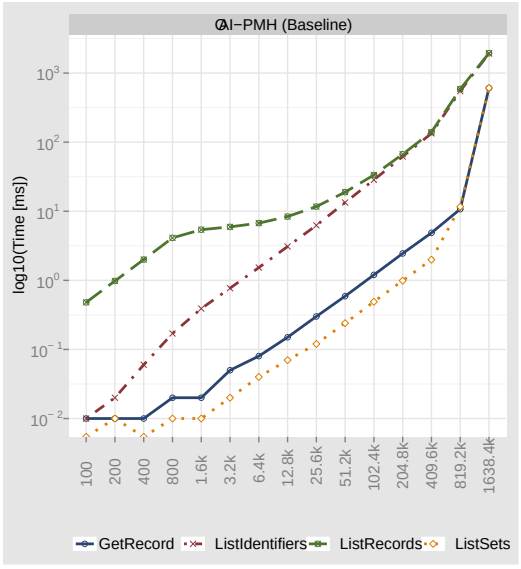


Figure 11: A plot showing the OAI-PMH data provider baseline performance benchmarks for all request verbs.

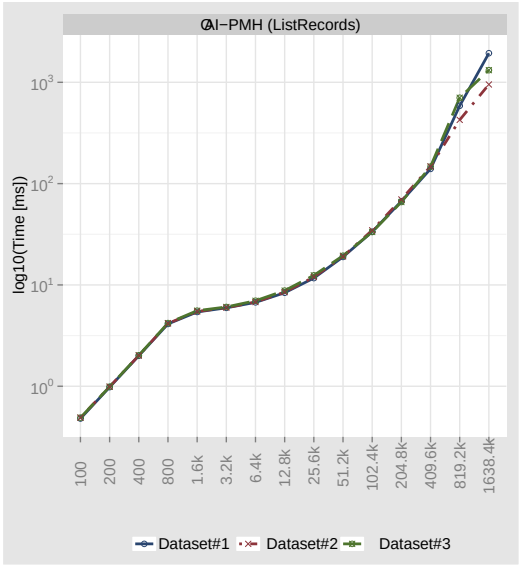


Figure 12: A plot showing impact of collection size/structure on OAI-PMH data provider ListRecords verb.

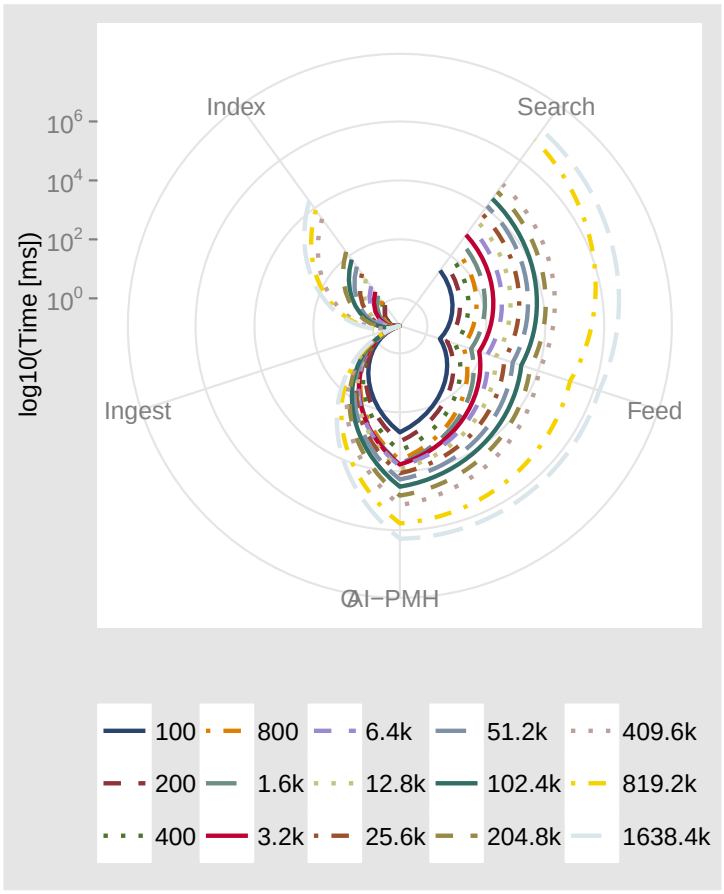


Figure 13: A Kiviat plot showing increases in response times relative to workload sizes for file-based stores for all evaluation aspects —full-text search, OAI-PMH data provider, RSS feed generator and single item ingestion, as well as batch indexing.

sive of the OAI-PMH verbs, each taking more than 10 seconds when the workload size goes beyond 25600 and 12800, respectively, for the baseline results.

Feed Generator.

The top N most recent records were identified using operating system creation and modification timestamps by traversing the 15 workloads to determine the response times.

The results indicate a substantial change in the response times for two-level and three-level structured workloads, relative to one-level structured workloads. This change is as a result of the increase in the traversal times as the hierarchies are increased.

Summary.

The performance experiment results strongly indicate that the performance is within generally acceptable limits for medium-sized collections, as evidenced in the Kiviat plot shown in Figure 13. Figure 13 also indicates that ingestion performance is significantly better than the other services.

In addition, the performance degradation for all the services occurs for collections with larger than 12800 objects.

Furthermore, the results indicate that performance degradation of information discovery services —full-text search, feed generation and OAI-PMH associated services is comparable and is largely as a result of parsing, a problem that can easily be remedied through the use of an index.

4.2.5 *Performance Comparisons*

This experiment was conducted to evaluate and compare performance results from non-indexed file-based repositories with an equivalent DSpace-based setup. A total of 15 DSpace 3.1[15] instances were set up, corresponding to the 15 experiment workloads described in Section 4.2.2. The following operations were then performed on the DSpace instances, and subsequently compared with the results of the performance benchmarks described in Section 4.2.3

- Single item ingestion
- Search query performance
- OAI-PMH data provider performance

Figure 15 shows that the average time taken to ingest a single item using the file-based approach is much more efficient in comparison to DSpace. In contrast, the DSpace ingest phase comprises of an item-level database write phase, a collection-level database write phase and an indexing phase.

As shown in Figures 14, 16 and 17, information discovery operations —search operations and OAI-PMH data provider operations— are orders of magnitude faster on DSpace in comparison to the file-based store. The response times on DSpace for these operations are faster as a result of a third-party indexing service (Apache Solr[16])

[15] https://wiki.duraspace.org/display/DSDOC3x
[16] http://lucene.apache.org/solr

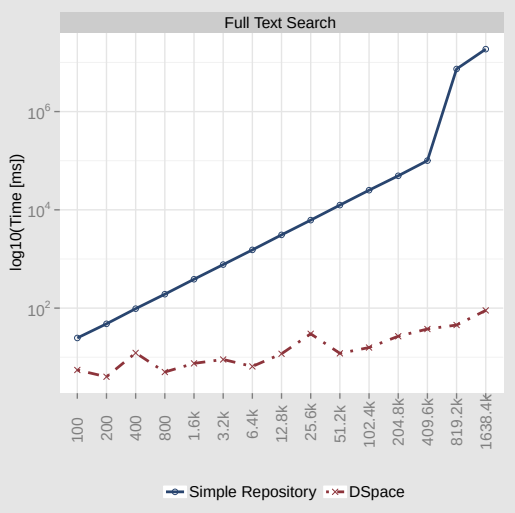


Figure 14: A plot showing the comparison of full-text search performance between the simple repository and DSpace.

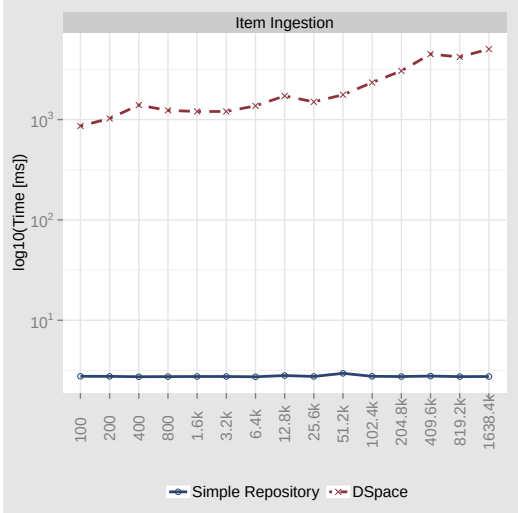


Figure 15: A plot showing the comparison of ingestion performance between the simple repository and DSpace.

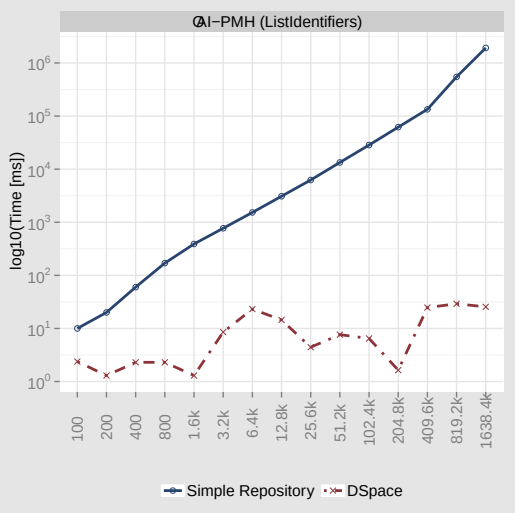


Figure 16: A plot showing the comparison of the OAI-PMH ListIdentifiers verb performance between the simple repository and DSpace.

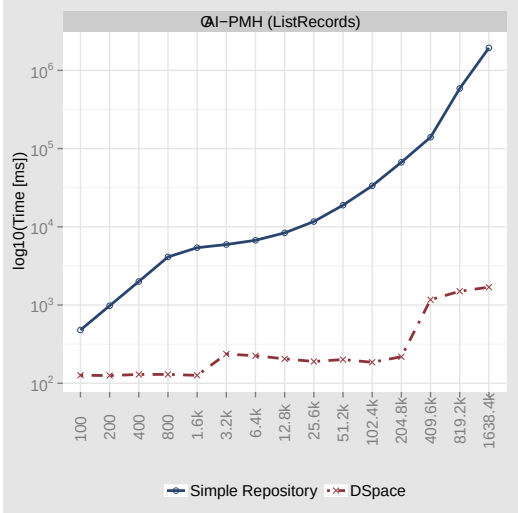


Figure 17: A plot showing the comparison of the OAI-PMH ListRecords verb performance between the simple repository and DSpace.

integrated with the application to facilitate fast search. Comparable speeds could be attained by the file-based store by integrating it with a similar indexing service.

5. CONCLUSION

The goal of this study was two-fold:

- Assess developer subjective views on using the proposed approach, in order to elicit the potential simplicity of resulting services
- Objectively assess the performance of a file-based store as workloads increase in size

The results from the developer survey suggest that the resulting simple file-based repository design is easy to work with and could potentially simplify repository management tasks. Furthermore, the results also indicate that a simple file-based repository design would have little impact on the extensibility of an application built on top of such a repository design.

The performance experiments show that performance is within generally-acceptable limits for small- and medium-sized collections with up to 25600 objects, on a given hardware platform. This is illustrated in the Kiviat plot shown in Figure 13. Performance degradation of operations such as information discovery and OAI-PMH services are largely as a result of extensive parsing, a problem that can easily be remedied through the use of an index. It was also shown that DSpace does indeed perform better at discovery-oriented services, but this is due to the use of an index for many operations.

It has been demonstrated that small- and medium-sized collections can easily use a file-based store as the core of an arguably more robust DL architecture, with no noticeable impact on performance. Larger collection sizes will work in a file-based repository without performance penalties for some services; other services require the use of appropriate (possibly minimal) indices to avoid full collection scans. It has been further demonstrated that the

approach is flexible and does not significantly impact the building of extensible services.

Ultimately, this study has explored the bounds of performance and shown that performance is manageable and need not be the reason not to use a simpler DLS architecture. When weighing architectural choices, the heavyweight system with brute-force indexed data stores must be reasonably balanced against the benefits of lightweight and simpler architectures.

6. ACKNOWLEDGEMENTS

This research was partially funded by the National Research Foundation of South Africa (Grant numbers: 85470 and 83998) and University of Cape Town. The authors acknowledge that opinions, findings and conclusions or recommendations expressed in this publication are that of the authors, and that the NRF accepts no liability whatsoever in this regard.

7. REFERENCES

[1]Michael Hart. *Project Gutenberg. The History and Philosophy of Project Gutenberg*. 1992.

[2]Lighton Phiri, Kyle Williams, Miles Robinson, Stuart Hammar, and Hussein Suleman. "Bonolo: A General Digital Library System for File-Based Collections". In: *Proceedings of the 14th International Conference on Asia-Pacific Digital Libraries*. Ed. by Hsin-Hsi Chen and Gobinda Chowdhury. Springer Berlin / Heidelberg, 2012, pp. 49–58.
DOI: 10.1007/978-3-642-34752-8_6.

[3]DELOS Working Group. *DELOS Workshop on the Evaluation of Digital Libraries*. 2004.

[4]Leonardo Candela, Donatella Castelli, Pasquale Pagano, Constantino Thanos, Yannis Ioannidis, Georgia Koutrika, Seamus Ross, Hans-Jörg Schek, and Heiko Schuldt. *The DELOS Digital Library Reference Model. Foundations for Digital Libraries*. ISTI-CNR at Gruppo ALI, 2008.

[5]Norbert Fuhr, Preben Hansen, Michael Mabe, Andras Micsik, and Ingeborg Sølvberg. "Digital Libraries: A Generic Classification and Evaluation Scheme". In: *Research and Advanced Technology for Digital Libraries*. Ed. by Panos Constantopoulos and Ingeborg T. Sølvberg. Springer Berlin Heidelberg, 2001, pp. 187–199.
DOI: 10.1007/3-540-44796-2_17.

[6]Norbert Fuhr, Giannis Tsakonas, Trond Aalberg, Maristella Agosti, Preben Hansen, Sarantos Kapidakis, Claus-Peter Klas, László Kovács, Monica Landoni, AndrÃąs Micsik, Christos Papatheodorou, Carol Peters, and Ingeborg Sølvberg. "Evaluation of digital libraries". In: *International Journal on Digital Libraries* 8.1 (2007), pp. 21–38.
DOI: 10.1007/s00799-007-0011-z.

[7]Ann Blandford, Anne Adams, Simon Attfield, George Buchanan, Jeremy Gow, Stephann Makri, Jon Rimmer, and Claire Warwick. "The PRET A Rapporter framework: Evaluating digital libraries from the perspective of information work". In: *Information Processing & Management* 44.1 (2008), pp. 4–21.
DOI: 10.1016/j.ipm.2007.01.021.

[8]Xie Iris Hong. "Users' evaluation of digital libraries (DLs): Their uses, their criteria, and their assessment". In: *Information Processing & Management* 44.3 (2008), pp. 1346–1373.
DOI: 10.1016/j.ipm.2007.10.003.

[9]Dharitri Misra, James Seamans, and George R. Thoma. "Testing the scalability of a DSpace-based archive". In: *Proceedings of the IS&T Archiving* 2008.1 (2008), pp. 36–40.

[10]David Bainbridge, IanH. Witten, Stefan Boddie, and John Thompson. "Stress-Testing General Purpose Digital Library Software". In: *Research and Advanced Technology for Digital Libraries*. Ed. by Maristella Agosti, JosÃl' Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas. Springer Berlin Heidelberg, 2009, pp. 203–214.
DOI: 10.1007/978-3-642-04346-8_21.

[11]William Y. Arms. *Digital Libraries*. Cambridge, Massachusetts: The MIT Press, 2001.

[12]William Y. Arms. "Key Concepts in the Architecture of the Digital Library". In: *D-Lib Magazine. The Magazine of Digital Library Research* 1.1 (1995).
DOI: 10.1045/july95-arms.

[13]William Y. Arms, Christophe Blanchi, and Edward A. Overly. "An Architecture for Information in Digital Libraries". In: *D-Lib Magazine. The Magazine of Digital Library Research* 3.2 (1997).
DOI: 10.1045/february97-arms.

[14]Lighton Phiri and Hussein Suleman. "In Search of Simplicity: Redesigning the Digital Bleek and Lloyd". In: *DESIDOC Journal of Library & Information Technology* 32.4 (2012), pp. 306–312.

[15]DCMI Usage Board. *Dublin Core Metadata Element Set, Version 1.1*. Dublin Core Metadata Initiative. 1999.

[16]Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner. *The Open Archives Initiative Protocol for Metadata Harvesting*. Dublin Core Metadata Initiative. 2002.

[17]Jakob Nielsen. *Response Times: The 3 Important Limits*. 1993.

[18]Hussein Suleman. *OAI-PMH2 XMLFile File-based Data Provider*. 2002.

[19]Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner. *Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting. Guidelines for Repository Implementers*. 2002.