Journal of Computing and Security

July 2015, Volume 2, Number 3 (pp. 223–231)

http://www.jcomsec.org

A New Scheme for Storage and Retrieval of Shared Text Files

Hussein Shirvani [a,*], Hamed Vahdat-Nejad [a]

[a] *Pervasive and Cloud Computing Laboratory (PerLab), University of Birjand, Birjand, Iran.*

ARTICLE INFO.

Article history:
Received: 27 May 2015
Revised: 08 March 2016
Accepted: 04 June 2016
Published Online:

Keywords:
Cloud Computing, Big Data, Storage Service

ABSTRACT

Recently, there has been an increasing need for storage spaces in the cloud centers. Efficient techniques for storing files can slow down this trend. In this paper, we investigate on the problem of storing replicated text files that have been changed by different users, individually. Each user could annotate their document by providing shapes and artworks, each of which may include text. The proposed scheme is based on storing the changes made by each user in a separate file for that user. Since the changes are applied to a minor part regarding the whole document text, storing them instead of the whole document could decreases the whole needed volume, significantly. Experimental results demonstrate the efficiency of the proposed scheme in terms of reducing the utilized storage space. Particularly, the proposed scheme is more efficient when the number of users or file size increases.

1 Introduction

Nowadays, big data and cloud computing have gained much popularity. On the one hand, various sources continuously generate a huge amount of data, and on the other hand, cloud storage is utilized as an enormous memory resource for storing data. However, the speed of generating data exceeds the speed of developing cloud infrastructure. As a prediction, the volume of data production in 2020 will be 44 times greater than whatever it was in 2009 [1]. Cloud Computing provides hosting and delivering different services over the Internet [2][3][4]. Clusters of data centers in a cloud allow customers to store, manipulate and process huge amount of data in a short time. Cloud Computing is interesting for both enterprises and individuals, because it allows them to get their services based on their demands.

Traditionally, several categories for cloud services have been proposed including infrastructure, platform, and software. Recently, data as a service [5] has been introduced as one of the major services provided by cloud. While the number of cloud users increases every day, each user may attempt to store a larger amount of data than before. This leads to an increasing demand for cloud storage capacities. As a result, proposing efficient approaches for storing users' data and decreasing the required space becomes important. Text documents and books are major documents that are stored in the cloud by millions of users around the world. Moreover, each text document may be edited by millions of users, separately. In general, users intend to customize them by highlighting important parts. Moreover, they usually want to annotate some pages by providing some text inside a box or shape to be distinguished from original text lines. If we store all customized files for numerous number of users, a huge amount of space will be needed.

In this paper we propose a new scheme that tries to efficiently store users' customized text files. It is implemented for common text file formats which are

* Corresponding author.
Email addresses: hussein.shirvani.1992@ieee.org (H. Shirvani), vahdatnejad@birjand.ac.ir (H. Vahdat-Nejad)
ISSN: 2322-4460

used in books, documents, etc. The approach is based on storing the original files which is shared among many users only once, but an intermediate file for each user, which contains editing information including italic, strikethrough, highlight, bold, and underline as well as shapes and artwork data provided by the user. Part of these features have been investigated in our previous work [6]. In this paper, the idea is extended by considering more features and in particular, shapes. In addition, the reverse procedure for building user's customized file is proposed and implemented. The experimental results show an excellent efficiency in decreasing the total volume of stored files of users.

The rest of the paper is organized as follows: Section 2 presents the related works. Section 3 describes the proposed scheme in great details. Section 4, provides the evaluation results, and finally Section 5 concludes the paper.

2 Related Works

So far, much research has been performed on controlling the amount of required space in a cloud data center. Besides, new platforms like Apache Hadoop [7], which is a software framework, are designed for distributed storage. It is designed to handle processing of very large data sets in a distributed manner on clusters of computers. Hadoop consists of two main parts. Storage part, which is called Hadoop Distributed File System (HDFS) and processing part, which is called MapReduce [8]. Although, Hadoop has many advantages in storing and processing large files, it has some shortcomings for small files. The following related works investigate this problem separately.

As data volume increases, the I/O pressure on the underlying storage service become higher [9]. A difficult challenge that the storage service must deal with, is to reach higher I/O throughput in spite of heavy concurrent massive data access which utilizes high bandwidth. This paper aims at minimizing two parameters: the storage space and bandwidth utilization by applying data compression to conserve these parameters at low computational overhead and reach minimal impact on I/O throughput when volume of data access is high.

Prithvika et al. [10] introduce Cloud Computing as a technology that can help people to share resources. It provides appropriate power of computing and reasonable cost of services as well as high scalability and performance. They claim that in some conditions, it can be cost-effective to keep intermediate datasets that are produced while processing data. Besides, any entities interested in a specific resource, may share the stored information. In the case of sharing resources, security problems arise as well. This paper investigates how to store intermediate datasets in a secure and cost-effective way.

Srinivasa Rao et al. [11] identify the problem of choosing appropriate storage service in cloud as a major challenge. They declare that some methods are designed in a way that work best for a specific kind of data. Some of the services are proposed and implemented to match small amount of data blocks while others are suitable for large blocks. Of course, the cloud storage service provider manages the storage policies. This paper proposes a new algorithm to select the best storage service which is efficient in terms of cost and storage.

Zhu et al. [12] investigate the problem of storing biomedical data in cloud which its amount is growing explosively with the development of biomedical methods. Unfortunately, the dimension and computation level of this type of data is high. However, using cloud computing in resource allocation, data storage, etc. helps them to solve huge biomedical data problems. They propose a new technique to compress and store biological and medicine data sets in high quality in cloud computing. Also, by using decompression method, actual data will be reconstructed with high accuracy.

Manzoor et al. [13] discuss about deduplication of data as a method for controlling redundancy in the cloud by storing only one physical copy. In this method, the data uploaded by each user is compared to the history of data which is already existed in the cloud. In order to ensure data confidentiality and protection, a new encryption technique has been designed and the encrypted data will be saved in the cloud.

Dong et al. [14] investigate the problem of working with small files on HDFS. In their context, working is defined as accessing and storing. Their approach reduces the overhead of NameNode and enhances the throughput of retrieving small files. All files which belong to a PPT courseware are merged in order to enhance small files storing efficiency. Also, a local indexing mechanism for accessing small files in merged files is utilized. These indices for each original file are based on the related offset and length. Thereafter, a two-level pre-fetching mechanism is utilized to improve accessing small files.

Alatorre et al. [15] mention the importance of storage services as a major part of organization's IT infrastructure services. Therefore, different service management methods are used in order to solve the storage resource usage optimization problem while considering requirements related to high availability, performance, etc. A lot of storage service management optimiza-

tions use virtualization as their basis, which results in reduction of various costs.

In another paper, Dong et al. [16] points out the problem of small files on HDFS and propose an optimized approach for improving storage and accessing small files. In this approach, all files that are related structurally are considered for merging and prefetching. Also, all files that are related logically are considered for grouping and prefetching.

Jiang et al. [17], reviewed the HDFS inability in storing small files and introduce an optimization method for small file storage which is based on HDFS I/O. Their approach allows saving many small files into one block that increases the MapReduce task speed for better overall performance. It also uses some meta-data about these files to optimize the speed of querying small files.

Zhang et al. [18] propose a new way for storing small files. It acts as an HDFS-independent engine and reduces the HDFS overhead and efficiently retrieve and fetch the small files. It takes advantage of using a special type of I/O in building the server. Also, reading and merging of small files is performed using non-blocking input/output. Moreover, the server uses small files caching mechanism to read and access them, efficiently.

Mackey et al. [19] investigate storing small files on HDFS, and propose an approach which improves space usage for meta-data. It is noted that small files can impact the performance of NameNode, *i.e.* Hadoop meta-data server, significantly. For both the space and the number of files that are in the file system, it is assumed that a quota is devoted. They take the advantage of Hadoop "harballing", which is an archiving method [7] aiming to aggregate small files into one large file, which enables accessing the original files in parallel.

Terzo et al. [20] propose a Data as a Service (DaaS) approach for processing and sharing large data collections, intelligently with the help of abstracting the location of data and completely disengaging the data and its processing. This approach offers DaaS to support large groups of users who require processing, accessing and sharing the data in order to generate knowledge from them, collectively.

Balasubramanian et al. [21] declare that cloud-based storage services must be access-efficient to reduce reading/writing files across the network in the system, and space-efficient to be able to manage the high volumes of data. They introduce a new approach with mentioned features for eliminating copies of a set of files with certain degree of similarity across a set of distributed servers. It results in minimizing the total space required for storing and deduplicating the files. Their approach is access-efficient with polynomial time complexity and controlled space overhead.

Al Nuaimi et al. [22] try to optimize the redundancy of data on the cloud in order to enhance the processing, which can result in a reduction in storage costs for the cloud service providers. They work on better utilization of storage space required on the cloud server by using a mechanism called data partitioning [23][24].

Google Docs [25] is a service provided by Google. It creates text, spreadsheet document, and slides. Moreover, it supports ".docx", ".doc", and ".pdf" for text, ".xls" and ".xlsx" for spreadsheet documents, and finally ".ppt" and ".pptx" for slides. Google Docs allows users to share their documents with other people and to collaborate with them, of course with specific permissions. When a user shares a document with other users, they can edit, annotate, and add formatting data such as bolding, highlighting, changing text color, etc. to the document.

In this paper, a cloud-based efficient storage service is proposed, which lets all users to modify their specific version of a text document. The scheme is different from all previous works and is applicable to both small and large files.

3 Proposed Scheme

The following scenario helps in understanding the proposed approach:

A philosophic instructor in the Department of Humanities at the University of Birjand gave a philosophical document to be reviewed by its students. Each student adds formatting data (highlight, underline, etc.) and shapes to major parts of the document reflecting his/her idea about each part. This document is uploaded to each student's cloud account in order to be accessed anywhere and also, each student can open, read and save the document using existing platform, which is provided by this university. The size of this document is 15.18 MB. There are 50 students in this class. Therefore, the total size for saving the document for each student in his/her account will be 759 MB. If this assignment is going to be given to 3 other classes with the same number of students, about 3 TB is needed, which is a huge amount respect to the total cloud storage that is installed for the university. Therefore, the cloud administrator seeks efficient approaches for decreasing the storage demands.

The proposed scheme is a solution to the mentioned problem and is regarded as an extension to our previous work [6]. It aims to store the main document only once and separately maintain the customized changes

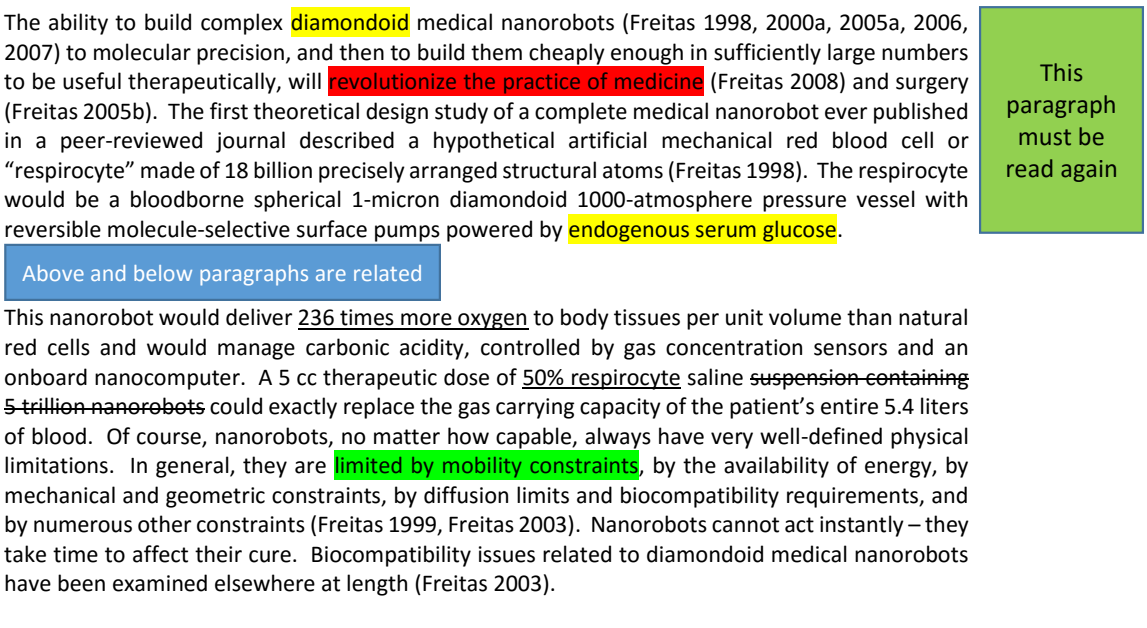

Figure 1. Example of a Shape Inside a Document

for each user. These changes are stored in an intermediate file for each user. In the following, we provide the details of the scheme.

Each word in a text file has some attributes. Among them, we have investigated the highlight color, underline, strikethrough, italic, bold, and position in the document as the main attributes that may be changed by users. When a user opens the shared file for the first time, it is fresh and clear from all formatting data. They consider the file and may highlight some statements or perform other changes (underline, Italic, etc.). When exiting the Word processing application and asking for save, the proposed scheme begins. It checks all of the mentioned attributes for all words of the document. Because the nature of these attributes are different, we systematically describe the operations. Each attribute has a specific number, which makes it distinct, and is used as its identification number (ID). One of the main attributes that has been investigated is shapes. They could convey text and annotations. We treat shapes' information different from other attributes. Figure 1 illustrates a shape inside a document.

After closing the text processing application, the server checks the document for any of these attributes. Upon receiving a signal, it extracts the position of the word or set of words, which consists of the start and end point, and the type of attribute that is applied. When the server reaches the end of the document, a list of words' position (start and end points) with the type of attributes are in our hand. It saves the list in a Spreadsheet document. At the end, the server saves this Spreadsheet document at the user account for later use and access.

If the user inserts some shapes like a circle, an oval or even a text box in the document, all shapes' information is saved in a text file. This file contains the shape type, position and the anchor. We prefer to use text files to save shapes' data for better management of produced intermediate files. The graphical pseudo-code of the extraction algorithm is shown in Figure 2.

As Figure 2 illustrates, after opening the document, two lists for words (wordList) and shapes (shapeList) are created. If a word is detected, and it was formatted, the related type (highlight, underline, etc.), start position and end position for that word is inserted to the wordList. If a shape is detected, its type, related page number, its position from left and top of the page, the width and height of the shape and the anchor is inserted to the shapeList. The scheme investigates the user document word by word for any changed attribute. Therefore, it stores wordList in a spreadsheet file. For detected shapes, the shapeList is saved in a text file.

When a user opens their exclusive document, the scheme attaches these two intermediate files to the original document in order to construct the user's customized text file. For this, the server reads the shape file record by record and applies each record information to the related part of the original file. Based on the saved attributes, a new shape with the same type, position and dimension is created and the related anchor is applied (if exists). Then, the server opens the saved spreadsheet document and applies it element by element. The graphical pseudo-code of the attach algorithm is shown in Figure 3.

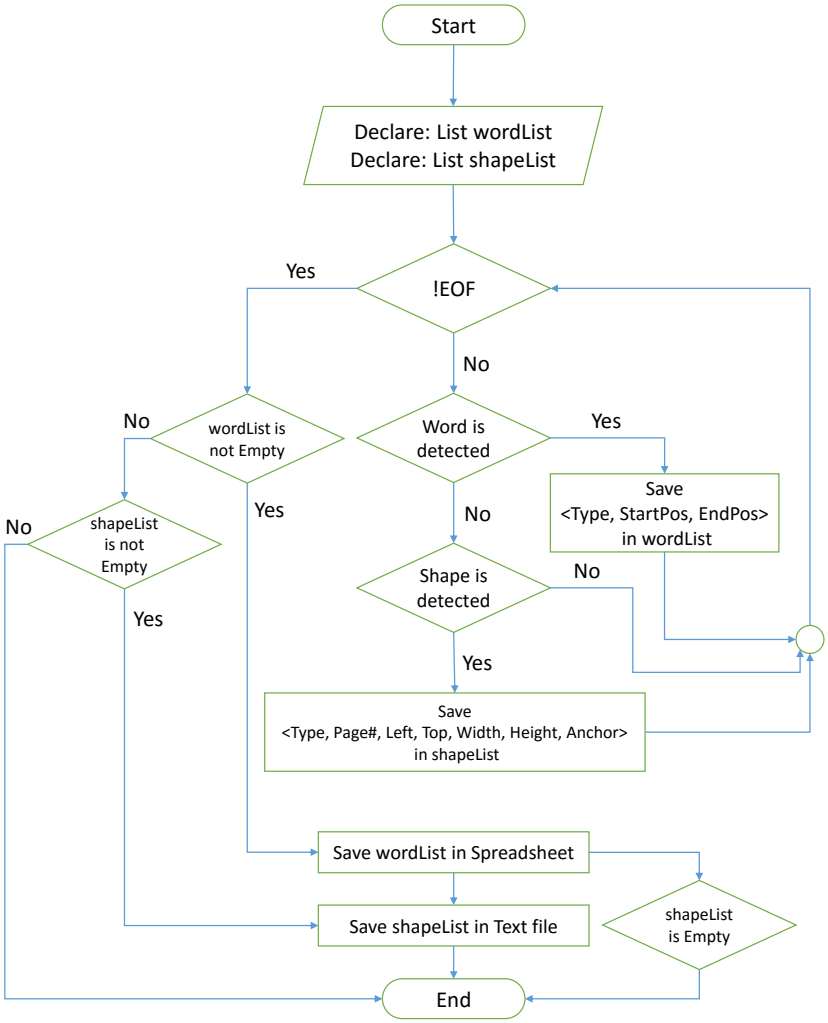

Figure 2. Flowchart of the Extract Algorithm

Every time the user closes the word processing application, the server compares the open document with the original one. If they are the same, it means that the document is unchanged. Otherwise, it performs the proposed scheme. The proposed scheme can be implemented for any word processing application.

Figure 4 shows the functionality of the proposed scheme in the cloud infrastructure. The original file is saved in the cloud server and each user can access and edit it. After editing, the Extractor detects the changes applied by the user to the document and saves the formatting and shapes information as explained above. On the other side, the cloud retrieves the original file and the formatting and shapes information from the user account and gives both of them to Attacher. The Attacher applies those changes to the original document and sends it to the user.

4 Evaluation

In this section, we discuss the implementation details, and then show the evaluation results of the proposed scheme. ".doc" and ".docx" formats as well as ".pdf", *i.e.* Portable Document Format, are considered as our experimental text documents. In order to maintain our intermediate files information, Microsoft Excel document and ordinary text files are utilized. Moreover, we use C# programming language to implement the proposed scheme, because it has valuable libraries produced by Microsoft that give access to every Microsoft Office document one wish to manipulate.

Our system consists of two major services: Extract and Attach. Extract service is designed to capture all formatting data as well as all shapes data from a text document and to store this data into a Microsoft Excel document and an ordinary text file, respectively. On the other hand, the Attach service gets these intermediate files as input and attaches them to the original document to generate user's own text document. The

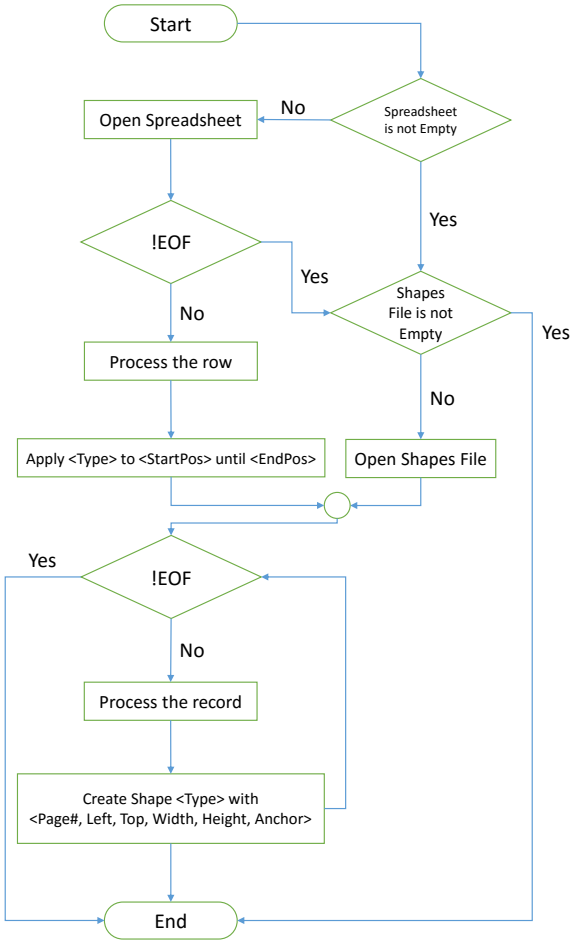

Figure 3. Flowchart of the Attach Algorithm

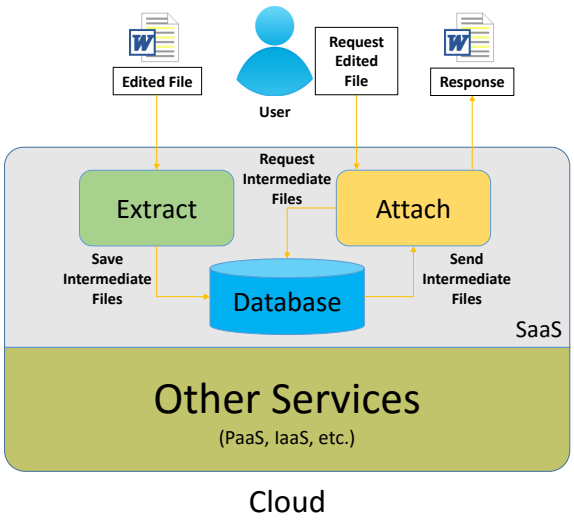

Figure 4. System Architecture

service finishes in appropriate time in a fraction of a second.

We test the proposed system by annotating the key points of the document and averaging the results acquired from all users. The algorithm is evaluated for two files with different sizes of 3756 KB (3.668 MB) and 8046 KB (7.857 MB). The mean size of files after being edited by 100 users is 6383 KB (6.233 MB) and 8823 KB (8.616 MB), respectively. In the traditional approach, each customized document is stored separately, therefore, the total size for 100 users becomes roughly 100 times of the size of the original file.

Table 1 shows the results of applying the traditional approach as well as the proposed scheme. It can be seen that there is a big difference between the occupied space by these two approaches. As it can be seen, the formatting part and shapes can be stored in negligible files comparing to the size of the original document. Therefore, storing them for many users can significantly decrease the volume of the data. In continue, we investigate our system procedure based on two major criteria including efficiency and execution time.

4.1 Efficiency

In order to compute the efficiency of the proposed scheme, we make use of the following formula:

$$Efficiency = 1 - (PSResult/TAResult) \quad (1)$$

where PS stands for Proposed Scheme and TA stands for Traditional Approach. Figure 4 shows the efficiency of the proposed scheme with respect to the traditional approach in Average-case. It expresses how efficiently this scheme can store data. The results demonstrate a high efficiency for both files.

4.2 Execution Time

We test our system on a server running Windows Server 2012 with Quad-core 2.8GHz CPU and 4GB RAM. Execution time is divided into two different parts: Extract time and Attach time. In Extract time, we mean the required time to extract all formatting and shapes data from the target document. In our experiment, the Extract time, is recorded as shown in Table 2. The time is not small; However, it could be executed much faster on a high speed cloud server. Because, cloud servers use virtualization technology which gives access to pool of resources that work in parallel. Beside the extract service could be performed in offline mode and time is not that matter. On the other hand, the attach service should be executed in online mode. As the table shows, the Attach time is small and Attach execution is completed in seconds in our experiment. Therefore, it could be executed in real-time, when we ran it on a high-speed cloud.

Finally, it should be noted that the execution time is related to the length of the document and as the document becomes larger, it will have impact on Extract

		File#1	File#2
	Original file Size	3756 KB	8046 KB
	Edited file Size	6383 KB	8823 KB
Traditional Approach	**Total file size for 100 users**	638300 KB	882300 KB
Proposed Scheme	**Formatted parts + Shapes**	37 KB	541 KB
	Total file size for 100 users	7456 KB	62146 KB

Table 1. Total Required Volume for the Traditional Approach Respect to the Proposed Scheme

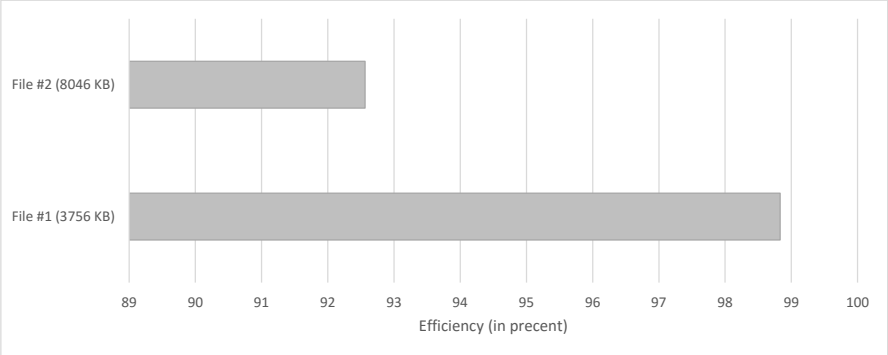

Figure 5. Efficiency of the Traditional Approach and the Proposed Scheme

Time	File#1	File#2
Extract	15 mins	18 mins
Attach	4 secs	5 secs

Table 2. Execution Time of the Proposed Scheme

time. Although, Attach time remains small because we have all positions and related formatting data as well as shapes' information.

5 Conclusion

Rapid migration of commercial, scientific, and industrial data to cloud servers leads to a need for huge data storage. In this paper, a new scheme for storing and retrieving text files that are shared by many people is proposed. Each user could customize its own file by annotating the document via inserting comments inside shapes or arts. The architecture of the proposed scheme consists of two main components extract and attach. The extract component is responsible for extracting changes made by the user especially the annotations provided via shapes and artworks. The attach component is responsible for building user's customized document by applying the extracted files to the original document. Experimental results have shown a huge save in disk space especially for larger files and larger groups of users. However, by increasing the file size, the processing time for extracting the formatting data increases. This approach can be developed as an API in order to be accessed easier. Moreover, it could be extended by adding more functions, which can gather other features that can be inserted into the document without changing its content. Finally, it could be implemented by other programming languages like Java for other file formats.

References

[1] Doug Howe, Maria Costanzo, Petra Fey, Takashi Gojobori, Linda Hannick, Winston Hide, David P Hill, Renate Kania, Mary Schaeffer, Susan St Pierre, et al. Big data: The future of biocuration. *Nature*, 455(7209):47–50, 2008.

[2] Mehdi Bahrami and Mukesh Singhal. The role of cloud computing architecture in big data. In *Information Granularity, Big Data, and Computational Intelligence*, pages 275–295. Springer, 2015.

[3] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[4] Wikipedia. Cloud computing — wikipedia, the free encyclopedia, 2015. URL `https://en.wikipedia.org/w/index.php?title=Cloud_computing&oldid=723983411`. [Online; accessed 13-February-2015].

[5] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. Big data and cloud computing: current state and future opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 530–533. ACM, 2011.

[6] Hussein Shirvani and Hamed Vahdat-Nejad. A new efficient approach to store data in a cloud server. In *Proceedings of the 3rd International Conference on Context-Aware Systems and Applications*, pages 179–184. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[7] Apache Software Foundation. Welcome to apache (tm) hadoop, 2014. URL `http://hadoop.apache.org/`. [Online; accessed 18-December-2014].

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[9] Bogdan Nicolae. High throughput data-compression for cloud storage. In *International Conference on Data Management in Grid and P2P Systems*, pages 1–12. Springer, 2010.

[10] Sarah PC Prithvika, Kalai S Arasi, Angel A Princes, and S Ramani. Cost effective management of intermediate datasets in cloud environment. *Australian Journal of Basic and Applied Sciences*, 10(1):232–237, 2016.

[11] L Srinivasa Rao and I Raviprakash Reddy. Study and comparison of non-traditional cloud storage services for high load text data. In *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 2*, pages 299–311. Springer, 2016.

[12] Changming Zhu. Data-compression-based resource management in cloud computing for biology and medicine. *Journal of Computing Science and Engineering*, 10(1):21–31, 2016.

[13] Shyma Manzoor, Amrutha Chandran, Raji Jayan, RS Archana, Sowmya KS, et al. A single instance storage: An efficient space optimization in cloud storage. *Imperial Journal of Interdisciplinary Research*, 2(4), 2016.

[14] Bo Dong, Jie Qiu, Qinghua Zheng, Xiao Zhong, Jingwei Li, and Ying Li. A novel approach to improving the efficiency of storing and accessing small files on hadoop: a case study by powerpoint files. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 65–72. IEEE, 2010.

[15] Gabriel Alatorre, Sandeep Gopisetty, Divyesh Jadav, Bryan Langston, Nagapramod Mandagere, Ramani Routray, and Heiko Ludwig. *Optimizing Cloud Storage Management Services*, pages 404–429. IGI Global, Hershey, PA, USA, 2015. ISBN 9781466684966. URL `http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-8496-6.ch014`.

[16] Bo Dong, Qinghua Zheng, Feng Tian, Kuo-Ming Chao, Rui Ma, and Rachid Anane. An optimized approach for storing and accessing small files on cloud storage. *Journal of Network and Computer Applications*, 35(6):1847–1862, 2012.

[17] Liu Jiang, Bing Li, and Meina Song. The optimization of hdfs based on small files. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 912–915. IEEE, 2010.

[18] Yang Zhang and Dan Liu. Improving the efficiency of storing for small files in hdfs. In *Computer Science & Service System (CSSS), 2012 International Conference on*, pages 2239–2242. IEEE, 2012.

[19] Grant Mackey, Saba Sehrish, and Jun Wang. Improving metadata management for small files in hdfs. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–4. IEEE, 2009.

[20] Olivier Terzo, Pietro Ruiu, Enrico Bucci, and Fatos Xhafa. Data as a service (daas) for sharing and processing of large data collections in the cloud. In *Complex, Intelligent, and Software Intensive Systems (CISIS), 2013 Seventh International Conference on*, pages 475–480. IEEE, 2013.

[21] Balamurugan Balasubramanian, Tian Lan, and Mung Chiang. Sap: Similarity-aware partitioning for efficient cloud storage. In *INFOCOM, 2014 Proceedings IEEE*, pages 592–600. IEEE, 2014.

[22] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi, and Jameela Al-Jaroodi. A self-optimized storage for distributed data as a service. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2015 IEEE 24th International Conference on*, pages 84–89. IEEE, 2015.

[23] Jameela Al-Jaroodi and Nader Mohamed. Ddftp: dual-direction ftp. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 504–513. IEEE Computer Society, 2011.

[24] Nader Mohamed, Jameela Al-Jaroodi, and Abdulla Eid. A dual-direction technique for fast file downloads with dynamic load balancing in the cloud. *Journal of Network and Computer Applications*, 36(4):1116–1130, 2013.

[25] Wikipedia. Google docs, sheets, and slides — wikipedia, the free encyclopedia, 2015. URL `https://en.wikipedia.org/w/index.php?title=Google_Docs,_Sheets,_and_Slides&oldid=723429279`. [Online; accessed 5-January-2015].

Hussein Shirvani received his B.Sc. degree from University of Birjand, Birjand, Iran in 2014. Currently, he is a M.Sc. Student of Computer Networks at the Department of Information Technology of University of Isfahan, Isfahan, Iran. Also he is an IEEE, IEEE Computer Society, and IEEE Cloud Computing Graduate Student Member. His research interests include Cloud Computing, Pervasive Computing and Context-Awareness.

Hamed Vahdat-Nejad is currently an assistant professor at the computer engineering department of the University of Birjand. He received his PhD from computer engineering department of University of Isfahan in 2012, his master degree from Ferdowsi University of Mashhad in 2007, and his bachelor's degree from Sharif University of Technology in 2004. He was a research scholar at the Middleware laboratory of Sapienza University of Rome in 2011. Currently, his research is focused on cloud computing, pervasive computing and context-awareness. He has (co)published about 30 papers in conferences and journals, and leads the Pervasive and Cloud computing Lab at the University of Birjand. He has served as the chairman of the 1st and 2nd International Workshop on Context-aware Middleware for Ubiquitous Computing Environments, as well as the "3rd and 4th International workshop on Pervasive and Context-aware middleware". He has served as TPC member for ICCKE, IWCMC, ISIEA, ICCIT-WCS, PerCAM, ChinaCOM, MELECON2014, COGNITIVE-2014, IBMSGS2015, EMERGING 2015, ICACCI, ADMMET'2015 ICCME-2015, CoCoNet'15, AR4MET'2016, REEGETECH'2016, ISTA'16, etc. Currently, he serves as guest editor for Elsevier Computers and electrical engineering.