Efficient Storing Generally-Shared Individually-Customized Text Documents in a Cloud Server

Hussein Shirvani
Pervasive and Cloud Computing Laboratory (PerLab)
Faculty of Electrical and Computer Engineering
University of Birjand, Iran
hussein.shirvani.1992@ieee.org

Hamed Vahdat-Nejad
Pervasive and Cloud Computing Laboratory (PerLab)
Faculty of Electrical and Computer Engineering
University of Birjand, Iran
vahdatnejad@birjand.ac.ir

***Abstract* - The present research aims at introducing a new efficient approach to store shared text documents, which are individually replicated and customized by users, in a cloud server. The approach is applicable to all text file formats supported by any existing application. We assume a text document consists of both text and images. Instead of replicating text of the original document, two tiny intermediate files for each user are generated and stored. Formatting data and comments are produced and saved in a spreadsheet document and a new file format ".comment" respectively. The first file is a spreadsheet document that contains metadata about customized information including highlighted and underlined (and other formatting styles like strikethrough, bold and italic) parts of the document. The other file aims to store comments provided by user in a ".comment" file format. Furthermore, the approach addresses efficiently storage of annotated images inside the document. For this, an algorithm is proposed to store the difference between the annotated and the original images. Whenever a user wants to access his/her customized document, the intermediate files are attached to the original document and appeared to him/her. Finally, we evaluate the proposed approach through a real scenario. The experimental results show a large amount of disk space are saved.**

***Keywords*—Cloud computing, DaaS, Storage**

I. INTRODUCTION

Data plays a significant role in our daily life. It is generated from different sources like people, devices, sensors, etc. Massive amount of data from these sources, forms "Big Data" [1]. "Big data refers to datasets whose size is beyond the ability of typical database software tools and applications to capture, store, manage and analyze" [2]. Therefore, we are dealing with a major issue, which impacts all disciplines of science. As a prediction, in 2020, the volume of data production will be 44 times greater than whatever it was in 2009 [3]. Several approaches have been proposed to deal with big data issues in order to reach a relative reasonable solution. They encompass diverse fields including systems of education [4], life science [5, 6], material science [7], social networks [8, 9], and the like. The massive amount of data requires to be analyzed to produce less-amount valuable data.

Cloud Computing technology provides new opportunities to collect and process big data. Cloud Computing is a new paradigm for hosting and delivering different services over a network or the Internet [1, 10]. It hosts clusters of datacenters to allow customers to store and process huge amount of data. This technology is interesting for enterprises, because it allows them to start their business from small resources (e.g., CPU, memory and storage) and increase them whenever the service demand rises. This enables the customers to pay for the storage and computation resources only based on their required capacity.

Data as a service [40] has been recently introduced as one of the major services provided by cloud. Every day, the number of cloud users increases and each user needs to store a large amount of data. This results in an increasing need for the cloud storage service. It prompts the requirement of using efficient approaches to store data in cloud.

Developing efficient methods for storing data in the cloud can slow down the movement to install more storage resources. Text files and books constitute part of the data stored in the cloud. They include both text and images. This kind of file is generally annotated by users and also some formatting data such as highlight, underline are added by them. In a very common situation, a specific text file is shared and used by a large number of users. The following scenario expresses this situation:

The department of Computer Engineering at the University of Birjand, has opened a new course in the field of Cloud Computing for undergraduate and graduate students. In the previous semester 100 students enrolled in this course. Based on this motivation, the department decided to open this course in this semester virtually for other local universities, as well. At the end of enrollment process, the system shows that about one thousand students have enrolled in this course. The enrollment system gives each user a new account from university's private cloud in order to access his/her documents during the course. The course instructor has uploaded the course materials to the cloud which consists of a book, a lecture, and a few slide documents, totally 15.18MB. For two reasons, students are not allowed to download the materials on their computers: (a) the copyright of the materials prevents this, and (b) to be accessible anywhere and on any device by students. Because each student wants to customize its files (e.g. highlight, insert comments, etc.), the whole materials are mirrored on each student's account. As a result, the university's cloud is occupied about 15TB for this course. Upon this successful experience, Computer Engineering and other departments of the University gain the motivation to introduce virtual courses for other

universities students; however, they face the problem of increasing storage requirements in the university cloud.

The current approach to store all the customized files, which are dedicated to different users, results in a huge mass of data storage. The present research proposes a new approach to storing generally-shared, individually customized text data in a cloud server. The approach is implemented for text file formats which includes books, documents, etc. A text file usually contains images as well. The approach is based on storing the original shared files only once in the cloud, but an intermediate file for each user, which contains edited information including highlight, underline, strikethrough, bold, and italic as well as comments provided by user. When a user wants to access his/her customized document, the intermediate files are applied to the original documents and appeared to him/her. On this basis, according to experimental results, a large amount of space is saved.

The rest of the paper is organized as follows: In section 2, we review the past related works on this subject. In section 3, the proposed approach (PA) is explained and discussed in details including system architecture and the implemented application. In section 4, the evaluation results are presented, and finally section 6 concludes the paper.

II. RELATED WORKS

Up to now, several approaches and applications have been proposed to control and reduce the amount of space that data occupies in a cloud datacenter. Moreover, new platforms have been developed for this purpose such as Hadoop [16] and MapReduce [17].Although these efforts result to an improvement in data storage, the problem still exists. In the following, we review the major related works in this area.

Bonvin et al. [18] emphasizes that "as data scales up, its availability becomes more complex." They work on controlling resource allocation in data cloud for several applications that demand different availability levels.

Dong et al. [19] consider the problem of storing and accessing small files on Hadoop distributed file system (HDFS). HDFS is a distributed file system [16] and a representative for Internet service file system [20]. It is designed to support lots of applications running in the Internet as file systems. Their proposed approach uses the correlations among small variations of a Microsoft PowerPoint file courseware, and merges the correlated files into a larger file. In fact, a ".ppt" file is uploaded to a web server and will be converted into multiple picture series. Then, by providing a local index for each original file based on related offset and length, they all together will be merged in a single file and uploaded to HDFS through HDFS client. This scheme reduces the metadata burden. Afterward, a two-level pre-fetching mechanism is utilized to enhance the efficiency of accessing small files.

Jiang et al. [21], consider the disability of HDFS in storing small files and introduce an optimization method based on HDFS I/O for small file storage, which lets one block to save many small files and also some metadata about these files. They define a new metadata structure of small files in the original HDFS in order to optimize the I/O speed of small files. As well as taking the advantage of caching, it has been trying that the files belongs to the same directory are written to the same block which helps increasing the task speed performed by MapReduce. This approach provides a better performance for HDFS.

In another work, Zhang et al. [22] focus on big data storage and confirm the big data storage problem. Specifically, they focus on the exponential growth in the amount of data, which in turn results to a bottleneck in the conventional data storage approaches. They address some major shortcomings of the distributed file system such as small files problem and explain the approach was proposed by Jiang et al. [21] as a relative solution to this problem.

Zhang et al. [23] propose a new approach for storing small files. It works as an engine separately from HDFS and reduces the overhead of HDFS. It uses a special kind of I/O to build the server and utilizes non-blocking I/O to read and merge small files. Moreover, the server caches the small files to perform reading and accessing them, efficiently.

Mackey et al. [24] propose a mechanism for storing small files in HDFS, which makes an improvement in metadata space storage. They assume that each client is allocated a quota for both the space and number of files in the file system. They utilize a special compression method 'harballing', which is provided by Hadoop for better utilization of HDFS.

Sangeetha et al. [25] focus on big data and cloud, discuss about these topics and the way big data are handled in cloud computing environment in their paper. They mention that relational databases have some shortcomings on storing and handling big data which consists of various and heterogeneous data. They review the big data management systems such as Google Bigtable [26], Yahoo! PNUTS [27], Microsoft Dryad [28] and SQL Azure [29], and Amazon Dynamo [30] which are available to handle big data in a cloud. Also, they talk about some existing programming models such as MapReduce [17], HadoopDB [31] and HaLoop [32] which designed to process large amounts of data.

Balasubramanian et al. [33] in his project, SAP, introduce a new approach for deduplicating a set of files which have a certain degree of similarity across a set of distributed servers such that the total space required for deduplicating and storing the files is minimized.

Google Docs [34] is another service provided by Google, which interacts with Google Drive to store users' documents. It creates text, spreadsheet document, and also slides. Moreover, it supports various kinds of well-known formats such as ".docx", ".doc", and ".pdf" for text, ".xls" and ".xlsx" for spreadsheet documents, and finally".ppt" and ".pptx" for slides. Google Docs enables users to share their documents with their friends, colleagues, and other people, with one click. It provides information and knowledge exchange and also users' collaboration. In the case of sharing, users can edit, annotate, change text color, and also add formatting data (bold, underline, and italic) to the document. Also their friends (or someone they share documents with them) can suggest the main user to perform some actions on the document.

In this paper, a cloud-based efficient storage system is proposed, which lets all users to modify their specific version of a text file and image. The approach is applicable to both small and large files.

III. Proposed Approach

To decrease data redundancy, the proposed approach aims at storing the target file including text files and images just the one. For text files, two intermediate files are created, which contain all formatting data and comments that are left by the user. For images, the difference between the original image and the edited one is required which are described later. These intermediate files contain customizations and are created and stored for each user. In continue, the details of the proposed approach are provided in two subsections for text files and images, respectively.

A. Saving Text Files

In the text files such as a Text document, each word has some attributes like, font color, effects, highlight color, underline, strikethrough, italic, bold, and position in the document. The six later attributes are within our focus. With these attributes, the needed data can be captured from the text. When a user opens the shared file for the first time, it is fresh and clear of all formatting data. He/she goes through the file and highlights (underlines and the like) or even adds comments on some parts of the document. Then he/she quits Word processing application and asks the application to save his/her changes. The proposed approach begins at this time and is hidden from the user's view.

Because the attributes of our focus can be applied to a word simultaneously, so all of them must be checked for each selected word. The ways we behave for extracting the needed information from the document for the considered attributes are different. So, in order to understand the whole approach, exploiting distinct categories can be more appropriate. We categorize these attributes into three groups: the first group involves highlight, underline together. Bold, italic, and strikethrough form the second group. Third group consists of comment. The criterion of this categorization is applied by type of the required code that should be written for extracting them. In our approach, each attribute has a specific number which makes it distinct, and is used as its identification number. Table I shows the considered number for each attribute.

TABLE I
NAME OF ATTRIBUTES AND THEIR IDENTIFICATION NUMBER

Type of attribute	Identification number	Example
Highlight	1	Alice
Underline	2	Alice
Strikethrough	3	~~Alice~~
Italic	4	*Alice*
Bold	5	**Alice**

We treat comments' information different from other attributes, because they have different structure. They keep text in themselves, so in addition of saving the position, the related text must be saved.

After closing the application, the server checks the document for any of these attributes. When it receives a signal, it extracts the position of the word or set of words means 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 user account of the cloud for later use and access. Also, if the user annotates some parts in the document, all comments are saved in a file with a new extension called ".comment". ".comment" is an extension for our comment file. This file can be produced by programming languages for the purpose of storing data. This file contains the comment position and the text which has been written as comment. We prefer to use text files to save comment's data for better management of produced intermediate files. The pseudo-code of the extraction algorithm is as shown in Fig. 1.

The proposed approach looks through the document for any applied attribute. When it detects one of first five attributes, it saves its type, start and end point in a list. For comments, it saves the text, and start and end position, and finally the product

1. *For each considered attribute, create a list with three columns, type of the attribute, start position and end position*
2. *Create a list for "comments" with three attributes, related text, start position and end position*
3. *Open the Text document and put the cursor at the beginning of the document*
4. *Process the first, do*
    a. *If one word is highlighted Then save number 1, as well as the start and end position of this word, in the highlight's list*
    b. *If one word is underlined Then save number 2, as well as the start and end position of this word in the underline's list*
    c. *If one word is strikethrough Then save number 3, as well as the start and end position of this word in the strikethrough's list*
    d. *If one word is Italic Then save number 4, as well as the start and end position of this word in the Italic's list*
    e. *If one word is Bold Then save number 5, as well as the start and end position of this word in the Bold's list*
    f. *If a comment is detected Then save its text, as well as the start and end position of the comment's part in the Comment's list*
    g. *Process the next word*
5. *While not EOF, go to 4.a*
6. *Close the Text document without any changes*
7. *Open a new Spreadsheet document*
8. *Save lists in the new Spreadsheet document in three columns, the first column is the type of attribute, the second is the starting point and the third is the end point consecutively*
9. *Save the Spreadsheet document in the user account for later access.*
10. *Close the Spreadsheet document*
11. *Open a new output stream to .comment file*
12. *Save Comment's list in .comment file in the user account for later access*
13. *Close the output stream to .comment file*

Fig. 1 Extract algorithm

list is saved as a new file with ".comment" as its extension, which is much lighter and smaller in size. On the other hand, when a user wants to open his/her exclusive document, the server must attach these two intermediate files to the original in order to present it for him/her. At the beginning of execution of the attach function, the server opens new input stream to the ".comment" file to read it record by record and applies each record information to the related part or parts in the original file. The file is read sequentially, and for each start and end point, the related text is On the other hand, when a user wants to open his/her exclusive document, the server must attach these two intermediate files to the original in order to present it for him/her. At the beginning of execution of the attach function, the server opens new input stream to the ".comment" file to read it record by record and applies each record information to the related part or parts in the original file. The file is read sequentially, and for each start and end point, the related text is applied. After the EOF signals, the server closes input stream to file and opens the saved Spreadsheet document and goes through that. For each start and end point, the specified type of attribute is applied to that word or set of words. The pseudo-code of the attach algorithm is shown in Fig. 2.

1. *Open a new stream to .comment file*
2. *Process the first record*
3. *While not EOF, get the start and end point and the comment's text*
    a. *Apply the text to the word or set of words according to start and end point*
    b. *process the next record and go to 3*
4. *Close the file stream*
5. *Open the Spreadsheet document*
6. *Process the first row*
7. *Apply that relative format type to the word or set of words according to start and end point*
8. *If any valued row exists Then process the next row and go to 9*
9. *Close the Spreadsheet document*

Fig. 2 Attach algorithm

B. Saving Images

Images available in text documents play an important role in increasing the storage occupation. They usually occupy the major storage part of the document. For storing images, we need to find the difference between the edited image and original one. If we store these differences, every time we need to retrieve the image, these differences can be applied to the original image, and then the edited image will be produced and shown to the user. So, it does not require to store the same file twice or more.

To find the difference between the edited file and the original, simply, we subtract the edited file from the original and save it as a picture. To reduce the amount space which can be occupied by this image, we use JPEG compression algorithm. The experiments show that this algorithm can compress images properly.

On the other hand, when one student decide to view his/her document, the difference image(s) will be added to the original image(s) and previewed for the user

The two documents which are resulted from applying the algorithm 1 to text files and the algorithm 2 to images, are hidden from the user view. However, the text file can be shared among users. Therefore, users can exchange their ideas, thoughts, visions and innovations among each other, on the other hand, collaborate with each other.

One of the most important advantages of using this approach is that it gives this ability to use .docx or .doc files instead of .pdf where the size of .pdf file is bigger than .docx or .doc or vice versa. Every time the user closes the word processing application, the server compares the open document with the original one. If they are the same, the document is unchanged and if they are different, the server detects that some changes were performed by the user. If the server gives this permission to the user to keep a new copy of the document for him/herself, the user can save it as a new document but if the writing permission is not granted, an error message will be displayed declaring impossibility of any change. It should be noted that the proposed approach can be implemented using any word processing application, but all of explained concepts are applicable to other file formats with the help of existing libraries for manipulating such a format.

Fig. 3 shows the functionality of proposed approach 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 save the formatting and annotation information as well as difference image in their specific file format explained above in details.

Fig. 4 shows the other side of system architecture. It shows how the system operates when a user requests to access his/her document. If sending request to the cloud, the cloud retrieves the original file and the formatting, annotation information and difference image from the user account and gives both of them to Attacher. The Attacher will apply those changes to the original document and send it to the user.

IV. EVALUATION

In this section, at first we go through the details of implementation, and then evaluate the proposed approach. We use Microsoft Word file formats ".doc" and ".docx" and also Adobe Portable document format ".pdf" as text files. Microsoft Excel document and ".comment" file are utilized as our intermediate and C# programming language is exploited for implementing and running the system. C# has various libraries for manipulating Microsoft Office documents and files. We make use of them for implementing the proposed algorithm. So, the two libraries produced by Microsoft is required for accessing data on this two applications' documents.

Two main components of the system are *Extractor* and *Attacher*. The *Extractor* component is responsible for extracting the formatting data from text document and then save the extracted data in a Microsoft Excel document and a ".comment" file. The *Attacher* component uses the extracted data in Microsoft Excel document and ".comment" file and attaches them to the original document in order to generate user's own text document.

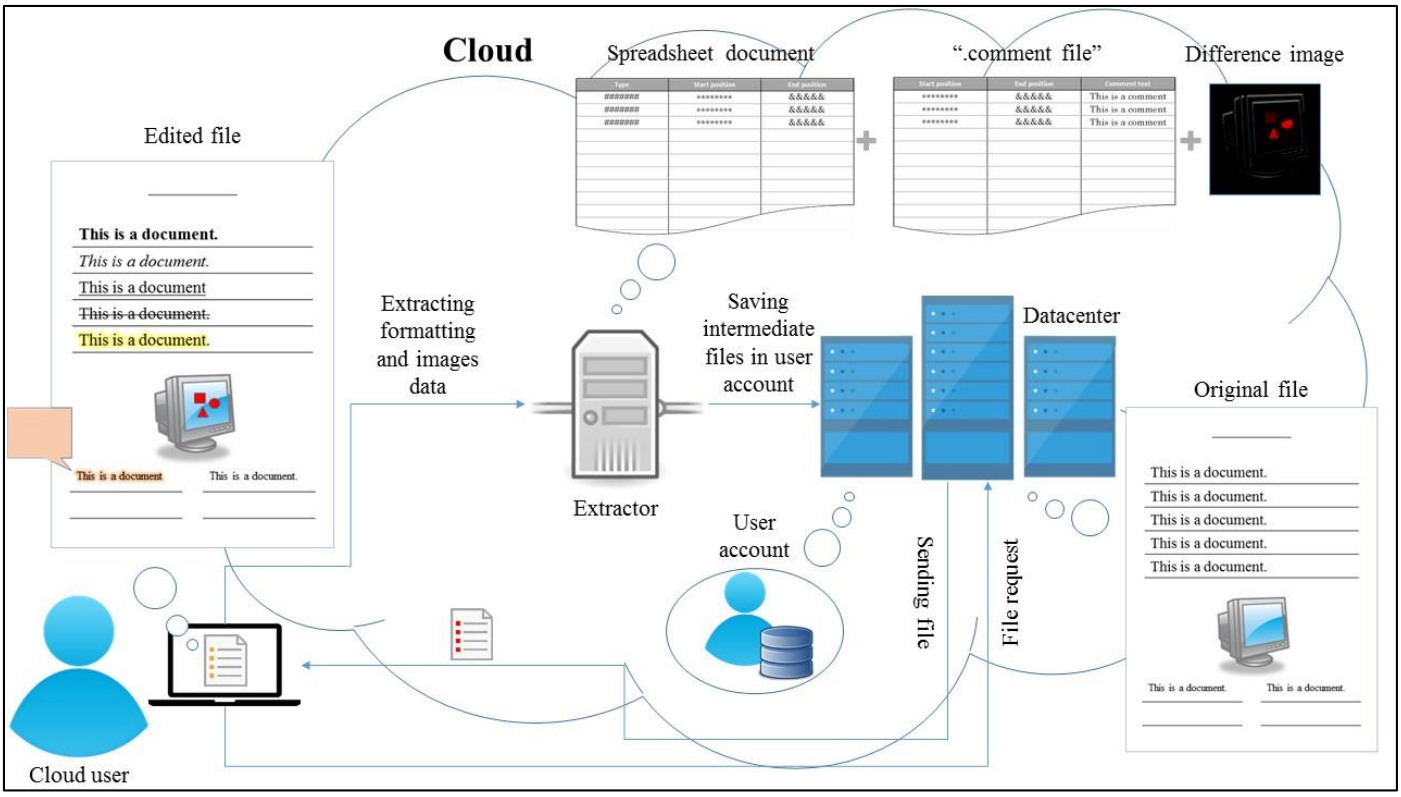

Fig. 3: System architecture from Extractor point of view

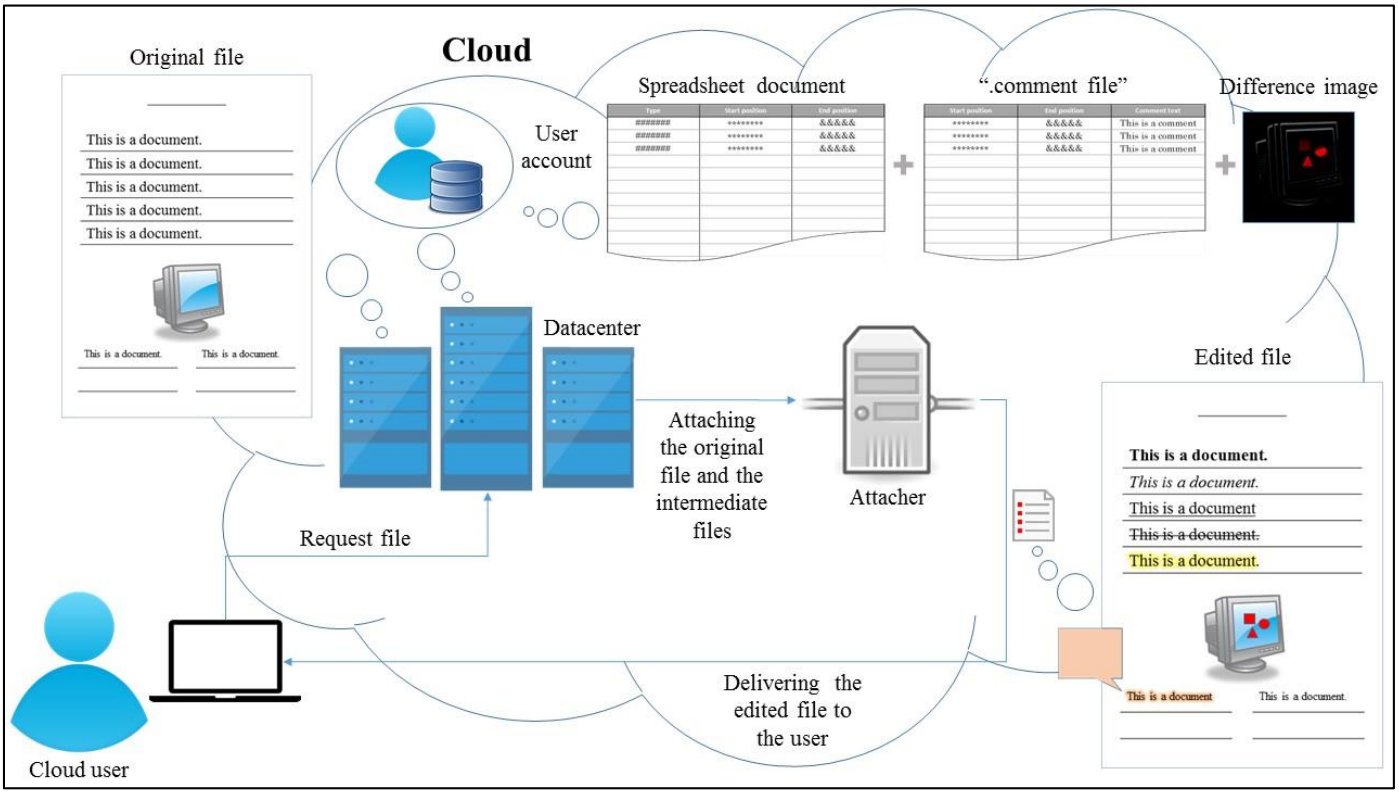

Fig. 4: System architecture from Attacher point of view

The results are investigated in Average-case that is defined to be the situation in which just the key points of the document are highlighted or in the other words, the document is annotated normally. We obtained the average case by averaging the results acquired from all users. The algorithm is evaluated for 100 users taking two different courses. The criterion for choosing courses is their lecture size. We consider two different lectures in size.

Table II shows size of the files, which have been used for the experiment. The first column is the original file size of two different courses and the second column is the average size of file after being edited normally by 100 users.

TABLE II
SIZE OF FILE AFTER BEING EDITED

Original file size	Size of file after being edited
5886 KB = 5.748 MB	6703 KB = 6.546 MB
8659 KB = 8.456 MB	10040 KB = 9.805 MB

Table III shows the results of using traditional approach. The first column is the size of document after annotating and the second column is the total size for 100 users. It is clear that the total file size becomes 100 times bigger than the original file.

TABLE III
TRADITIONAL APPROACH

Original file size	Total file size for 100 users
6703 KB = 6.546 MB	670300 = 654.6 MB
10040 KB = 9.805 MB	1004000 = 980.5 MB

Table IV shows the results of using the proposed approach in the average-case. The first column is the size of the original file and the second column is the intermediate files size. The third column is the size of original file plus 100 intermediate files and the original images in file that is the total volume of files in the proposed approach.

TABLE IV
PROPOSED APPROACH

Original file size	Spreadsheet document for text parts + ".comment" file + Difference images	Total file size for 100 users
5886 KB = 5.748 MB	881.457 KB	94031.700 KB = 91.828 MB
8659 KB = 8.456 MB	647.4 KB	73399.000 KB = 71.6 MB

Fig. 5 shows the comparison of two different cases in which files can be stored in cloud server. The results for large data files imply that a large amount of storage can be saved by the proposed approach.

In Table V, second column shows the efficiency of proposed approach respect to the traditional approach in average-case. It means how much it can be effective in storing data. We define the efficiency in terms of TA and PA results where PA is the proposed and TA is the traditional approach. The formula below declares the efficiency of the proposed approach:

$$\text{Efficiency} = 1 - (\text{PA Result} / \text{TA Result}) \quad (1)$$

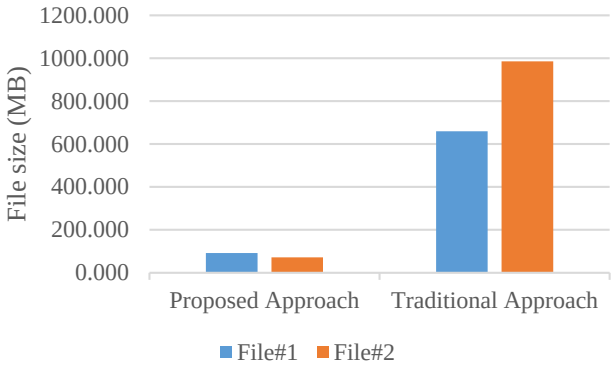

Fig. 5: Comparison of Traditional Approach and the Proposed Approach

TABLE V
RESULTS OF COMPARISON IN TERMS OF EFFICIENCY

Original file size	Efficiency
5886 KB = 5.748 MB	0.860803617
8659 KB = 8.456 MB	0.927284183

The results show when the size of the file goes larger, the efficiency ratio will rise and this means the difference between the size of file in the traditional approach and the size of file in proposed approach becomes bigger and bigger. Now we return to our example and check the result for it. Total space required to save students data for one courses is 14.825TB. If we apply this approach, the space required decrease to about 1.472TB. It means that about 13.353TB can be saved in cloud for other purposes.

V. CONCLUSION

Cloud computing has recently a large impact on industry and every day the number of users which go toward using its services grows on. This leads to a large and significant growth of data storage by cloud server. Thus, using a method which served us an efficient way of storing our data can be useful.

The proposed approach aims to provide an efficient approach to store generally-shared and individually-customized text files. It is to detect formatting data of a document like position of formatted words or any annotation on document. Formatting means that any words can be highlighted, underlined, be strikethrough, bolded or be italic. The algorithm can be efficient and decrease various costs in a company or institute. The algorithm is simple and easy to implement. For a file that is shared through Internet via millions of people and only its content wants to be changed, the total size can be enormous. Hence, using the proposed approach can save significant disk space in cloud server.

Many key challenges in this field and domain are starting which their key purpose is to propose various approaches of storing data in cloud server in efficient way in order to reduce different costs for the company or institute that are using this technology.

VI. REFERENCES

[1] M. Bahrami and M. Singhal, "The Role of Cloud Computing Architecture in Big Data," in *Information Granularity, Big Data, and Computational Intelligence*, Merced, Springer International Publishing, 2015, pp. 275-295.

[2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh and A. Hung Byers, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute, 2011.

[3] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. S. Pierre, S. Twigger, O. White and S. Y. Rhee, "Big data: The future of biocuration," *Nature*, vol. 455, pp. 47-50, 2008.

[4] W. Tan, M. B. Blake, I. Saleh and S. Dustdar, "Social-Network-Sourced Big Data Analytics," *IEEE Internet Computing*, vol. 17, no. 05, pp. 62-69, 2013.

[5] D. Vellante, "Revisited: The Rapid Growth in Unstructured Data," Wikibon, 17 08 2010. [Online]. Available: http://wikibon.org/blog/unstructured-data/. [Accessed 05 12 2014].

[6] L. A. Wilson, "Survey on Big Data gathers input from materials community," *MRS Bulletin*, vol. 38, no. 09, pp. 751-753, 2013.

[7] M. Hanna, "Data mining in the e-learning domain," *Campus-Wide Information Systems*, vol. 21, no. 01, pp. 29-34, 2004.

[8] M. Buscema, E. Grossi, D. Snowdon and P. Antuono, "Auto-Contractive Maps: An Artificial Adaptive System for Data Mining. An Application to Alzheimer Disease," *Current Alzheimer Research*, vol. 05, no. 05, pp. 481-498, 2008.

[9] H. Jing, W. Kai, L. Lok Kei, M. Seungbeom and M. Melody, "A Tunable Workflow Scheduling Algorithm Based on Particle Swarm Optimization for Cloud Computing," *International Journal of Soft Computing and Software Engineering (JSCSE)*, vol. 03, no. 03, pp. 351-358, 2013.

[10] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 01, no. 01, pp. 7-18, 2010.

[11] Wikipedia, "Cloud computing," Wikipedia, 11 02 2015. [Online]. Available: http://en.wikipedia.org/wiki/Cloud_computing. [Accessed 13 02 2015].

[12] Wikipedia, "Dropbox," Dropbox, 09 2008. [Online]. Available: https://www.dropbox.com/. [Accessed 10 12 2014].

[13] Wikipedia, "Google Drive," Google, 24 04 2012. [Online]. Available: https://www.google.com/drive/. [Accessed 10 12 2014].

[14] Wikipedia, "Apple - iCloud," Apple, 12 10 2011. [Online]. Available: https://www.apple.com/icloud/. [Accessed 10 12 2014].

[15] Wikipedia, "Microsoft OneDrive," Microsoft, 01 08 2007. [Online]. Available: https://onedrive.live.com/. [Accessed 10 12 2014].

[16] Apache Software Foundation, "Welcome to Apache (TM) Hadoop," Apache Software Foundation, 27 12 2011. [Online]. Available: http://hadoop.apache.org/. [Accessed 18 12 2014].

[17] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 01, pp. 107-113, 2008.

[18] N. Bonvin, "A self-organized, fault-tolerant and scalable replication scheme for cloud storage," in *1st ACM symposium on Cloud computing*, Indianapolis, 2010.

[19] B. Dong, J. Qiu, Q. Zheng, X. Zhong, J. Li and Y. Li, "A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop: a Case Study by PowerPoint Files," in *IEEE International Conference on Services Computing*, Miami, 2010.

[20] W. Tantisiriroj, S. Patil and G. Gibson, "Data-intensive file systems for Internet services: A rose by any other name...," Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-08-114, Pittsburgh, 2008.

[21] L. Jiang, B. Li and M. Song , "The optimization of HDFS based on small files," in *3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, Beijing, 2010.

[22] X. Zhang and F. Xu, "Survey of Research on Big Data Storage," in *12th International Symposium on Distributed Computing and Applications to Business, Engineering & Science*, Kingston upon Thames, 2013.

[23] Y. Zhang and D. Liu, "Improving the Efficiency of Storing for Small Files in HDFS," in *International Conference on Computer Science and Service System*, Nanjing, 2012.

[24] G. Mackey, S. Sehrish and J. Wang, "Improving Metadata Management for Small Files in HDFS," in *IEEE International Conference on Cluster Computing and Workshops (CLUSTER '09)* , New Orleans, 2009.

[25] K. Sangeetha and P. Prakash, "Big Data and Cloud: A Survey," in *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems (ICAEES 2014)*, 2015.

[26] F. Chang, J. Dean , S. Ghemawat , W. C. Hsieh , D. A. Wallach , M. Burrows , T. Chandra , A. Fikes and R. E. Gruber , "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 02, 2008.

[27] B. F. Cooper, R. Ramakrishnan , U. Srivastava , A. Silberstein , P. Bohannon , H.-A. Jacobsen , N. Puz, D. Weaver and R. Yerneni , "PNUTS: Yahoo!'s hosted data serving platform," *Proceedings of the VLDB Endowment*, vol. 01, no. 02, pp. 1277-1288, 2008.

[28] M. Isard, M. Budiu, Y. Yu, A. Birrell and D. Fetterly , "Dryad: distributed data-parallel programs from sequential building blocks," in *2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, Lisbon, 2007.

[29] Microsoft, "Microsoft SQL Azure," Microsoft, [Online]. Available: http://azure.microsoft.com/en-us/. [Accessed 16 12 2014].

[30] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels, "Dynamo: amazon's highly available key-value store," in *21st ACM SIGOPS symposium on Operating systems principles*, Stevenson, 2007.

[31] A. Abouzeid , K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz and A. Rasin, "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads," *Proceedings of the VLDB Endowment*, vol. 02, no. 01, pp. 922-933, 2009.

[32] Y. Bu , B. Howe, M. Balazinska and M. D. Ernst, "HaLoop: efficient iterative data processing on large clusters," *Proceedings of the VLDB Endowment*, vol. 03, no. 01, pp. 285-296, 2010.

[33] Wikipedia, "Google Docs," Google, 06 06 2006. [Online]. Available: http://www.google.com/docs/about/. [Accessed 05 01 2015].

[34] B. Balasubramanian, T. Lan and M. Chiang, "SAP: Similarity-Aware Partitioning for Efficient Cloud Storage," in *IEEE Conference on Computer Communications (INFOCOM 2014)*, Toronto, 2014.

[35] W. Voorsluys, J. Broberg and R. Buyya, "Introduction to Cloud Computing," in *Cloud Computing: Principles and Paradigms*, Weily, 2011, pp. 3-41.

[36] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang and K. Pentikousis, "Energy-Efficient Cloud Computing," *The Computer Journal*, vol. 53, no. 07, pp. 1045-1051, 2009.

[37] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 05, pp. 755-768, 2012.

[38] R. Buyya, A. Beloglazov and J. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges," in *International*

Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010), Las Vegas, 2010.

[39] C. Vecchiola, S. Pandey and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, Kaohsiung, 2009.

[40] D. Agrawal, S. Das and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in 14th International Conference on Extending Database Technology , Uppsala, 2011.