Fine-grained Scalability of Digital Library Services in the Cloud

Lebeko Poulo
Department of Computer Science
University of Cape Town
Private Bag X3
Rondebosch 7701
Cape Town, South Africa
lpoulo@cs.uct.ac.za

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

Modern digital library systems are increasingly handling massive data volumes; this content needs to be stored, indexed and made easily accessible to end users. Cloud computing promises to address some of these needs through a set of services that arguably support scalability of service provision. This paper discusses a set of experiments to assess the scalability of typical digital library services that use cloud computing facilities for core processing and storage. Horizontal scalability experiments were performed to benchmark the overall performance of the architecture with increasing load. The results of the experiments indicate that stable response times and some degree of variability are attainable due to multiple middleware servers when browsing and/or searching a collection of a fixed size. There is minimal variation in response times when varying collection sizes and equally after the caching phases. Most importantly, request sequencing proved that the quantity and age of requests have no impact on response times. The experimental results thus provide evidence to support the feasibility of building and deploying cloud-based Digital Libraries.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation*

General Terms

Design, Experimentation, Performance

Keywords

Amazon AWS, Cloud Computing, Scalability

1. INTRODUCTION

The exponential increase in the amount of data currently being generated creates a need for robust, reliable and cost-effective computational and storage solutions, possessing the ability to easily scale with increasing capacity. While there exists various application domains where scalability is a critical requirement, Digital Library Systems (DLSes) are one such domain where scalability is a critical requirement. Common services, such as search engines, need to deal with large amounts of data and large numbers of user requests. These have traditionally been implemented on custom-designed compute clusters, but cloud computing presents the possibility of standard mechanisms for scalability.

Cloud computing (Armbrust et al., 2010) is increasingly becoming an attractive option for the deployment of information management systems, largely due to its elasticity with respect to economies of scale (Tak et al., 2011). It is this potential offering of cloud computing, as a utility service, that particularly makes it a potentially viable option for achieving the scalability requirements of modern DL infrastructure. The feasibility of migrating and hosting DL systems in the cloud is evident (Teregowda et al., 2010b; Teregowda et al., 2010a). However, most migrations to cloud environments have occurred at a high level and it is not known if DL services can be deeply integrated with cloud services to maximise the potential for cloud-service-level scalability.

This paper describes a set of prototype DL components, notably for search and browse operations—the two most common DL services. These components were designed to specifically make maximal use of cloud services for processing and storage. They are evaluated in terms of performance of speed and data store sizes, in order to assess the feasibility (or not) of integrating DL services with cloud services at a low level.

The remainder of this paper is organised as follows: Section 2 presents some relevant related work, specifically highlighting architectural design considerations and performance experiments of cloud-based applications; in Section 3, the proof-of-concept DLS implemented as part of this work is described, together with the design considerations used to arrive at the architecture of the system; Section 4 provides a detailed account of the scalability experiments that were conducted to assess the overall performance of the system; and, finally, Section 5 provides concluding remarks and po-

tential future work.

2. RELATED WORK

With the elasticity case having been proven on numerous occasions (Tak et al., 2011), a number of enterprise-level applications (Khajeh-Hosseini et al., 2010) with high-availability requirements and, now increasingly, scientific applications (Vecchiola et al., 2009) requiring massive computing resources are being migrated to the cloud. There has also been an increasing wave of application of cloud computing to Digital Libraries. Teregowda et al. (Teregowda et al., 2010a) explored the feasibility of deploying CiteSeer^{x}[1] into the cloud, with experimental results highlighting the feasibility of an equivalent cloud implementation. In a followup paper (Wu et al., 2014), Wu et al. outline their experiences migrating CiteSeer^{x} into a private cloud environment. More recently, Rosenthal and Vargas (Rosenthal and Vargas, 2013) carried out experiments to explore the implications of running LOCKSS—a distributed peer-to-peer Digital preservation network—boxes in Amazon's cloud service; incidentally, their results indicate that cloud storage is more costly, in comparison to local storage. Even more encouraging are the emerging Digital Libraries cloud platforms, such as DuraCloud[2]. DuraCloud is a Web-based open technology platform aimed at facilitating long-term preservation of digital content for libraries, universities and cultural heritage organisations. It is offered as a service that replicates and distributes content across multiple cloud providers.

Most cloud service providers provide general architectural design guidelines to aid potential clients during the cloud migration process. In an AWS white paper (Varia, 2011), Varia provides comprehensive best practice design guidelines for building scalable applications above the AWS cloud services layer. Basing their work on the lack of cloud computing standards, Rimal et al. (Rimal et al., 2010) provide key guidelines for architects and application developers of cloud enterprise applications; in addition, they provide a classification of cloud computing architectural features according to end-user requirements.

One of the key concerns of cloud computing is performance evaluation of applications deployed in cloud environments. Various studies have been conducted, providing a basis and possible technique to use when conducting performance-based experiments. In a study that is very similar to this work in terms of evaluation, Moschakis and Karatza (Moschakis and Karatza, 2011) present performance evaluation of integrating mechanisms for job migration and handling of job starvation. Their evaluation—conducted through simulation under varying workloads, job sizes, migration and starvation handling schemes, however, did not show significant improvements in response times in a cloud environment. Khazaei et al. (Khazaei et al., 2012) present a novel approximate analytical model for performance evaluation of cloud server farms and solve it to obtain an accurate estimation of the complete probability distribution of the request response time, among other things. The results from the performance of the cloud server farm indicated that their proposed approximation method provided more accurate results for the mean number of tasks in the system blocking probability of immediate service and response times in the cloud (Khazaei et al., 2012). However, there were longer waiting times for clients in a cloud centre that accommodated heterogeneous services as opposed to its homogeneous equivalent with the same traffic intensity (Khazaei et al., 2012).

[1] http://citeseerx.ist.psu.edu
[2] http://www.duracloud.org

3. SYSTEM DESIGN

While the expected high-level design of Digital Library Systems is already well-established (Arms, 1995), achieving flexibility and high levels of scalability for cloud-based DLSes, in part, requires the use of appropriate architectural patterns for the service and storage layers of resulting systems.

3.1 System architecture

Previous work (Suleman, 2009) identified potential architectural designs that can be employed to implement on-demand Digital Library Systems within cloud infrastructure. Of particular interest are architectures that emulate parallel programming architectures such as: shared-memory machines and distributed memory machines. Four architectural designs—The Proxy Architecture, The Redirector Architecture, The Round-Robin Architecture and The Client-side Architecture—were considered as potential candidate architectural designs, in order to take advantage of the master/manager paradigm that they utilise. The master/manager paradigm enables the master node to steer/proxy connections and manage application servers, handle requests/responses, and monitor machines.

The Proxy Architecture, shown in Figure 1, with some aspects of the Client-side Architecture (Suleman, 2009), was adopted and used as the basis for the architectural design of the system.

3.2 Digital Library services

The proof-of-concept DLS was specifically implemented to leverage Amazon Web Services (AWS) cloud services. AWS provides a suite of services that are designed to solve application growth needs through on-demand scalability, processing and storage. Specifically, the AWS services outlined below were utilised, and Figure 2 shows the high-level architectural design of the proof-of-concept system.

- Amazon Elastic Compute Cloud (Amazon EC2)[3]—to provide resizable computing capacity
- Amazon Simple Storage Service (Amazon S3)[4]—a highly distributed data store, to enable storage and retrieval of large numbers of data objects
- Amazon SimpleDB[5]—a Web service, that would facilitate the lookup and querying of the stored structured data
- Amazon Elastic Block Store (Amazon EBS)[6]—to ensure storage persistent of the Amazon EC2 instances

[3] http://aws.amazon.com/ec2
[4] http://aws.amazon.com/s3
[5] http://aws.amazon.com/simpledb
[6] http://aws.amazon.com/ebs

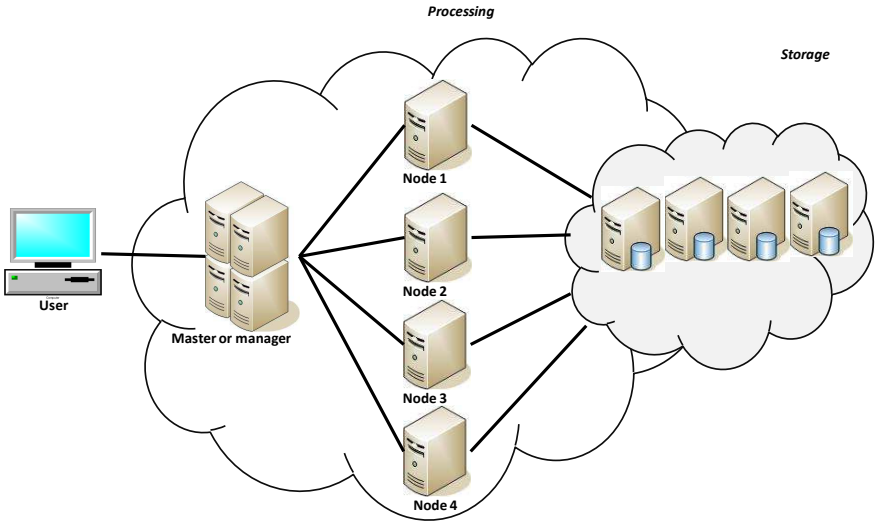

Figure 1: The Proxy Architecture: The "Master" or "manager" acts as a proxy between users and the cloud nodes

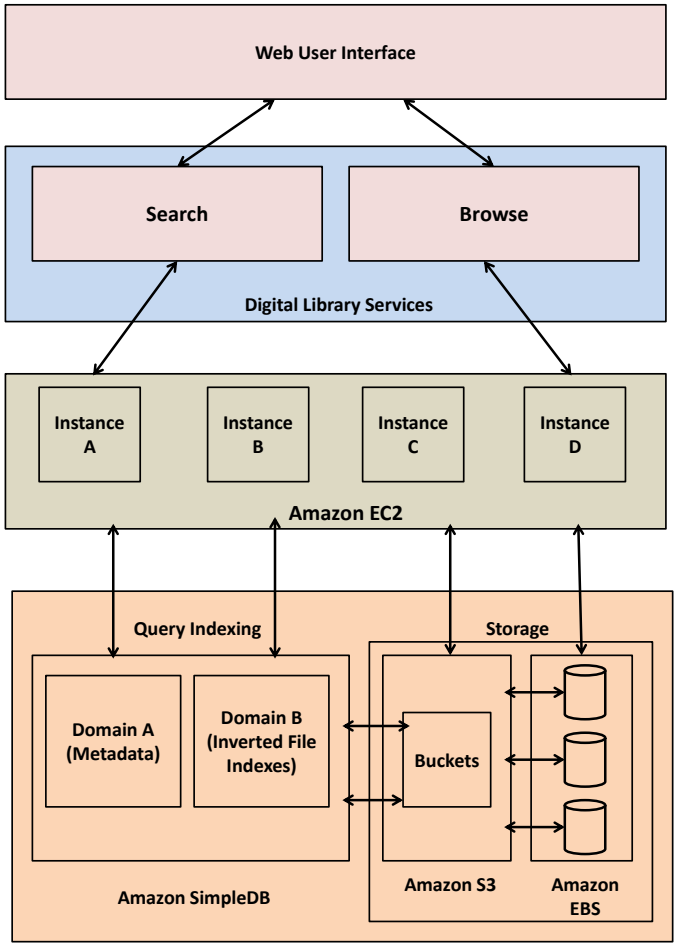

Figure 2: Conceptual design showing high-level system architecture

Two typical Digital Library services—browse service and search service—were implemented as part of the proof-of-concept system.

3.2.1 Browse module

Browsing is an information retrieval technique that enables the gradual and pre-determined refinement of end users' query requests by filtering through document subsets (Godin et al., 1989). The browse criteria are based on content categories common to collection digital objects. This effectively makes it possible for end users to narrow down to desired content through filtering.

The browsing service was implemented by specifying three browsing criteria fields: the document title, the document author and the document publication date. In addition, all three criteria can be browsed in ascending/descending order. The browse operation uses SimpleDB—described in Section 3.2.2—for indexing and querying operations, with communication occurring over a REST API.

Browsing is performed through the public light-weight Web user interface described in Section 3.3 and accepts end user queries to browse the collection by author, title and date. When the collection is browsed, users select the criteria they wish to browse by. The browse request is internally represented as a URL request and further transformed into a Amazon SimpleDB equivalent request used to filter matching digital objects in the collection. The matched results are returned in an XML response format, which are parsed and displayed back on the Web user interface.

3.2.2 Search module

The application was designed to store digital objects in Amazon S3, with Amazon SimpleDB used to index the metadata and corresponding documents to facilitate full-text search, using typical information retrieval algorithms. Figure 3 illustrates how data is stored on Amazon S3 and subsequently indexed by Amazon SimpleDB.

In order to facilitate query efficiency, an inverted index representation was created on Amazon SimpleDB by mapping the conceptual inverted file onto the attributes; this was achieved by associating a SimpleDB domain with one term, and subsequently have the attributes store the list of documents containing the word. Separate Amazon SimpleDB domains were used to store a URL table to reduce the inverted file size.

3.3 Web user interface

The Web user interface provides an entry point through which end users can access publicly available services—browse and search. The text-based Web interface was designed to be light-weight, and issues requests corresponding to user queries. The query results are then ranked and presented such that the highest ranked documents are displayed at the top.

4. EVALUATION

One of the motivations for the deployment of applications into the cloud is the promise of virtually unlimited scalability as data volumes increase. In order to evaluate potential

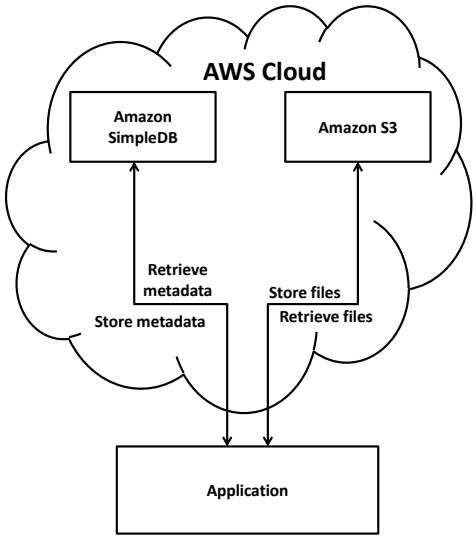

Figure 3: Data storage in the AWS cloud using S3 and SimpleDB

scalability advantages associated with cloud-based Digital Libraries services, a number of experiments were conducted to assess the linear scalability of data and service capacity needs. The experiments were specifically set up to evaluate data and service scalability, and additionally, to conduct processor load tests.

4.1 Experimental setup

All experimental tests were conducted on Amazon EC2 cloud infrastructure, using an Amazon EC2 instance of type `t1.micro` for server side processing. A 32-bit Ubuntu Amazon Machine Image (AMI) configuration was subsequently installed in order to correspond to development and test environments used during application development prior to deployment. Apache JMeter was used to simulate multiple user requests. The dataset used during experimentation was harvested via OAI-PMH from two publicly available spaces—The Networked Digital Library of Theses and Dissertations (NDLTD) portal[7] and The South African National Electronic Theses and Dissertation (NETD) portal[8] portal—as outlined in Section 4.1.1.

4.1.1 Test dataset

A metadata harvester Perl script was used to harvest metadata records from the NDLTD and NETD portals. The harvested metadata records were ingested into Amazon S3 using JetS3t2[9]—a free, open-source Java toolkit and application suite for Amazon S3, CloudFront content delivery network and Google Storage for Developers. The JetS3t toolkit provides Java programmers with a powerful, yet simple, API for interacting with storage services and managing data stored there. Figure 4 shows the simple metadata harvester that stored the harvested metadata records on Amazon S3.

4.2 Service scalability

[7] http://www.ndltd.org
[8] http://www.netd.ac.za
[9] http://www.jets3t.org

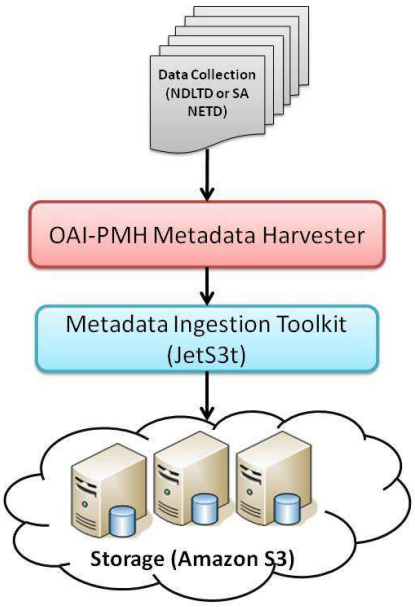

Figure 4: Metadata harvesting from NDLTD and NETD

The service scalability experiments were aimed at investigating the response times of DL services—search and browse—in a server farm configuration. Furthermore, the experiments were conducted to determine if multiple middleware servers affected the response times. In both the browse and search experiments, JMeter was set up to simulate 50 users accessing one Web service ten times for the browse function. The Web service was hosted on four identical Amazon EC2 instances. A logic controller was added in JMeter to ensure a round-robin order for the use of different servers. Five-run averages were computed for each request.

4.2.1 *Browse module*

A comparative analysis of three browsing categories was carried out by partitioning the results into blocks of 50 and taking the average of each block. This was necessary to obtain a better comparison of the average response time against the number of requests for each of the three browsing categories—author, title and date. Each category was individually tested because the performance of browse indices often relies on data type and complexity.

Figure 5 shows the results of 500 requests, clearly indicating a noticeable time for connecting to AWS services at the start of the experiment; this is in fact attributed to the pre-caching involved. Oscillations in response times for different browsing criteria suggest that there were multiple AWS back-end servers with caches that were not completely shared. The results further show that there were stable response times and that there was some variability because of multiple middleware servers, but nothing significant after the initial cache priming stage. Figure 6 is an alternative representation of the results, showing browse service performance for linearly increasing blocks of 50 user requests.

4.2.2 *Search module*

In the case of a search, the experiments were carried out at least five times for each of the five different types of queries listed below:

- Popular words comprised of frequently occurring words within the test dataset such as “computer”.
- Multiple popular words comprised of combined popular words such as “cloud computing”–both “cloud” and “computing” are popular words
- Unpopular words comprised of infrequently occurring words within the test dataset such as “immunochemical”
- Multiple unpopular words comprised of a combination of infrequently occurring words such as “carbohydrate conformations of pneumococcal antigens”
- Hybrid of popular and unpopular words comprised of a combination of popular and unpopular words such as “computational chemistry and immuno-chemical dynamics”

In general, larger inverted files contain the most popular words in documents. Therefore, the largest inverted file was the one with more occurrences of the popular words in a query string. Unpopular words on the other hand were contained in the smallest inverted files Query complexity was determined based on the size of the inverted file size when results were retrieved from S3. These categories were used to explore the space of inverted files of different sizes and varying amounts of post-processing.

As with the browse service, Figure 7 shows a noticeable time to connect to AWS services at the start of all experiments. Similarly, oscillations in response times for queries of different complexities suggest that there were multiple AWS back-end servers with caches of data that were not completely shared. Figure 8 show the case in which the data was partitioned into an average of blocks of 50 requests.

4.2.3 *Varying EC2 instances*

To ascertain how the number of instances, on which the application was run, impacts the response time when browsing or searching SimpleDB data, experiments were conducted while varying the number Amazon EC2 instances.

Figure 9 depicts the results obtained from varying the number of instances when searching the collection. In this case, the average response time is significantly higher when using one instance and it increases slightly as the number of requests increases. This is caused by a possible bottleneck as the instance gets over-whelmed with numerous requests, thus slowing it down. There is no distinct difference in response time between 2 and 3 instances. The performance of the application is similar in both cases and an additional fourth server shows a significant performance improvement. The average response time is lowest when using four instances, showing that the number of instances does impact on the application performance when searching a digital collection.

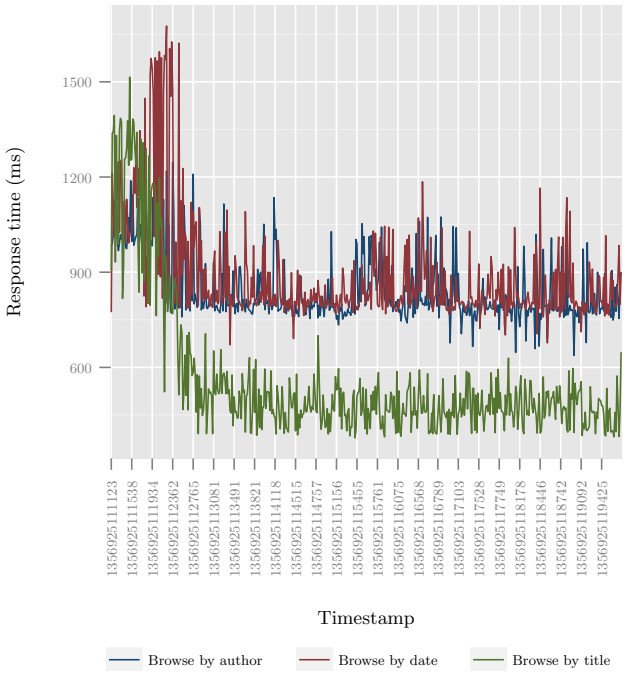

Figure 5: Plot of average time vs. request timestamp obtained from browsing the collection via different criteria

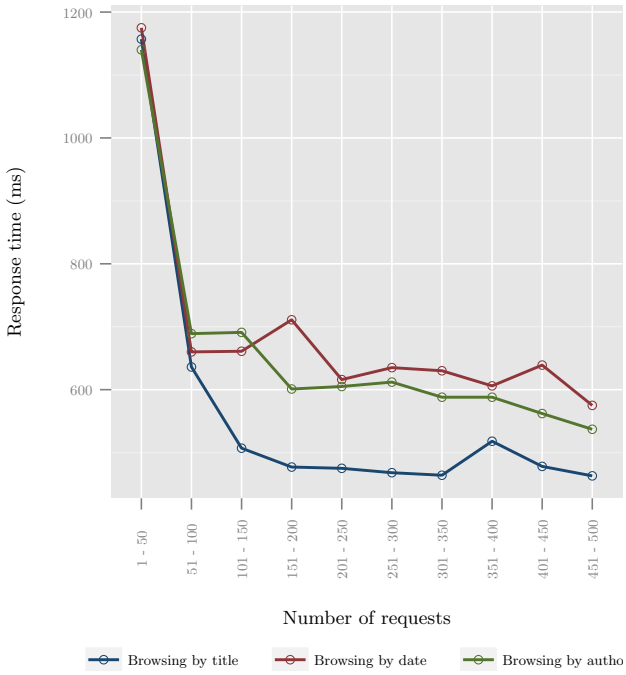

Figure 6: Combined plot of average time vs. number of requests processed when browsing browsing via different criteria

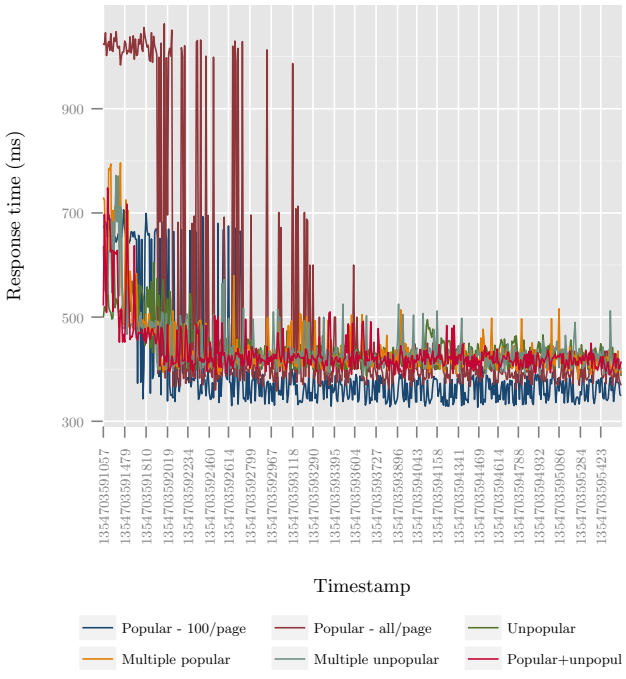

Figure 7: Plot of the average response time in milliseconds (ms) vs. request timestamp when processing queries of different complexities

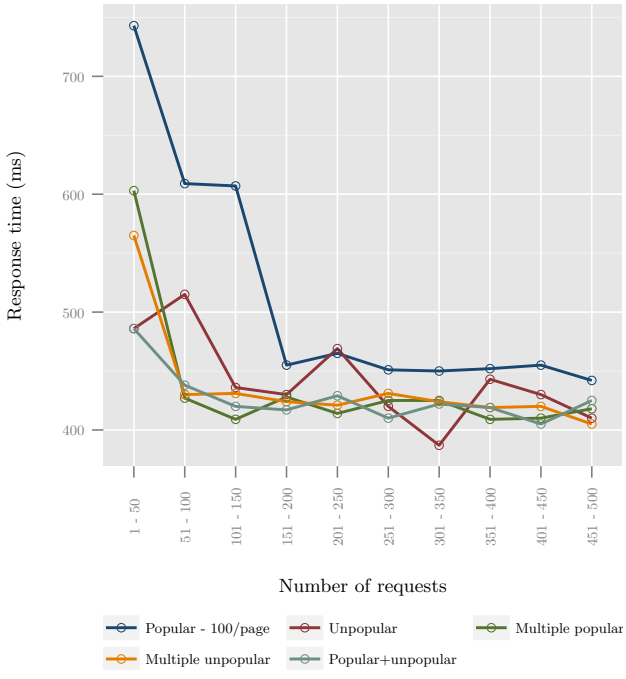

Figure 8: Plot of average response time in milliseconds (ms) vs. number of requests when processing queries of different complexities

Figure 10 is an illustration of a scenario in which the number of instances was varied when browsing the collection. The plot shows that using one instance suffers possible performance bottlenecks and therefore the average response time is higher than all the other three cases (two, three and four instances). With browsing, the difference is significant at the start up to 200 requests, after which there is no major difference when using two, three and four instances. However, a closer look at the results indicated that the case of four instances shows better performance in comparison to all the other cases, with an average response time of 483 milliseconds.

Finally, Figure 11 shows a speedup plot, illustrating the holistic results of the experiments when the number of EC2 instances was varied.

4.3 Data scalability

The data scalability experiments were conducted on linearly increasing collection sizes, in order to assess the impact of collection sizes on the performance of DL services—search and browse. These experiments were particularly aimed at revealing whether or not the application could cope with increasing volumes of data in digital collections.

JMeter was set up to simulate 50 users accessing a single Web service ten times. An important consideration was varying the number of servers. For instance, this first experiment involved the use of four identical servers and was run with collection sizes of 4000, 8000, 16000 and 32000 records. The experiment was run at least five times for each collection size. The average of the results was computed and, for each collection size, the average response time obtained was further partitioned into blocks of 50 requests to better illustrate the results.

4.3.1 Browse module

The average response time taken to browse the collection was plotted against the number of requests, and the results are as shown in Figure 12. There is a noticeable high response time due to the time taken to connect to AWS and because there was a large number of results retrieved when the collection was browsed. The average response time does not greatly impact on browsing collections of different sizes but more experimentation is needed with larger datasets.

4.3.2 Search module

Figure 13 shows the results, indicating marginal differences in the average response times.

4.4 Processor load stress-tests

Processor load tests were conducted to determine the volume of requests the application was capable of processing for an increasing number of concurrent users.

JMeter was set up to simulate different numbers of users accessing one Web service. The first scenario was a simulation of 5 users, each accessing the Web service 10 times. The second and subsequent scenarios were a simulation of 10, 20, 50, 100, 250 and 500 users, each accessing the Web service 10 times for a search query. The Web service was again hosted on four identical Amazon EC2 instances. In order to determine the order in which the Samplers are processed, a logic controller was added and, in this particular case, it was a Random Controller. The Random Controller picks a random sampler or sub-controller at each pass, so all the servers have an equal chance of being selected for processing. This experiment was repeated at least five times for search and browse for each of the 5, 10, 20, 50, 100, 250 and 500 users simulated. The overall average response time for each case was computed and used to generate a plot of average response time against the number users.

Figure 14 shows the results of the experiment. The results show that the average response times are relatively low when there are fewer users for both search and browse. The average response time shows a slight increase when the number of concurrent users is increased. As expected, the quantity and age of requests does not impact on the response times. However, more experimentation is needed with a simulation of a larger number of concurrent users.

4.5 Summary

Experimental results have shown that there are stable response times and some degree of variability because of multiple middleware servers when browsing and/or searching a collection of a fixed size. There are no significant response time differences after the results caching phase. When processing typical requests on varying collection sizes in the cloud, response times do not differ much, although more experimentation with larger datasets is needed. Lastly, request sequencing has shown that the quantity and age of requests does not have an impact on response times.

5. CONCLUSION

The cloud application proposed and developed on top of the Amazon Web Services cloud computing stack provides some insight into the pros and cons of developing Digital Library applications in AWS. The complexity of applications differs but development of digital library service components in the cloud has proven to be feasible, given the storage and computation services provided by AWS.

Performance evaluation of the application deployed in the cloud has shown that response times are not greatly affected by differences in request complexity, collection sizes or request sequencing. There is a noticeable time taken to connect to AWS services. In production systems this should really be persistent, like ODBC/JDBC connections in persistent database-driven Web applications. There is a ramp-up time where internal caching has a small impact on the results. This occurs consistently for all requests and request types. This will not affect busy services but may have a small impact on services that are rarely used. Oscillation in response times suggests that there are multiple AWS back-end servers with caches that are not completely shared. This results in a degree of unpredictability in results but this averages out over a period of time.

However, in developing the services, a number of features had to be redesigned from typical database-driven versions. This may constrain what is possible in, for example, query syntax, and may also affect the viability of other, less-popular, services because of the limitations of S3/SimpleDB.

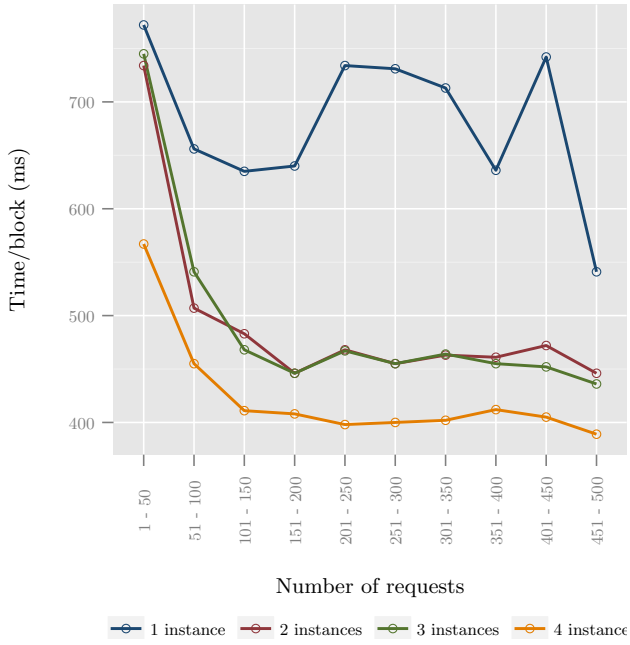

Figure 9: Plot of time in milliseconds (ms) vs. number of requests when searching the collection over a varying number of EC2 instances

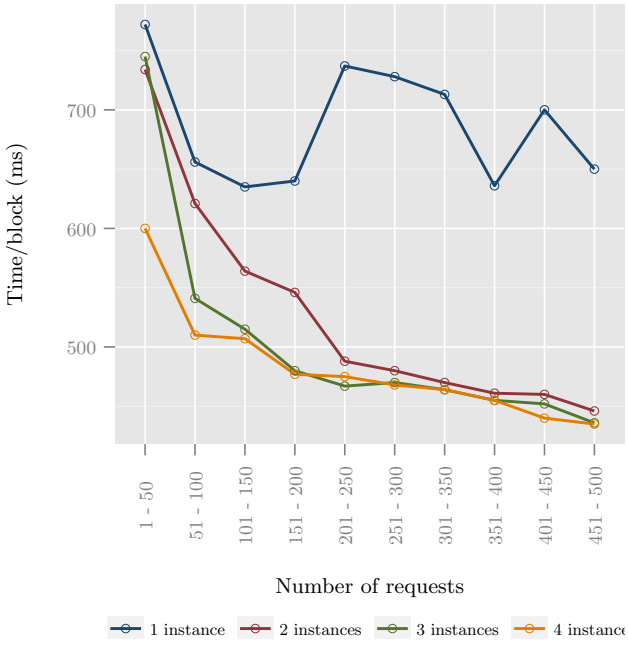

Figure 10: Plot of time in milliseconds (ms) vs. number of requests when browsing the collection over a varying number of EC2 instances

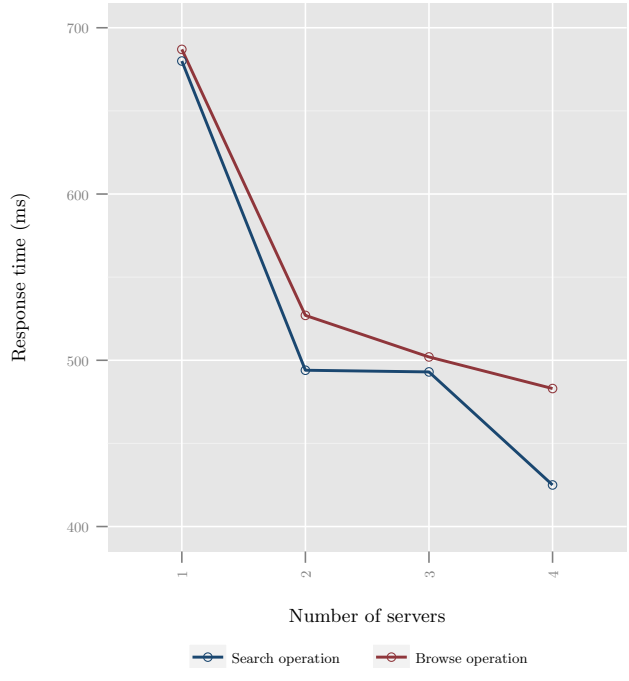

Figure 11: Plot showing speedup of average response times in milliseconds (ms) vs. the number of EC2 instances

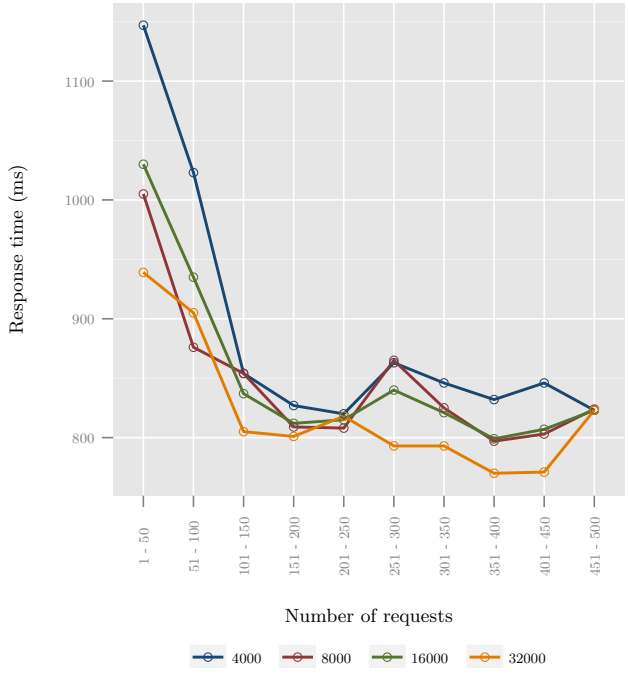

Figure 12: Plot of average response time in milliseconds (ms) vs. number of requests when browsing varying collection sizes

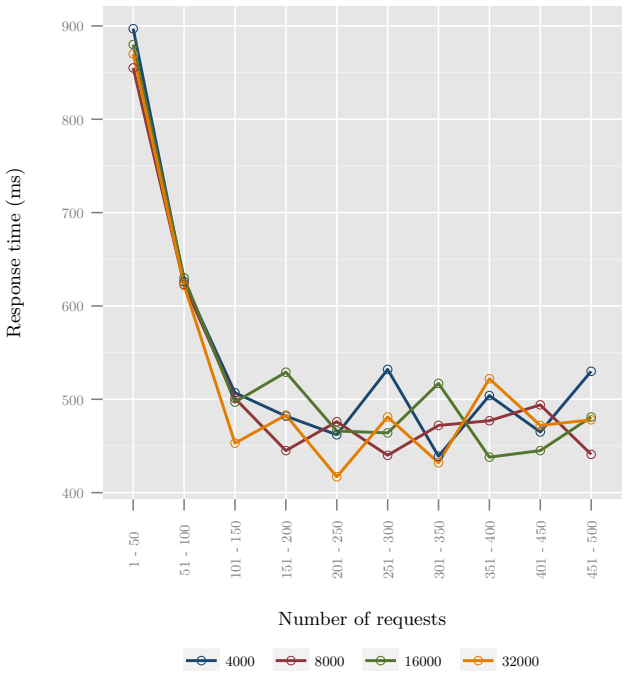

Figure 13: Plot of average response time in milliseconds (ms) vs. number of requests when searching varying collection sizes

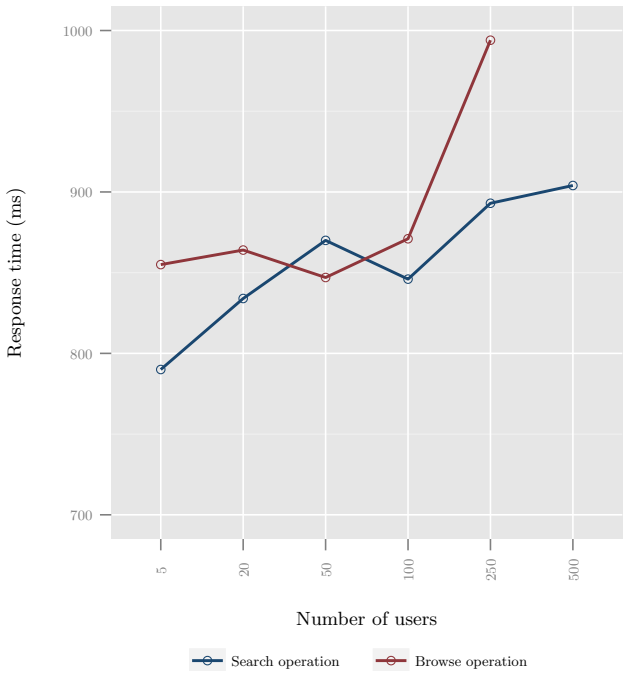

Figure 14: Plot of average response time in milliseconds (ms) vs. number of user requests when searching and browsing a collection

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. BIBLIOGRAPHY

[1]Michael Armbrust et al. “A View of Cloud Computing”. In: *Communications of the ACM* 53.4 (Apr. 2010), p. 50. DOI: 10.1145/1721654.1721672.

[2]William Y. Arms. “Key Concepts in the Architecture of the Digital Library”. In: *D-Lib Magazine* 1.1 (July 1995). DOI: 10.1045/july95-arms.

[3]R. Godin, C. Pichet, and J. Gecsei. “Design of a browsing interface for information retrieval”. In: *ACM SIGIR Forum* 23.SI (June 1989), pp. 32–39. DOI: 10.1145/75335.75339.

[4]Ali Khajeh-Hosseini, David Greenwood, and Ian Sommerville. “Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS”. In: *2010 IEEE 3rd International Conference on Cloud Computing* (July 2010), pp. 450–457. DOI: 10.1109/CLOUD.2010.37.

[5]Hamzeh Khazaei, J. Misic, and V. B. Misic. “Performance Analysis of Cloud Computing Centers Using M/G/m/m+r Queuing Systems”. In: *IEEE Transactions on Parallel and Distributed Systems* 23.5 (May 2012), pp. 936–943. DOI: 10.1109/TPDS.2011.199.

[6]Ioannis a. Moschakis and Helen D. Karatza. “Performance and cost evaluation of Gang Scheduling in a Cloud Computing system with job migrations and starvation handling”. In: *2011 IEEE Symposium on Computers and Communications (ISCC)* (June 2011), pp. 418–423. DOI: 10.1109/ISCC.2011.5983873.

[7]Bhaskar Prasad Rimal, Admela Jukan, Dimitrios Katsaros, and Yves Goeleven. “Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach”. In: *Journal of Grid Computing* 9.1 (Dec. 2010), pp. 3–26. DOI: 10.1007/s10723-010-9171-y.

[8]David S. H. Rosenthal and Daniel L. Vargas. “Distributed Digital Preservation in the Cloud”. In: *International Journal of Digital Curation* 8.1 (June 2013), pp. 107–119. DOI: 10.2218/ijdc.v8i1.248.

[9]Hussein Suleman. “Utility-based High Performance Digital Library Systems”. In: *Second Workshop on Very Large Digital Libraries*. 2009, pp. 1–8.

[10]Byung Chul Tak, Bhuvan Urgaonkar, and Anand Sivasubramaniam. “To Move or Not to Move: The Economics of Cloud Computing”. In: *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*. HotCloud’11. Berkeley, CA, USA: USENIX Association, 2011, pp. 5–5.

[11]Pradeep Teregowda, Bhuvan Urgaonkar, and C Lee Giles. “Cloud Computing: A Digital Libraries Perspective”. In: *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, July 2010, pp. 115–122. DOI: 10.1109/CLOUD.2010.49.

[12] Pradeep B. Teregowda, Bhuvan Urgaonkar, and C. Lee Giles. "CiteSeerx: A Cloud Perspective". In: *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 9–9.

[13] Jinesh Varia. *Architecting for The Cloud: Best Practices*. Retrieved August 23, 2014, from Amazon Web Services Blog. 2011. URL: http://aws.amazon.com/blogs/aws/new-whitepaper-architecting-for-the-cloud-best-practices.

[14] Christian Vecchiola, Suraj Pandey, and Rajkumar Buyya. "High-Performance Cloud Computing: A View of Scientific Applications". In: *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks* (2009), pp. 4–16. DOI: 10.1109/I-SPAN.2009.150.

[15] Jian Wu, Pradeep Teregowda, Kyle Williams, Madian Khabsa, Douglas Jordan, Eric Treece, Zhaohui Wu, and C Lee Giles. "Migrating a Digital Library to a Private Cloud". In: *Proceedings of the 2nd IEEE International Conference on Cloud Engineering*. Boston, MA, 2014. DOI: 10.1109/IC2E.2014.77.