Benchmarking a File-based Digital Library System Repository Architecture

Lighton Phiri and Hussein Suleman

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

Abstract. Digital Library Systems (DLSes) have over the past few decades evolved into complex tools and services used to manage Digital Libraries (DLs). However, as the amount digitised and born digital content being generated increases, there is increasingly a growing need for much simpler tools for the storage, management and long term preservation of data. The simplification in the design of DLS components has obvious an implication of, among other things, adversely affecting overall performance of resulting tools and services. This paper builds on previous work, which resulted in a prototype simple repository design, by outlines experimental results from a series of performance benchmarks that were conducted on to determine the extent to which such a simple repository architecture would scale to provide acceptable response times. The designed experiments were executed on a dataset with 1 638 400 objects, and involved execution of common DL operations on linearly increasing workloads designed based on the simple architecture. In addition, the experimental setup was replicated on a commonly used DL software –DSpace– to provide comparative metrics. The results indicate that collection sizes with at most 25 600 objects yield desirable response times. Furthermore, performance degradation typically manifests in information discovery operations.

Keywords: Benchmarks, DSpace, Evaluation, Performance, Scalability

1 Introduction

Data curation is increasingly becoming common place and there is a growing demand for simpler tools for storage of digital content. However, simplifying the design of such tools has implications that could potentially affect the performance of resulting system. In addition, there is a requirement from content curators for readily available information outlining, among other things, information related to the scalability of such tools. This paper builds upon previous work [10] by outlining a series of performance experiments conducted to determine how well simple architectures can be scaled up.

The remainder of this paper is structured as follows: Section 2 is a discussion of background information and related work. Section 3 details the architecture of

the file-based repository. Section 4 describes the performance benchmarks that were conducted to assess the scalability of the architecture. Finally, Section 5 concludes the paper.

2 Related work

Evaluation of DLs has been a subject of interest for DLs research from the very early stages. This is evidenced by early initiatives such as the D-Lib Working Group on DL Metrics[1] 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[2] Cluster on Evaluation [6,7], which is perhaps the most current and comprehensive DL evaluation initiative, was initiated with the aim of addressing the different aspects of DLs evaluation.

The DELOS DL evaluation activities have yielded some significant results; in an attempt to understand the broad view of DLs, Fuhr et al. [1] developed a classification and evaluation scheme using four major 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. [2] proposed a new framework for evaluation of DLs with detailed guidelines for the evaluation process.

There have 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. [4] conducted ingest performance experiments to confirm the capability of DSpace[3] to serve as a large archive. Nonetheless, their focus was on the performance of the ingest process. Bainbridge et al. [3] provided a comprehensive report of stress-tests and scalability experiments conducted on three widely used DL open source DLSes –DSpace, Fedora Commons[4] and Greenstone[5] 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 ingestions and importation of content. There are additional performance benchmarks that have been conducted; for instance the Fedora Performance and Scalability Wiki [5] gathers data and document limits and constraints to help improve Fedora Commons.

In as much as the mentioned studies are useful, non of them provide all-encompassing results detailing the impact on typical DL operations as collections are scaled up in size. In Section 4 we present experimental results detailing collection sizes when response times exceed generally acceptable limits.

[1] http://www.dlib.org/metrics/public/index.html
[2] http://www.delos.info
[3] http://www.dspace.org
[4] http://fedora-commons.org
[5] http://www.greenstone.org

3 Repository architecture

The architectural design of the prototype simple repository is based on a set of design principles presented in previous work conducted [10], and is centred around designing a simple repository, which at a bare minimum is capable of facilitating the core features of a typical DLS long term preservation, and ease of access and management of digital objects. The repository design is file-based and makes use of a typical native operating system filesystem as the core infrastructure.

The main components that make up the repository sub-layer, with all the components residing on the filesystem, arranged and organised as normal operating system files regular files and/or directories as shown in Figure 1. A typical DLS repository would be located in an application accessible base root directory node, and is composed of two types of digital objects -Container Objects and Content Objects- both of which are created and stored within the repository with companion Metadata Objects that store representational information associated with the object.

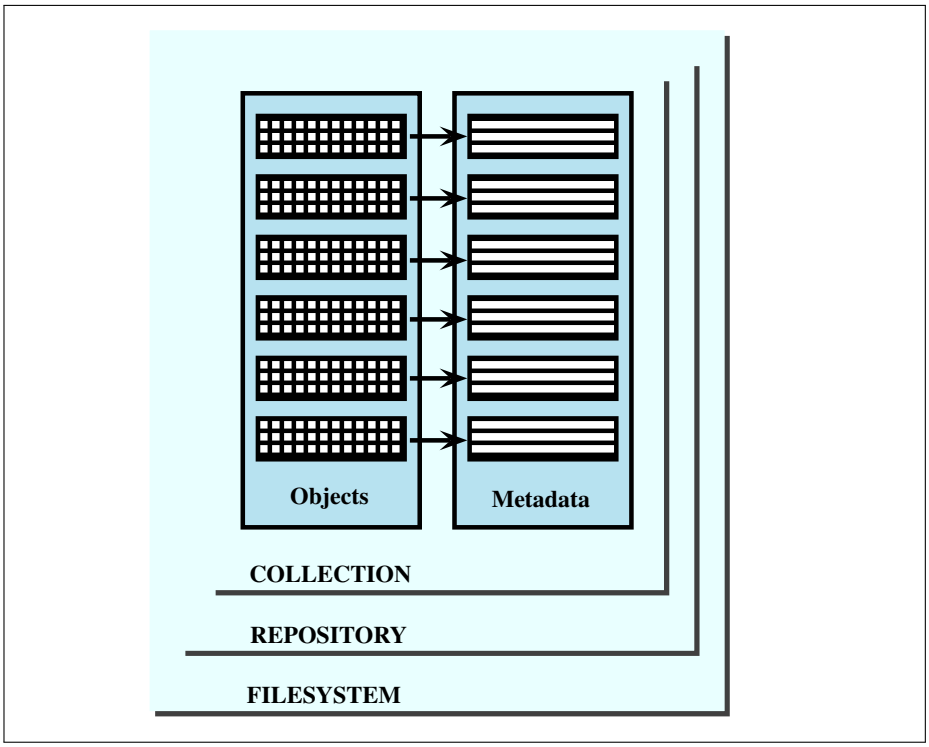

Fig. 1: Repository object structure

Container Objects can be recursively created within the root node as the repository scales, and exhibit an interesting characteristic of a enabling the creation of additional Container Objects within them. Metadata Objects associated with Container Objects holds information that uniquely identifies the object; optionally describe the object in more detail, including relationships that might exist with other objects within the repository; and a detailed log of objects contained within it the manifest. Content Objects represent digital objects typically bitstreams to be stored within the repository. The representational information stored in the Metadata Objects associated with Content Objects is similar to that of Container Objects, with the exception of manifest related information.

4 Experimental results

A significant architectural change performed to the design and implementation of the simple repository outlined in Section 3 involves changing the way metadata records are stored in the repository sub-layer of DLSes. More specifically, the proposed solution advocates for the use of a typical operating system filesystem for the storage of metadata records, as opposed to the conventional use of a database management system. This design decision is motivated by two key factors –simplicity and manageability. However, conventional wisdom and folklore all point to the fact that system performance would evidently be adversely affected for relatively large collections.

4.1 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[6] and Siege 2.70[7] were used to simulate a single user request, with five run-averages taken for each aspect request.

The dataset used for the experiments is a collection of XML records, encoded using simple Dublin Core, which were harvested from the NDLTD Union Catalog[8] using the OAI-PMH 2.0 protocol. The harvested records were separated into individual XML-encoded files that were used to design the experiment workloads described in Section 4.2.

4.2 Workload design

A random sampling technique was used to generate linearly increasing workloads, with records randomly selected from the 131 SetSpecs[9] constituting the harvested records, as shown in Table 1. An additional two datasets were then spawned to create a total of three experiment datasets with varying hierarchical

[6] http://httpd.apache.org/docs/2.2/programs/ab.html.
[7] http://www.joedog.org/siege-home
[8] `http://union.ndltd.org/OAI-PMH`
[9] Optional construct for grouping items in an OAI-PMH repository

structures –two- and three-level structures; the second level container was created using record publication dates, whilst the third level container was created using the initial letter of record author's last name.

Table 1: Experiment dataset workload design

	W1	**W2**	**W3**	**W4**	**W5**	**W6**	**W7**	**W8**	**W9**	**W10**	**W11**	**W12**	**W13**	**W14**	**W15**
Records	100	200	400	800	1600	3200	6400	12 800	25 600	51 200	102 400	204 800	409 600	819 200	1 638 400
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

4.3 Performance benchmarks

The performance benchmarks were aimed at evaluating the performance and scalability of varying collection sizes. Using Nielsen's three important limits for response times [11], a series of experiments were designed, with each experiment specifically focusing on determining the break-even point at which performance and scalability drastically degrades.

The performance experiments were carried out on the aspects listed below. The aspects were arrived at after conducting a transaction log analysis of a production digital library system[10] –a subject repository running EPrints 2.1.1[11].

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

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 was then noted.

Figure 2 shows the results of the experiment. The ingestion response times generally remain constant irrespective of the workload size. This is because the

[10] `http://pubs.cs.uct.ac.za`
[11] `http://www.eprints.org`

only overhead incurred results from disk write IO. The workload size does not significantly affect the ingestion response times.

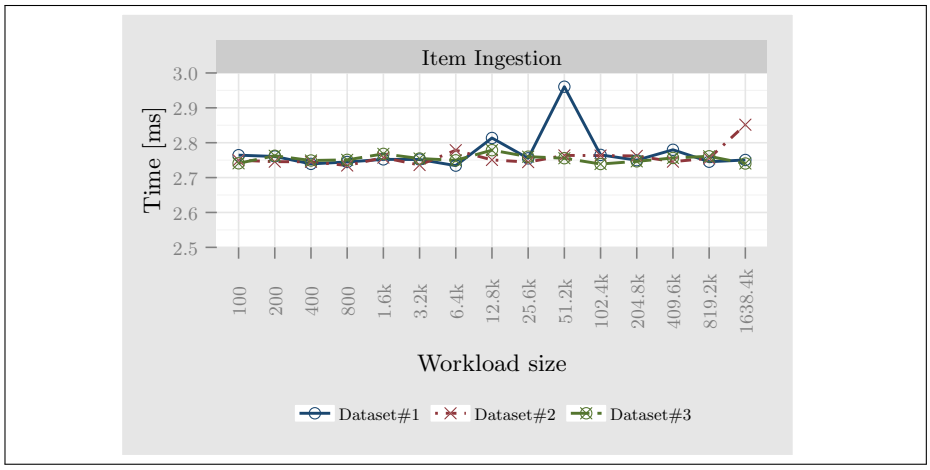

Fig. 2: Performance evaluation ingestion.

Search The most frequent occurring terms in the 15 workloads were identified using the Apache Solr LukeRequestHandler[12], 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 show in Figure 3. There is linear correlation between the workload size and the query response time. The results are further strengthened by the fact that all metadata records need to be analysed each time a search query is issued. In addition, a significant amount time is spent parsing and querying the record with each of the tasks accounting for an average of 39 % and 46 % respectively, before the workload size exceeds 409 600, at which point the parsing phase becomes extremely expensive –accounting for 95 % of the total search query time.

OAI-PMH data provider The XMLFile Perl data provider module [9] was used to conduct the experiments. The module was configured and deployed within a mod_perl enabled Apache 2.2.22 Web server. All potential experiment factors –resumptionToken size and container levels in datasets– were identified and varied.

Figure 4 shows baseline results, for the four OAI-PMH verbs, conducted on the dataset with one-level container and configured with a resumptionToken size

[12] `http://wiki.apache.org/solr/LukeRequestHandler`

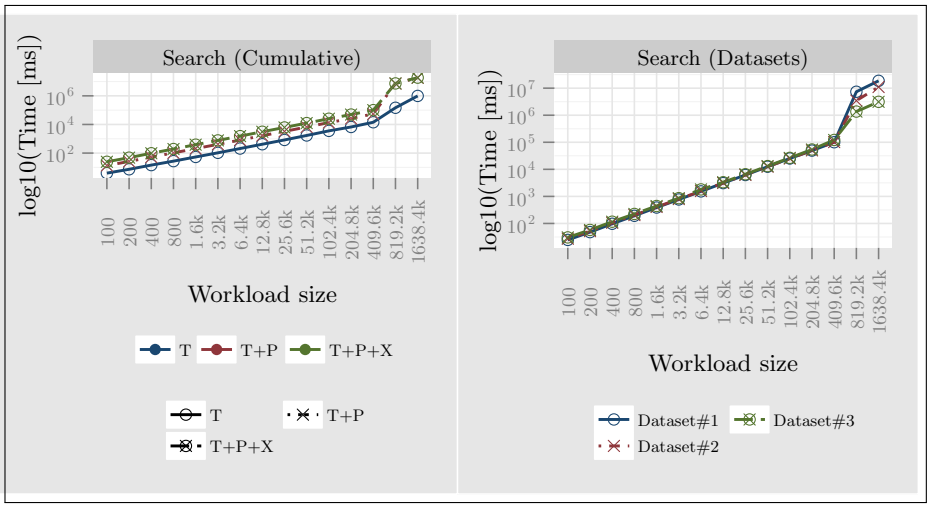

Fig. 3: Performance evaluation search aspect.

1000; and results for the ListRecords verb when executed on the three datasets with varying container levels whilst keeping the resumptionToken size constant. The ListRecords and ListIdentifiers verbs are the most expensive of the OAI-PMH verbs, each taking more than 10 seconds when the workload size goes beyond 25 600 and 12 800 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 significant 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.

4.4 Performance comparisons

This experiment was conducted to evaluate and compare performance results from potential non-indexed file-based repositories with an equivalent DSpace-based setup. A total of 15 DSpace 3.1[13] instances were set up corresponding to the 15 experiment workloads described in Section 4.2. The operations described in Section 4.3 were then performed on the DSpace instances.

Figure 5 shows that the average time taken to ingest a single item using the proposed approach is significantly much more efficient in comparison 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.

[13] `https://wiki.duraspace.org/display/DSDOC3x`

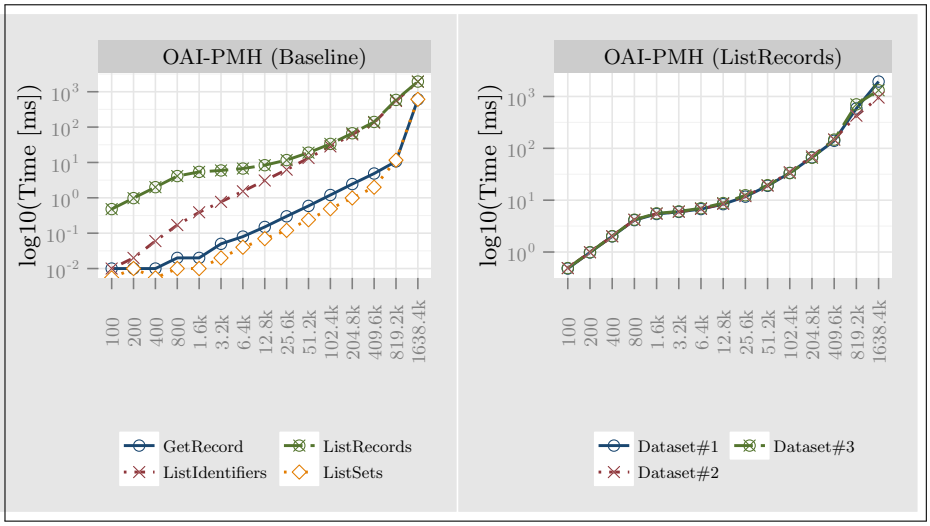

Fig. 4: Performance evaluation OAI-PMH aspects.

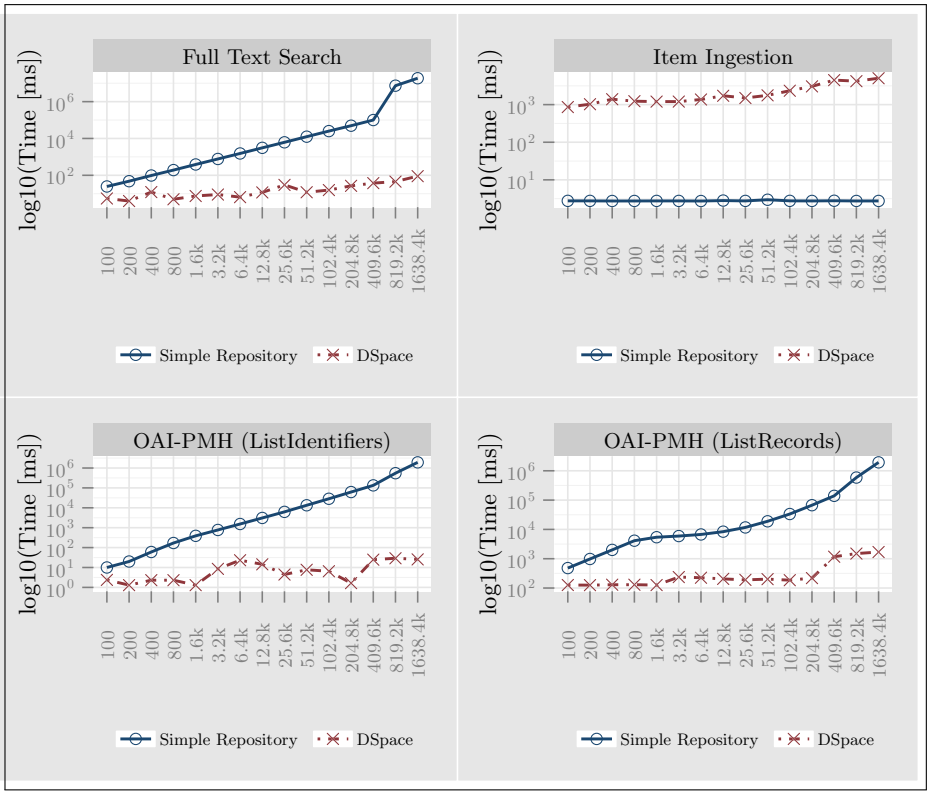

Fig. 5: Performance evaluation comparisions.

As shown in Figure 5, 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 significantly faster as a result of a third-party search service (Apache Solr[14]) integrated with the application to facilitate fast search. Incidentally, comparable speeds could be attained by integrating the file-based repository with a search service.

5 Conclusion

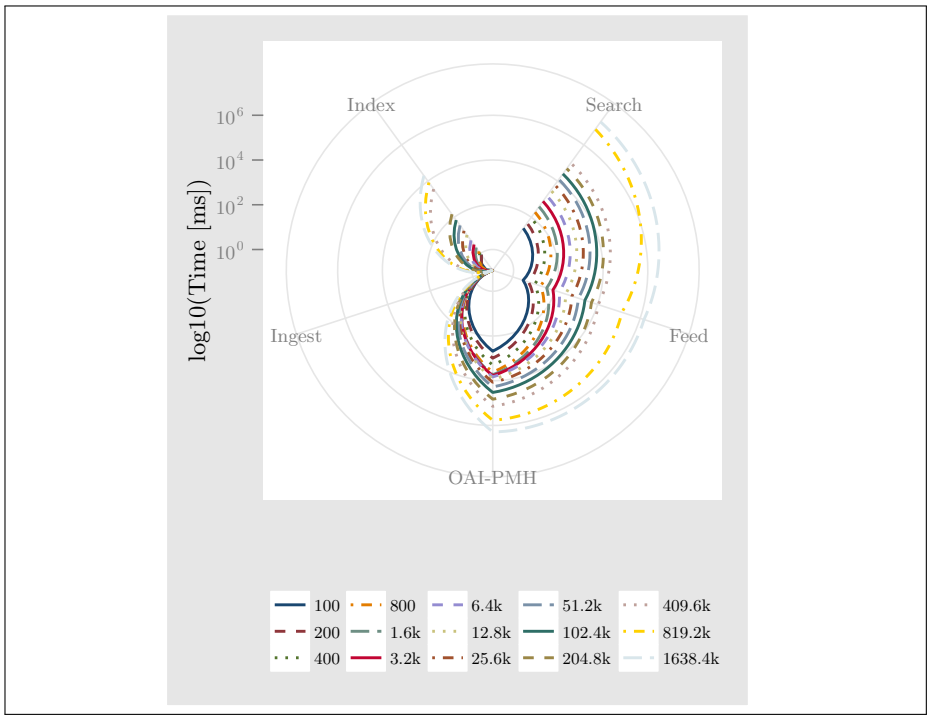

Fig. 6: A kiviat plot showing performance degradation (increase in response times) of evaluation aspects –batch indexing, full-text search, OAI-PMH data provider, RSS feed generator and single item ingestion– relative to increasing workload sizes, with each polar line representing the 15 experiment workloads.

The scalability performance experiments yielded results that strongly indicate that the performance would be within generally acceptable limits for medium-sized collections with at most 25 600 objects, as evidenced in the kiviat

[14] `http://lucene.apache.org/solr`

plot shown in Figure 6. It was further shown that performance degradation of operations such as information discovery and OAI-PMH associated services are largely as a result of parsing, a problem that can easily be remedied through the use of an index.

Finally, it was shown that the superior performance results from the comparative experiments done with DSpace are attributed to the external search service –Apache Solr and Lucene– integrated with DSpace to facilitate fast search. However, integration of such an external search service could easily be performed using the proposed approach.

References

1. Fuhr, N., Hansen, P., Mabe, M., Micsik, A., Sølvberg, I.: Digital Libraries: A Generic Classification and Evaluation Scheme. In: Constantopoulos, P., Sølvberg, I. (eds.) ECDL 2001. LNCS, vol 2163, pp. 187–199. Springer Berlin / Heidelberg (2001)
2. Fuhr, N., Tsakonas, G., Aalberg, T., Agosti, M., Hansen, P., Kapidakis, S., Klas, P., Kovács, L., Landoni, M., Micsik, A., Papatheodorou, C., Peters, C., Sølvberg, I.: Evaluation of digital libraries. International Journal on Digital Libraries (2007)
3. Bainbridge, D., Witten, I., Boddie, S., Thompson, J.: Stress-Testing General Purpose Digital Library Software. In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol 5714, pp. 203–214. Springer Berlin / Heidelberg (2009)
4. Misra, D., Seamans, J., Thoma, G R.: Testing the Scalability of a DSpace-based Archive. IS&T Archiving 2008. Bern, Switzerland. (2008)
5. Fedora Performance and Scalability Wiki. `http://fedora.fiz-karlsruhe.de/docs`
6. Fourth DELOS workshop. Evaluation of digital libraries: Testbeds, measurements, and metrics. Budapest: Hungarian Academy of Sciences (2002)
7. Revised Notes of the DELOS WP7 Workshop on the Evaluation of Digital Libraries. DELOS Workshop on the Evaluation of Digital Libraries. `http://dlib.ionio.gr/wp7/workshop2004.html`
8. Candela, L., Castelli, D., Pagano, P., Thanos, C., Ioannidis, Y., Koutrika, G., Ross, S., Schek, H., Schuldt, H.: The DELOS Digital Library Reference Model. Foundations for Digital Libraries. `http://eprints.port.ac.uk/4104`
9. Suleman, H.: OAI-PMH2 XMLFile File-based Data Provider `http://www.dlib.vt.edu/projects/OAI/software/xmlfile/xmlfile.html`
10. Phiri, L., Williams, K., Robinson, M., Hammar, S., Suleman, H.: Bonolo: A General Digital Library System for File-Based Collections. In: Hsin-Hsi, C., Gobinda, C. (eds.) ICADL 2012. LNCS, vol. 7634, pp. 49–58. Springer, Heidelberg (2012)
11. Nielsen, J.: Response Times: The 3 Important Limits. `http://www.nngroup.com/articles/response-times-3-important-limits`