CRITICAL ANALYSIS OF ECM APPLICATIONS IN THE CLOUDS: A CASE STUDY

James Xue[1] and Amjad Yahya[2]

[1]Department of Computing, University of Northampton, UK
james.xue@northampton.ac.uk
[2]PricewaterhouseCoopers, Saudi Arabia
amjadyahya@yahoo.com

ABSTRACT

Enterprise Content Management (ECM) has become one of the most important strategies in enterprise to manage enterprise contents over their lifecycle. ECM solutions are commonly used in many areas such as document management, web content management, record management, digital asset management, etc. Key features of ECM solutions are capturing, indexing, preserving and retrieving of digital information. In our work, an Electronic Dissertation and Thesis (EDT) application has been chosen as a typical ECM application for the case study and has been developed using state-of-the-art technologies. This paper considers the key characteristics of EDT applications and critically analyses various cloud options for deployment of such an application to maximise availability, scalability and data consistency. This paper also addresses the limitation of some existing cloud services that are related to query documents, and provided a workaround solution for abstract queries and parallel execution of complex queries.

KEYWORDS

ECM, resource management, virtualisation, cloud computing, hybrid clouds.

1. INTRODUCTION

Content management is maintaining, organising and searching across information sources, which could be structured (database) or unstructured (such as documents, emails and web pages) [6]. Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organisational processes [4, 17]. ECM involves not only technical but also strategic aspects of management of enterprise contents over their lifecycle, therefore is critical to the success of an enterprise. Some key features of ECM systems are:

- *Capturing.* The process of creating and obtaining the new content either from electronic source as an email or electronic document or from paper source by scanning process.
- *Management.* The process help organisations to better manage the creation, revision and approval of electronic documents. Important features implemented in this process like check in, checkout, version control, determining who will access and how it will be reused.
- *Storage & Preservation.* The way of saving the unstructured content in structured format taking consideration the content size and adding compression techniques as well as encryption mechanism to add security at the repository level. Classification and retention will be considered at this stage.

- *Delivery.* The way of providing the end users with the requested information, either using search template, ad hoc search or another technique.

Traditionally ECM applications are deployed in dedicated servers. For such deployment, considerations need to be taken to address issues such as capacity provisioning, load balancing, fault tolerance, etc. Although the issues can be resolved in some way, it can be expensive and time consuming to upgrade or add software/hardware components and it can take a few hours or even days.

As the number of documents increases, the storage requirement can grow rapidly. Suppose during the first year, there are 10000 documents uploaded (this assumption is realistic for an Internet-based application). If the database size for one document is around 3MB, a total of 30GB of storage is required to keep the electronic documents. The storage requirement is higher when more documents are to be uploaded. Clearly, fixed storage will no longer be sufficient in such cases; other solutions need to be provided. Among the solutions, cloud storage seems to be a good choice as the storage can be increased on demand with minimum disruption to the service.

As the advance of virtualisation [8, 9, 18] and cloud computing [3, 7, 10, 14] technologies, it provides a good opportunity to use the 'clouds' to address resource management issues in enterprise systems. The goal of cloud computing is to enable IT organisations to offer a cost-effective, elastic provision of IT services that are good enough for business. As physical resources can be virtualised and virtual instances can grow and shrink within seconds to cope with changes of workload demand, therefore operational costs can be reduced. Resource management in the clouds is transparently to the end users as virtual instances can be created and destroyed in response to workload changes. It also offers service orchestration, which helps IT organisations to automate many of the manual tasks that are involved in provisioning and controlling the capacity of dynamic virtualised services.

Cloud services can be categorised to three different services: Infrastructure as a Service (IaaS) (e.g., Amazon EC2), Platform as a Service (PaaS) (e.g., Azure platform), and Software as a Service (SaaS) (e.g., Salesforce, Dropbox). Figure 1 depicts three main different cloud services and how service purpose flexibility varies among them.

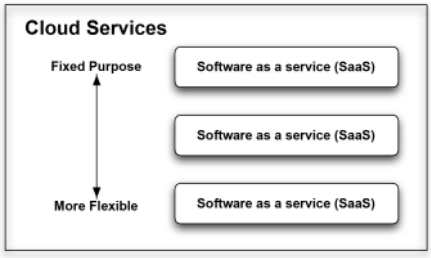


Figure 1. The cloud services.

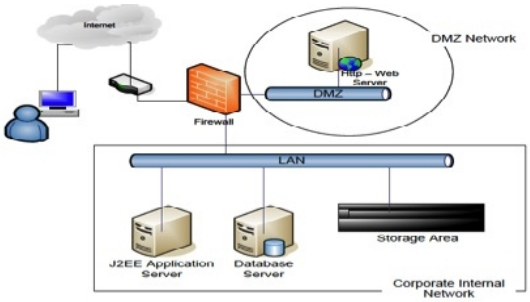


Figure 3. Local deployment of EDT systems.

In this paper, an Electronic Dissertation and Thesis (EDT) application has been chosen as a typical ECM application for the case study. It has been developed using a number of state-of-the-art technologies (e.g., J2EE, Hibernate [11], ICEfaces [12], Java Persistence APIs, etc.) for robustness of the application. The purpose of this paper is to find out how to utilise the available cloud services to achieve high scalability, availability and data consistency. Amazon cloud services are used in our work for the well-establishment, rich APIs and large adoption by organisations around the globe. The principles of the analysis can be applied to similar ECM applications using other cloud services.

The following Amazon services are analysed in our case study: a) the Simple Storage Service (S3) is a storage service that provides a simple web service interface for storing and retrieving information from the Web; b) the SimpleDB is a highly scalable and flexible non-relational data store that is commonly used for storing unstructured information such as web pages, log files; c) the Elastic BeanStalk (EBS) provides a quick way for deployment and management of Java web applications in the AWS cloud. It helps to achieve availability and scalability by defining auto-scaling group policy based on specific scaling triggers such as request count or CPU utilisation, etc. One of the main purposes of this paper is to analyse the available cloud services and find out how they can be used be used effectively for EDT applications.

The contributions in this paper can be summarised as: a) identification of the main characteristics of EDT applications and prototype deployment based on the findings using state- of-the-art technologies; b) critical analysis of various cloud services and how can be best used for ECM applications; c) this paper also addresses the limitation of some existing cloud services that are related to query documents, and provides a workaround solution for abstract queries and parallel execution of complex queries.

The remainder of this paper is organised as follows: section 2 gives an overview of related work; section 3 compares the similar systems and identifies essential features to be implemented in our prototype system; section 4 describes various solution options; section 5 describes the solution design; section 6 gives an overview of the solution architecture; section 7 describes the solutions for querying documents using metadata; and section 8 concludes this paper.

2. RELATED WORK

ECM and content management in general have attracted enormous research interests in the last decade. [5] is an open- source content management system with portal functionality. It was initially only used by public broadcasters, MMBase has been adopted by a growing number of organisations. In [16], the authors examine a strategic development and implementation process of enterprise content management (ECM) in a large oil company. Alfresco [1] is the open-source ECM solution. It makes use of Amazon EC2 as cloud computing platform and uses RighScale [13] to bridge the application and the infrastructure. Alfresco is similar to our work, however our work focuses on a specific type of ECM applications, and storage is the main consideration of this study.

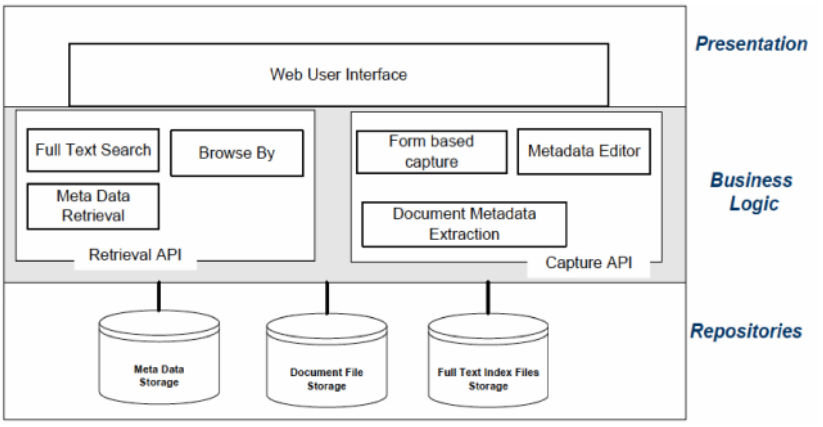


Figure 2. Electronic Dissertation and Thesis Layers.

3. COMPARABLE SYSTEMS INVESTIGATION

In this section, an investigation will be conducted to identify the common features in available EDT systems, which will be the basis of the prototype application to be implemented for the case study. Using this type of investigation it is also possible to discover industry trends by looking at several different systems, thus the required functionality can be altered accordingly.
Table I shows a list of common features in existing EDT applications:

- *Upload by students.* This feature allows university students to share their thesisses or dissertations.
- *Keyword search.* This implies that EDT system has implemented full text search behind the scene in which the system will search across the repository metadata and the content of the document based on the keyword criteria entered. This feature will consume more computing power and it will need intensive resources.
- *Metadata search.* The end user of the system can filter the search results based on metadata criteria in which might contains Author name, Title, Year and Subject.
- *User profile.* A user profile will be created to each registered user in the system, this enhance the system usability, whereas the end user can manage the uploaded documents and user profile upon log on.
- *Browse by.* By adding this feature end user will be able to browse the documents by title or by university name alphabetically.

The architectures of EDT systems are three-tier layers: presentation, business logic and database as can be seen from Figure 2.

Table 1. List of features of different EDT applications.

Website Feature	Metadata search	Document by students	Keyword search	User profile	Browse by
OpenThesis	x	x	x	x	
NETD	x		x		x
CollectionsCanada	x		x		

4. SOLUTION OPTIONS

4.1. Local Deployment

Before moving to the cloud, when the EDT application is deployed, we need to take care of securing the internal network of the organisation that is hosting the application. In order to fulfil the best practice, De Militarised Zone (DMZ) concept will be applied to protect the internal network from malicious external attack. In addition, capacity planning has to be done as documents are constantly added to the system. Figure 3 depicts the deployment architecture to implement the solution.

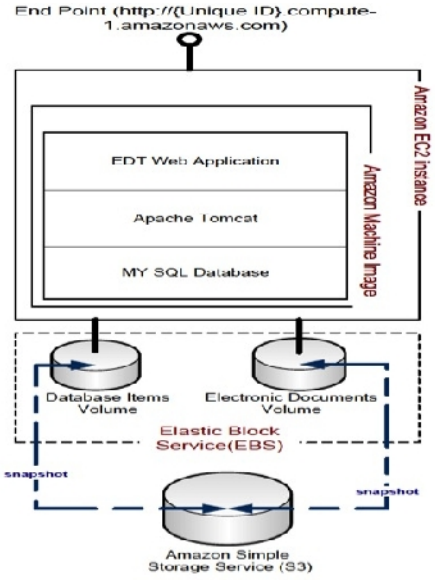


Figure 4. Single instance cloud deployment.

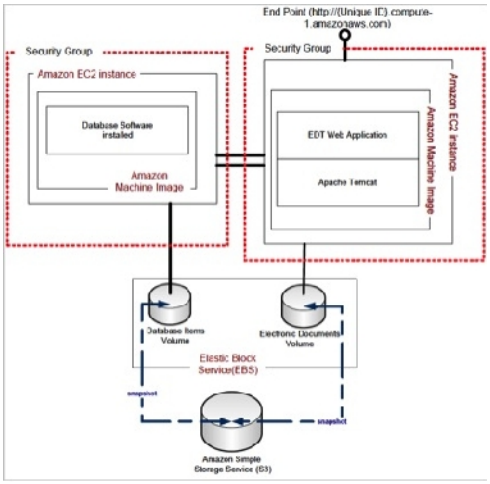


Figure 5. Multiple instance cloud deployment.

4.2. Single Instance Deployment

For this deployment, an Amazon Machine Image (AMI) needs to be created and attached to the elastic block storage. The AMI then needs to be uploaded into Amazon S3 bucket, from which Amazon EC2 will launch the machine image when needed. The configuration of database to store the data files is the same as if it was done in the traditional environment. Backup of machine images in S3 is done automatically, and backup of data files in the mounted volume can be done using snapshots, which can be developed based on elastic block service APIs.

This solution is not different that much, from the traditional deployment out of cloud, but this solution can give the ability to expand the storage and scale out the solution by adding more instances in couple of minutes instead of spending days in the traditional environment and that is one advantage of virtualisation technology used in the cloud.

From storage point of view, Amazon provides two options for persistence storage: Elastic Block Service (EBS) and Simple Storage Service (S3). To implement the solution above we have the following theoretical options to distribute the application data among:

- Storing database and electronic files on S3.
- Storing database on EBS and electronic files on S3.
- Storing database and electronic files on EBS, creating backup snapshot on S3.

By analysing the above options, we can notice that the first option is not feasible because of storing the database files on S3 bucket firstly need to alter the database software to consume S3 web services and put the database files in S3 bucket; secondly, from performance perspectives it will kill the performance and cause degradation in applications performance as well as it might reach time out because of the high latency between the communication of the EC2 instance and the S3 buckets.

In the second option, storing the Database files on EBS is the correct way to deal with the database file and keep it persistence, not dependent on the instance life span and within a

reasonable database access performance. However, storing the Electronic files will affect the performance of adding and retrieving the files but it will assure high availability and durability for the data. So it rely on the application purpose to either use EBS or S3 as file storage, As in [15], the average bandwidth between the EC2 instance and the S3 buckets is around 15 MB/s while the latency is high, on other hand, the average bandwidth between EC2 instance and EBS is around 50 MB/s with low latency [2, 15].

The third option to store both the database files along with the electronic files on the attached EBS drive, is the best option from performance wise, but if we need to maintain an appropriate level of data durability and availability we need to use S3 service to store the manually created snapshot of the EBS volumes. By this S3 will provide the durability for the backed up EBS volumes.

4.3. Design for Scalability

Since the chosen EDT application has multi-tiered architecture, scalability can be achieved either at the application tier or the database tier, depending on the system bottleneck.
To achieve high scalability, multiple instances can be used for the deployed application using auto-scaling group. The user can configure the thresholds of certain parameters (e.g., CPU utilisation, memory usage, etc.), which are used to determine when to create new EC2 instances. The auto scaling service monitors the system state, creates or remotes instances de- pending on the pre-configured thresholds. For instances, a new instance can be created when the CPU utilisation reaches 80% and an existing instance will be removed when CPU utilisation is below 20%.

4.3.1. Scaling the Database Tier

In order to scale the database, first of all the database server need to be run on separate instance of the application. To take the database server in separate instance in Amazon Cloud we have the following options:

- AMI loaded with database software.
- Amazon Relational Database Services (RDS).
- Amazon SimpleDB service.

Figure 5 illustrates the deployment view for the first option by taking the database server on separate EC2 instances. Using separate EC2 instance dedicated for serving database request lead to decouple the database tier from the application tier and let instances communicate via TCP channel based on the database software installed on the Machine image. Auto- scaling service can be used to scale up or down based on the triggered events.
Since the EDT is document-based application with a number of metadata big tables and no relations between the tables, therefore the second option is not suitable. Instead Amazon SimpleDB can be used to fulfil the requirement. Figure 6 shows the use of SimpleDB for the EDT application. By using SmpleDB the ability to utilise database-partitioning concept is available and can be used to increase the workload throughput.

4.3.2. Scaling the Application Tier

Scaling the application tier can be achieved using the autoscale service based on user-defined policies, health checking, and schedule monitored by the autoscale service. Amazon provides Elastic BeanStalk (EBS) service for enterprise Java deployment environment. EBS is good for

performance and persistence storage. How- ever, in terms of EDT, we cannot rely on EBS for the electronic documents as there is no alternative shared storage with equivalent performance. Therefore, designing of the application to comply with service oriented architecture or loosely coupled implementation in our case is desirable.

4.3.3. Elastic Load Balancer

Building fault tolerant solution in the traditional way requires a software/hardware load-balancing tool, which placed on front of the hosting servers from which the traffic will be distributed based on the load-balancing algorithm used. In similar way Amazon has provided Elastic Load Balancing in which Amazon tried to cover most of the required features to implement automatic distribution of the traffic among different EC2 instances located either in the same or different availability zone.

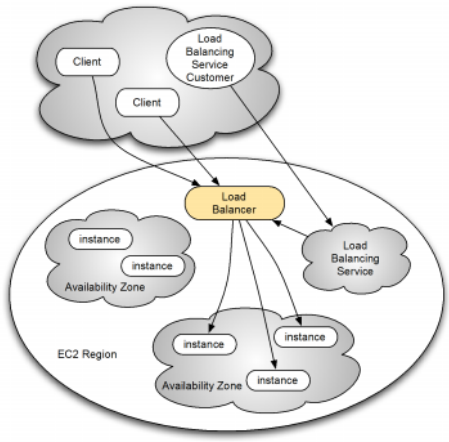


Figure 7. Elastic load balancer.

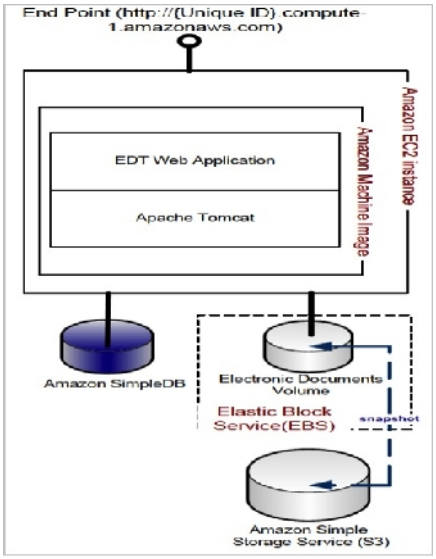


Figure 6. EDT application with SimpleDB.

5. SOLUTION DESIGN

5.1. System Sequence Diagram

As we can notice from the diagram (Figure 8) that the upload process start when upload event triggered by the end user by clicking on upload button, in which the associated backing bean configured to handle the event by executing the up- loadActionListener method. We can summarise the sequence of uploading in the following steps: 1) handling the trigged upload event; 2) validation the document content and its attributes against the required business rules; 3) preparation of the associated valid EDT document object with its attributes; 4) storage of the created EDT object in Amazon S3 bucket; 5) insertion of the associated EDT document metadata in Amazon SimpleDB domain.

In the above step we have executed the storing in Amazon S3 bucket before inserting the associated database record in simpleDB. This execution sequence assure to us in case of sudden failure happen during the storing in S3 that the failed document will not appear in the database in which we rely to query the available document. By all means, we consider that the document is saved in the solution after execution the last step, which is creating database record successfully.

As we don't want to have records in the database with no associated document content which may lead to false search result during search process.

The sequence diagram (Figure 9 show the sequence of searching process, which relies on Amazon SimpleDB records, since all the stored documents have their metadata stored in SimpleDB. The searching process can be summarised as: 1) End user fill the filtration fields and click on search button; 2) The searchActionListener methods handle the triggered search event; 3) Validation of the searching criteria fields; 4) DBUtil class object prepares the associated query string and its parameters; 5) DBUtil class object executes the searching and returns the Result Set.

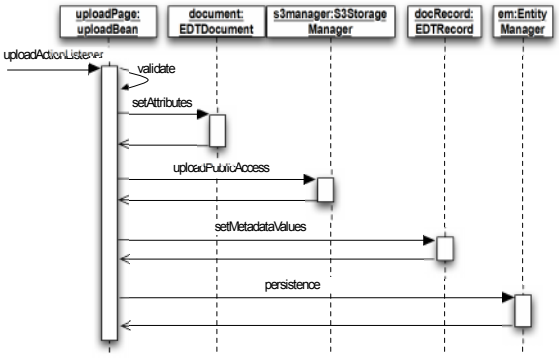


Figure 8. Upload process sequence diagram.

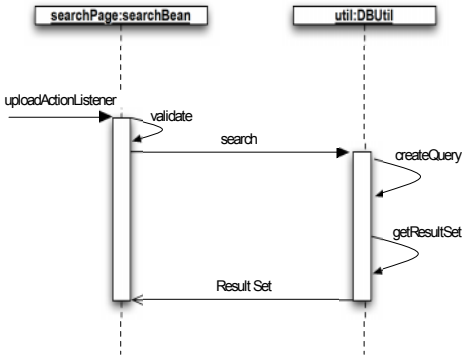


Figure 9. Search process sequence diagram.

5.2. Storage Access Layer

In order to access the S3 buckets during the storing process, a package (Figure 10) has been developed using Amazon Service Toolkit (AWT) SDK. Two main classes in the package are: EDTDocument and S3StorageManager. EDTDocument encapsulates the essential properties used during the storing process like MIME type, bucket name, storage path and its content. S3StorageManager is used to interface with Amazon S3 storage service to manage the saving and writing the content to S3 bucket as well as assigning specific access control list on the created entry.

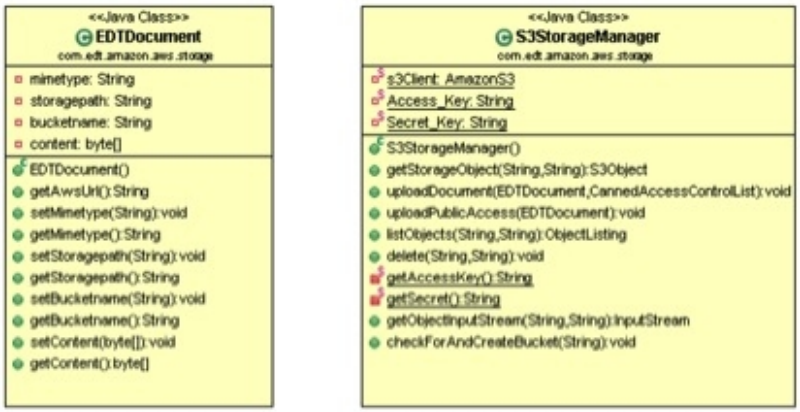


Figure 10. Storage access Java APIs

6. THE SOLUTION ARCHITECTURE

Figure 11 shows the final solution architecture of our case study, which summarises the analysis in previous sections. Amazon simpleDB web service is to be used to store the application metadata. Amazon S3 service to provide the file storage where document content stored by web service put request and retrieved by direct access to S3 service using HTTP end point provided by S3 service. Amazon elastic beanstalk service used to contain and manage the web application.

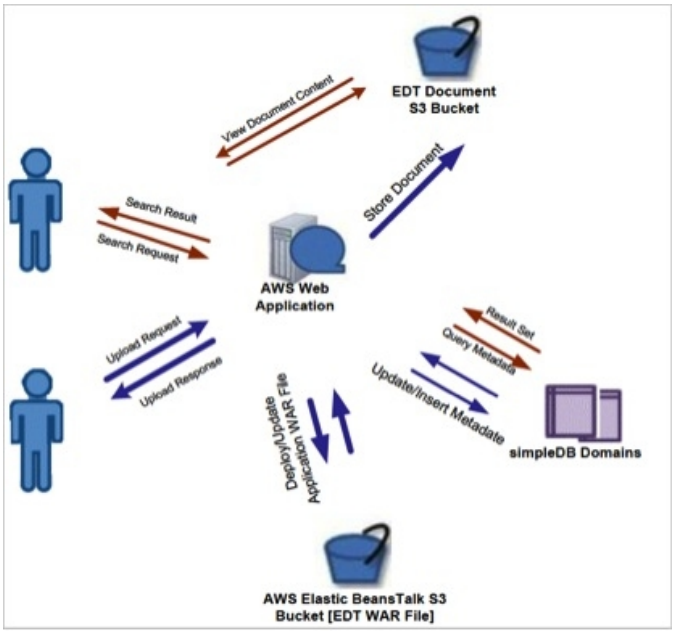


Figure 11. EDT solution architecture.

7. QUERYING USING METADATA

7.1. The Workaround Solution

A limitation of SimpleDB is that it does not support full text indexing to allow full text search. Our application implements this by creating generic database query, which include all available attributes in SimpleDB domain: abstract, title, author and keywords. In our case, the code to support the functionality looks like this:

*SELECT * FROM EDT-EDTrecord WHERE 'discipline'=:discipline AND ('abstract' LIKE '%: keyword %' OR 'title' LIKE '%:keyword%' OR 'author' LIKE '%:keyword%' OR 'keywords' LIKE '%keyword%');*

From perspective of SimpleDB attributes size, SimpledDB limits the maximum size for single attribute value to be 1024 byte. In our case, to allow the end user to enter more than 1024 character in abstract field will definitely break the limits of simpleDB. In order to overcome this limitation, instead of representing abstract field in simpleDB as one attribute we distribute its value among at max four attributes. Figure 12 gives the illustration of the idea.

7.2. Parallel Execution of Queries

As SimpleDB is optimised to handle parallel queries, we have utilised this feature by trying to execute the keyword query to multiple queries so we can retrieve the result set in parallel and merge them at application level.

The query in last section has been broken down into the following four queries:

*SELECT * FROM EDT-EDTrecord WHERE 'discipline'=:discipline AND 'abstract' LIKE '%:keyword%';*
*SELECT * FROM EDT-EDTrecord WHERE 'discipline'=:discipline AND 'title' LIKE '%:keyword%';*
*SELECT * FROM EDT-EDTrecord WHERE 'discipline'=:discipline AND 'author' LIKE '%:keyword%';*
*SELECT * FROM EDT-EDTrecord WHERE 'discipline'=:discipline AND 'keywords' LIKE '%:keyword%';*

The above four queries can be executed in parallel and the results are merged to get the queried document.

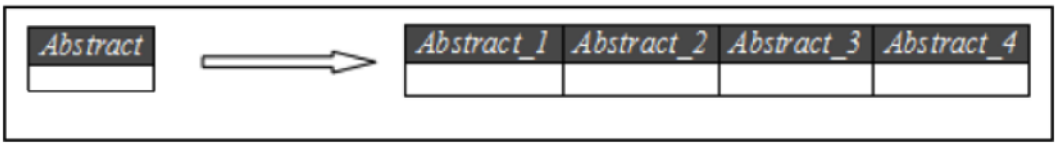


Figure 12. The workaround solution to represent one single attribute in abstract field as multiple attributes.

8. CONCLUSIONS

This paper has critically analysed the hosting of a typical ECM application as in cloud computing environment and evaluated the different Amazon solution architectures for deploying ECM applications and take advantages offered by the cloud services as well as the barriers may arise.
Some of the key findings of the case study are: firstly, the importance of cloud computing and those characteristics of scalability and availability which will be given to those applications deployed and consuming the cloud services. Its characteristics will add value to the application deployed will solve one of the main problem facing ECM solutions implementations like the capacity planning and expansion; secondly, the report investigated the current offering by Amazon cloud services.

Based on selecting electronic dissertation and thesis repository application to be considered as a case study for enterprise content management application in the cloud computing, as well as a deep analysis for this application components architecture and all available deployment architecture scenarios in both traditional non-cloud and cloud based environment. It was discovered that Amazon web services provides more than one solution to deploy your application in, one of solution pro- vides fully automated scalability and fault tolerance solution without administration intervention while other one provides scalability and availability with a need for a little intervention by the an administrator.

In order find out the challenges that might be faced during the development and deployment to cloud environment a real prototype implementation has been done. Most of the challenges occurred during the development of simpleDB access layer have been solved with a workaround solutions at the application level as well as a best practice multi threading technique used to execute complex database queries to utilize the parallel mechanism supported by simpleDB.
Although it was successful implementation for basic functionalities of ECM using simpleDB as database storage, some improvements are needed in order to develop complex ECM applications. Future work would consider some advanced features like reports generation tool and workflow

feature which needs concurrency handling and ACID transaction support which are not yet supported by simpleDB service. However, by using Amazon relational database service is expected to be a suitable for advanced and complex ECM applications.

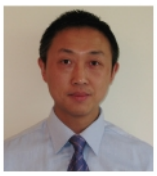
ACKNOWLEDGEMENTS

The authors would like to thank all staff in the department of computing at the University of Northampton for the support and advices. We also thank the anonymous reviewers for the useful comments, which have helped improve the paper.

REFERENCES

[1] Alfresco. Alfresco homepage. In http://www.alfresco.com/, last accessed 2012.
[2] Amazon. Amazon EC2 Storage Options Quick Reference. In http://docs.amazonwebservices. com/AWSEC2/latest/UserGuide/StorageOptions.html, Aug. 2011.
[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Knowinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A View of Cloud Computing. Communication of the ACM, 53(4):50–58, 2010.
[4] Association for Information and Image Management. What is enterprise content management? In http://www.aiim.org/ What-is-ECM-Enterprise-Content-Management, Aug 2011.
[5] J. Becking, S. Course, G. V. Enk, J. Lahaye, and e. a. D. Ockeloen. MMBase: An open-source content management system. IBM System Journal, 44(2), 2005.
[6] H. Collins. Enterprise Knowledge Portals: Next Generation Portal Solutions for Dynamic Information Access, Better Decision Marketing, and Maximum Results. Am. Management Assoc., 2003.
[7] R. Buyya et al. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems, 2009.
[8] N. Fallenbeck. et al. Xen and the Art of Cluster Scheduling. In The 1st International Workshop on Virtualisation Technology in Distributed Computing (VTDC2006), 2006.
[9] P. Barham. et al. Xen and the art of virtualisation. In Ottawa Linux Symposium, 2004.
[10] B. Rochwerger. et al. The Reservoir Model and Architecture for Open Federated Cloud Computing. IBM Journal of Research and Development, 2009.
[11] Hibernate home page. Relational Persistence for Java and .NET. In http://www.hibernate.org/, 2012.
[12] IceFaces. Homepage. Icefaces - the leading JSF framework. In http://www.icesoft.org/ projects/ ICEfaces/overview.jsf, 2012.
[13] RightScale. Homepage. Rightscale cloud management. In http://www.rightscale.com/, 2012.
[14] J. Metzler and A. M. et al. Virtualisation, Cloud Computing and Optimisation and Securing the Internet. The 2011 Application and Service Delivery Handbook, 2011.
[15] S. Nadgowda. Cloud performance benchmark series. In Cloud Computing Centre - NSAC, 2011.
[16] S.Nordheimand, T.Paivarinta. Implementing enterprise content management: from evolution through strategy to contradictions out-of-the-box. European Journal of Information Systems, 15:648–662, 2006.
[17] H. Smith and J. McKeen. Developments in Practice VIII: Enterprise Content Management. Comm. Assoc. of Information Systems, 11(33): 647– 659, 2003.
[18] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Internet Computing, 2009.

Authors



Dr. James Xue is a lecturer at the University of Northampton. He graduated with a first class honours in Computing and Information Systems in 2004 from the University of Nottingham, and obtained his MSc and Ph.D. in Computer Science in 2006 and 2009 from the University of Warwick, respectively. Dr. Xue had more than eight years industrial experience (both in China and Singapore) before his higher education in the UK. Upon the completion of his Ph.D., Dr. Xue worked as a Researcher and Lead Software Developer at the University of Warwick for ten months. Then he worked as a Research Associate at Cardiff University for a short period before he joined the University of Northampton as a lecturer. His research interests include performance modelling and evaluation of multi-tiered Internet services, request scheduling and resource management, virtualisation, cloud computing, distributed systems, etc. He has a good track record of publication in international conferences and journals.

Amjad Yahya is a senior IT professional with more than seven years of experience. Solid and progressive experience in leading, organising, planning and managing enterprise IT implementation and strategies to fulfil the desired aims and objectives. Practical hands-on experience in designing, executing, implementing and delivering projects which are Enterprise Content Management (ECM) solution focused. Practical experience supported by broad range of scientific knowledge aggregated during the preparation for the undergraduate and postgraduate degrees; BSc. Computer engineering and MSc. Internet Security. Amjad was employed by well known IT service providers such as Saudi Business Mach ine (SBM) and one of the big consultation service providers such as PricewaterhouseCooper (PwC) to develop a success stories with giant clients in Middle East region in different sectors such as governmental, banking and healthcare.