Fine-grained Scalability of Digital Library Services in the Cloud

Lebeko Poulo, Lighton Phiri and Hussein Suleman

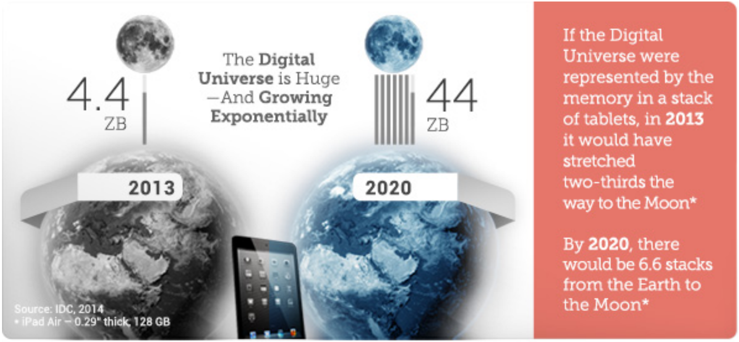
Digital Libraries Laboratory
Department of Computer Science
University of Cape Town

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Research Overview

- Digital Libraries (DLs) and Digital Library Systems (DLSes)
- Research objectives
  - Develop techniques for building scalable digital information management systems based on efficient and on-demand use of generic grid-based technologies
  - Explore the use of existing cloud computing resources
- Research questions
  - Can a typical DL architecture be layered over an on-demand paradigm such as cloud computing?
  - Is there linear scalability with increasing data and service capacity needs?

How Quickly Does Data Scale?



- Extent of data scalability
  - Data growth rates estimated at 40% per year
  - By 2020, data volumes will have grown to 44 times the 2009 size

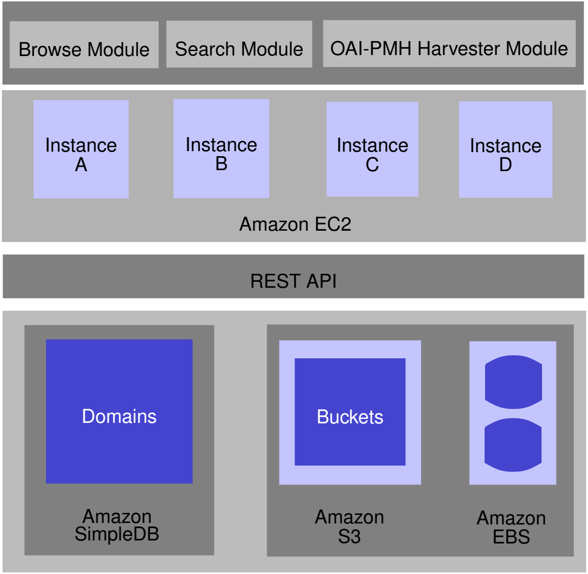
Scaling Digital Library Systems

- Key criteria for design/implementation of DLSes
  - Scalability
  - Preservation
- The promise of cloud computing proven many times
  - Feasibility of migrating and hosting DLs evident
- Investigation of deep integration of DL services with cloud services required
  - Investigate efficacy of DL cloud adoption
  - Verify extent of unlimited scale
  - Maximise potential for cloud-service-level scalability

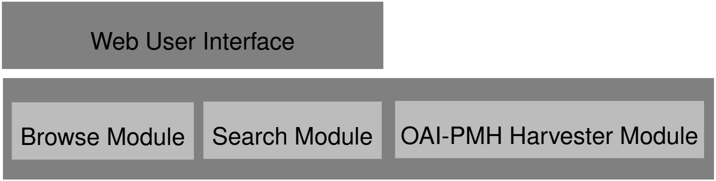
Prototype DLS - Design

- RQ #1—Can a typical DL architecture be layered over an on-demand paradigm?
- Prior work on potential architectural designs for utility clouds
  - Emulation of parallel programming architectures
  - Utility computing offers flexibility of multiple architectural models
  - Potential architectures for scalable utility services
- Two architectural patterns adopted as basis for design of prototype architecture
  - Proxy architectures
  - Some aspects of Client-side architecture

Prototype DLS - Architecture

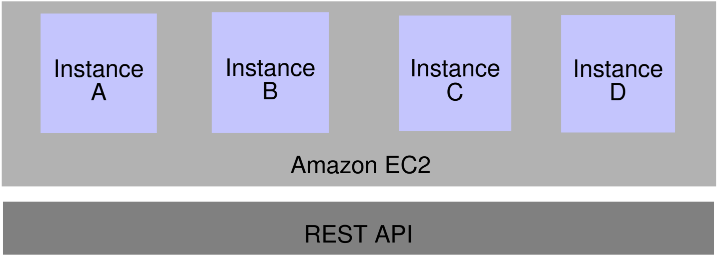


Prototype DLS - Services



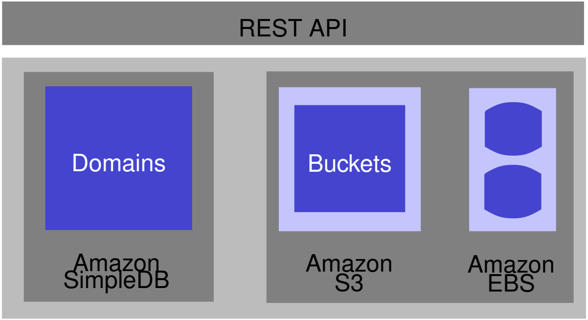
- Two typical DL services, accessible via publicly available Light-weight process Web interface
  - Browse module—enable access through gradual refinement
  - Search module—enable access through search queries
- OAI-PMH endpoint used to ingest data into collections

Prototype DLS - Application Server



- Amazon Elastic Compute Cloud (EC2) to provide sizeable computing capacity
- 32-bit Ubuntu Amazon Machine Images (AMIs)
  - Glassfish 3.1
  - Prototype DLS

Prototype DLS - Data Storage



- Amazon Simple Storage Service (S3) for storage and retrieval of large numbers of data objects
- Amazon SimpleDB for querying stored structured data
- Amazon Elastic Block Store (EBS) to enable storage persistence of EC2 instances

Evaluation - Experimental Design

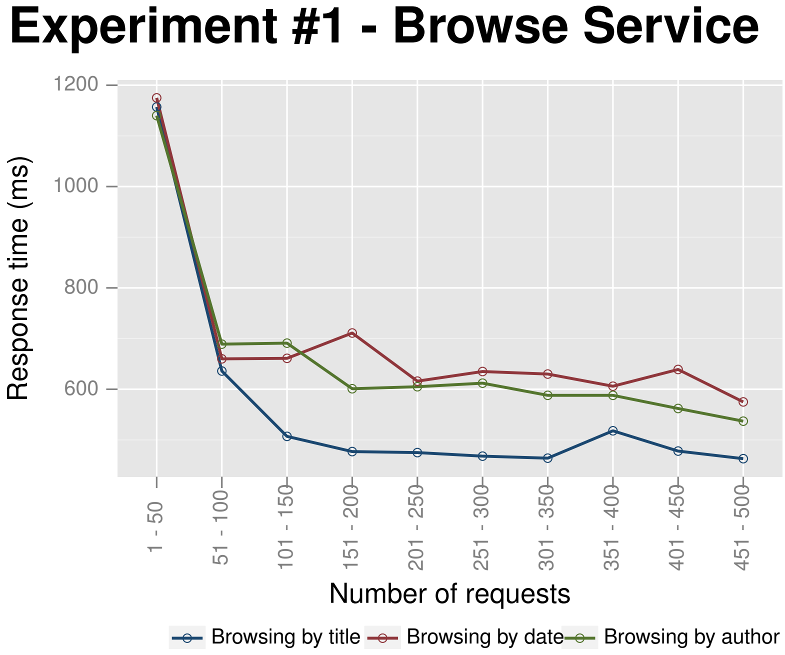
- RQ #2—Is there linear scalability with increasing capacity needs?
- Goals
  - Evaluate potential scalability advantages associated with cloud-based DLs
- Evaluation aspects
  - Data/service scalability and load testing
- Workload
  - Number of user requests, number of users and collection sizes
- Metrics
  - Response time
- Factors
  - EC2 instances, users, requests, collection size

Evaluation - Experimental Setup

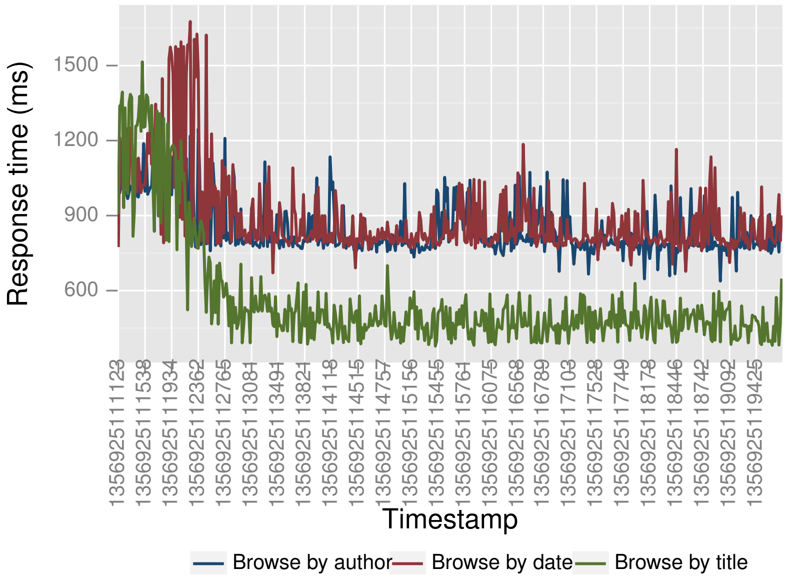
- Test dataset—NDLTD and NETD portals
  - Ingested using OAI-PMH harvester module
- Execution environment
  - All experimental test conducted on EC2 cloud infrastructure
  - EC2 instance of type `t1.micro` used for server-side processing
  - 32-bit Ubuntu Amazon Machine Image (AMI) configuration
- Apache JMeter used to simulate user requests
- All measurement results based on five-run averages

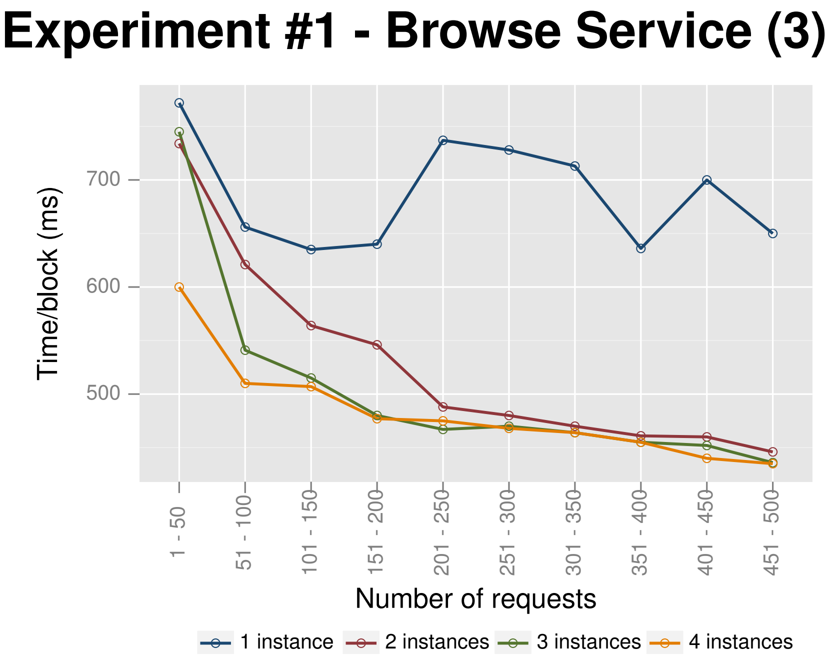
Experiment #1 - Service Scalability

- Determine the time taken for browse and search service requests
- Assess impact due to variation of multiple server front-ends
- Methodology
  - JMeter used to simulate 50 users for each Web service, ten times
  - Web services hosted on four identical EC2 instances
  - Experiments repeated at least five times for each service criteria
  - Comparative analysis—browsing categories for browse service—by partitioning requests into blocks of 50


Experiment #1 - Browse Service
Response time (ms)
1200
1000
800
600
1 - 50
51 - 100
101 - 150
151 - 200
201 - 250
251 - 300
301 - 350
351 - 400
401 - 450
451 - 500
Number of requests
Browsing by title
Browsing by date
Browsing by author

Experiment #1 - Browse Service (2)

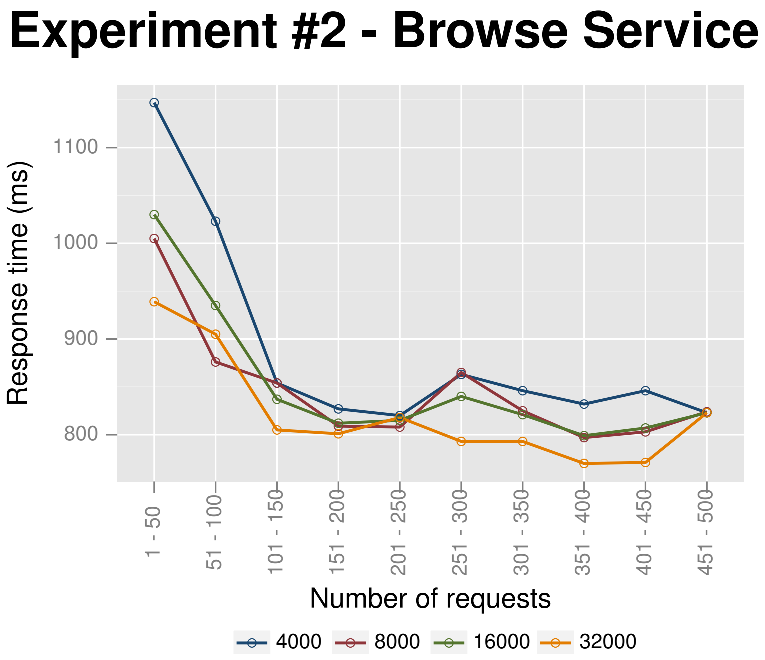




Experiment #1 - Browse Service (3)
Time/block (ms)
700
600
500
1 - 50
51 - 100
101 - 150
151 - 200
201 - 250
251 - 300
301 - 350
351 - 400
401 - 450
451 - 500
Number of requests
1 instance
2 instances
3 instances
4 instances

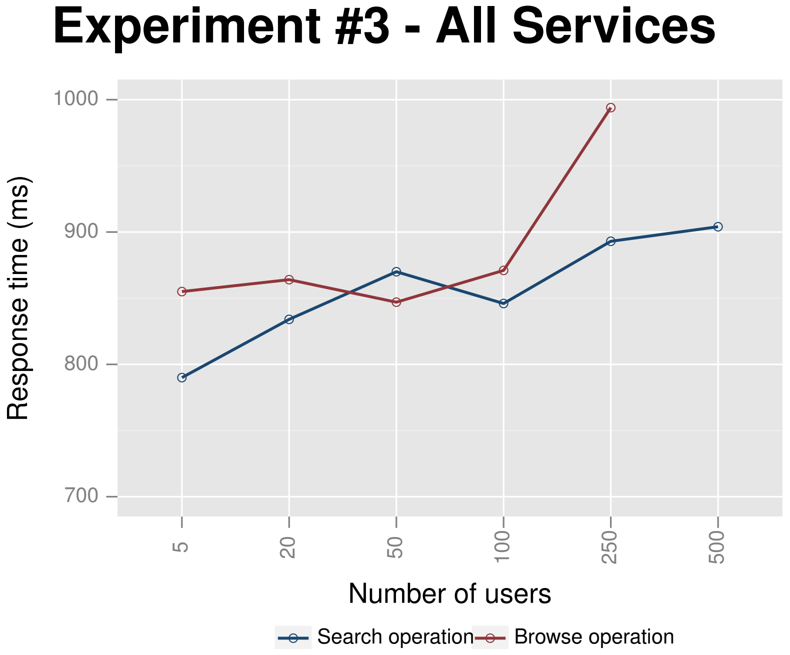
Experiment #2 - Data Scalability

- Determine service performance for varying collection sizes for fixed number of servers
- Ascertain if application can cope with increasing data volumes in DL collections
- Methodology
  - JMeter set up to simulate 50 users accessing a Web service ten times
  - Fixed number of identical servers with collection sizes of 4k, 8k, 16k and 32k records
  - Experiments repeated at least five times for each service
  - Comparative analysis by partitioning requests into blocks of 50


Experiment #2 - Browse Service
Response time (ms)
1100
1000
900
800
1 - 50
51 - 100
101 - 150
151 - 200
201 - 250
251 - 300
301 - 350
351 - 400
401 - 450
451 - 500
Number of requests
4000
8000
16000
32000

Experiment #3 - Load Testing

- Determine volume of requests application could process for increasing concurrent users
- Methodology
  - JMeter set up to varying number of users accessing a Web service
  - Fixed number of identical servers used
  - Initially simulate five users, each accessing a Web service ten times
  - Subsequent simulation of 20, 50, 100, 250 and 500 users
  - Experiments repeated at least five times for each service



Experiment #3 - All Services
Response time (ms)
1000
900
800
700
5
20
50
100
250
500
Number of users
Search operation
Browse operation

Conclusion

- Key findings
  - Redesign of application architectural components to conform to cloud service architecture
  - Results indicate that response times are not significantly affected by request complexity, collection size or request sequencing
  - Noticeable time taken to connect to AWS—ramp up time
- Study Limitations
  - Single EC2 instance type—`t1.micro`—used
  - Cloud service vendor
  - Experimental dataset size
  - Query optimisation
  - Synthetic load used

Bibliography

- Hussein Suleman (2009).
  Utility-based High Performance Digital Library Systems.
- Pradeep Teregowda et al. (2010).
  Cloud Computing: A Digital Libraries Perspective.
- Pradeep Teregowda et al. (2010).
  CiteSeerx: A Cloud Perspective.
- Byung Chul Tak et al. (2011).
  To Move or Not to Move: The Economics of Cloud Computing.
- Jinesh Varia (2011).
  Architecting for The Cloud: Best Practices.

Questions?

Additional information

http://dl.cs.uct.ac.za