A New Efficient Approach to Store Data in a Cloud Server

Hussein Shirvani
Pervasive & Cloud Computing Lab
Department of Computer Engineering
University of Birjand, Iran
hussein.shirvani.1992@ieee.org

Hamed Vahdat-Nejad
Pervasive & Cloud Computing Lab
Department of Computer Engineering
University of Birjand, Iran
vahdatnejad@birjand.ac.ir

ABSTRACT
The present study aims at introducing a new efficient approach to store data in a cloud server. The approach is applicable to a group of annotated text files, which are supported by Microsoft Word (docx, doc, pdf, etc.). Instead of the original document, an intermediate file is produced and saved as a Microsoft Excel document for each user. The file contains edited information including highlighted and underlined parts, comments, etc. Whenever a user wants to access his/her document, the intermediate file is attached to the original document and appeared to him/her. The experimental results of 50 users show that a 2.5MB file and in total, 126MB files can be reduced in size to 3MB. It means a large amount of disk space can be saved by the cloud server.

Keywords
Cloud computing, IaaS, PaaS, Storage

1. INTRODUCTION
Cloud computing is one of the advanced topics in Computer Science, which has gained numerous impact in industry. According to Zhang [1], "cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet." It 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 demand for service rises.

Three main and major models of this computing are known as infrastructure as a service (IaaS), software as a service (SaaS) and platform as a service (PaaS), which can be offered in a private, public or hybrid network. Among these services, IaaS and PaaS are of our concern. IaaS is defined as "on-demand provisioning of infrastructural resources, usually in terms of Virtual Machines" [1]. IaaS cloud providers supply these resources in an on-demand manner from their large installed data centers. PaaS refers to "providing platform layers resources, including operating system support and software development frameworks" [1]. Therefore, the PaaS cloud providers must deliver a computing platform, usually including database, web server and the like.

It can be inferred from these definitions that these services interact with storage resources. Hence, it is clear that storage resources have major roles in cloud servers. Every day, the number of cloud users increases and thus their storage needs rise. This prompts the need for using an efficient approach to store data in a cloud. Nowadays, several companies provide cloud storage services for storing, sharing and accessing users' data such as the Microsoft OneDrive[2], which is created by the Microsoft Corporation. Users can utilize these services through Internet, disregard of their location.

As users increase, a storage service provider company must boost its resources in order to remain its services alive and active. Developing efficient methods for storing data in a cloud can slow the movement to install more resources. An important type of data stored in the cloud involves text files, which are generally shared and individually annotated by a large number of users. For example, in a university, students enrolled in a course want to access their instructor's lecture notes, which are saved in the

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.257469

cloud server. However, each enrolled student wants to customize (e.g. highlight, annotate, etc.) them individually. Another example is a book, lecture, etc., which is generally shared via Internet among millions of people, in a way to be customizable by them. The traditional approach to store all the customized files, results in a huge mass of data storage, which mainly consists of redundant data.

The present study proposes a new approach to storing generally-shared individually-customized text data in a cloud server. The approach is implemented for Microsoft Word text file formats including docx, doc, pdf, etc. It is based on storing the original shared files in the cloud and an intermediate file for each user, which contains edited information including highlights and underlined parts and comments. When a user wants to access his/her customized document, the intermediate file will be applied to the original document and appeared to him/her. On this basis, a large amount of disk space can be saved.

The rest of the paper is organized as follows: In section 2, the proposed approach (PA) will be explained and discussed in more details including the pseudo-code and system architecture which is employed to solve the problem. In section 3, the application interface and some major parts of code will be shown. In section 4, the evaluation results will be described via three tables, and finally in section 5, we conclude our work.

2. PROPOSED APPROACH

To decrease redundancy, the proposed approach aims at storing the original text file just the once. On the other hand, an intermediate file, which contains customizations, is created and stored for each user. In continue, the details of the proposed approach are provided.

In the text files such as a Microsoft Word document, each word has some attributes. For instance, font color, effects, highlight color, and position in the document. The two former attributes are within our focus. With these attributes, the needed data is 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 (underline and the like) some parts of the document. Then she quits Microsoft Word application and asks the application to save his/her changes. The proposed algorithm begins at this time and it is hidden from user's view.

At first, the server checks the document for highlighting words and when it receives a signal, it extracts the position of the highlighted word, which includes the start and end point of it. It has been supposed that the highlight color is yellow by default, but it is possible to change highlight color or design the program to support more colors. When the server reaches the end of the document, a list of highlighted words position (start and end point) is in our hand. It saves the list in a Microsoft Excel document. At the end, the server saves this Excel document at user account of the cloud for later use and access. The pseudo-code of the extraction algorithm is shown below.

Algorithm 1 Extraction Algorithm

```
1:  procedure EXTRACT(WD)
    the words in Word document WD must be checked for the highlight attribute
2:      Create a list L with two attributes, start position and end position;
3:      Open the Word document WD and put the cursor at the beginning of the document
4:      Check the document word by word for the highlight attribute
5:      if one word is highlighted then
                Save the start and end position of this word in L
6:      end if
7:      if not EOF then
                Process the next word and go to 4
8:      end if
9:      Close the Word document WD without any changes
10:     Open a new Excel document ED
11:     Save L in ED in two columns, the first column is starting point and the second is the end point
12:     Save ED in the user account for later access.
13:     Close the Excel document ED
14: end procedure
```

The proposed approach checks the document for highlighted words and if it detects one of them, it saves its start and end point in a list, and finally,

when the end-of-file signals, the product list will be saved in a Microsoft Excel document.
On the other hand, when the user decides to access the right document and opens it, the information in Microsoft Excel document which has been saved before in the user account, will be attached to the original file and will be appeared to him/her. The pseudo-code of the attach algorithm is shown below.

Algorithm 2 Attach Algorithm

```
1:  procedure Attach(WD, ED)
    the highlight attribute must be applied to each position was saved in Excel document ED related to Word document WD
2:      Open the Microsoft Excel document ED
3:      Process the first row of ED
4:      Apply the highlight attribute to the start and end point of word in WD that is read from ED
5:      if any valued row exists then
            Process the next row and go to 4
6:      end if
7:      Close ED
8:      return WD
9:  end procedure
```

Therefore, the Microsoft Excel document will be opened, read row by row and the highlight attribute will be applied to each given start and end point for attaching the highlighted data.
This document is hidden from the user view by default, but it can be accessed by authorized users such as cloud server administrator or some others who have this authority.
Figure 1 shows the proposed system architecture. The original file is saved in the cloud using a storage server and then each user can access and edit it. It is shown in the figure that user 1 to 3 highlight different parts of the original file. When they close the file, the application is triggered and operates for each user and detects any formatting changes in that file. Then the intermediate file is saved in the cloud and the original file remains fresh and unchanged.

3. IMPLEMENTATION

Since C# has various libraries for manipulating Microsoft Office family documents and files, we make use of it for implementing the proposed algorithm. The program interacts with Microsoft

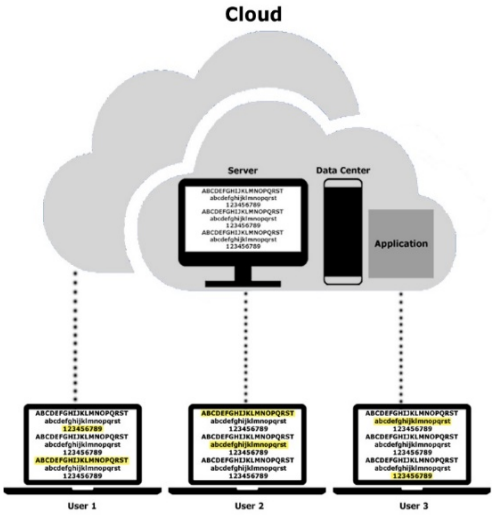

Figure 1: system architecture

Word and Microsoft Excel, so, the two corresponding libraries for accessing data on this two application documents are needed.

```
// check the word whether it was highlighted or not
if (docRange.HighlightColorIndex == Word.WdColorIndex.wdYellow)
{
    startOfWord = docRange.Start; // start point
    endOfWord = docRange.End; // end point
    // determine that a highlighted word is detected
    flagForHighlight = 1;
} // end if
// if the word was highlighted then
// add its start and end point to the list
if (flagForHighlight == 1)
{
    // determine the format type
    myFormattingDataList.Add(1);
    // add start point to the list
    myFormattingDataList.Add(startOfWord);
    // add an end point to the list
    myFormattingDataList.Add(endOfWord);
    // clear flag for later check
    flagForHighlight = 0;
} // end if
```

Figure 2: Part of the code that extracts formatting data from word

Figure 2 shows one of the most important parts of the application, which is the extraction function.

This part checks each word of the document and if a word is highlighted, the first if-statement begins execution. Afterwards, two variables `startOfWord` and `endOfWord` get the start and end point of the word, respectively, and the second if-statement checks for highlight occurrence. If it occurs in a word, the start and end point will be saved in a list which contains the start and end point of highlighted words. This process continues until the detection of end of file. Every time the user wants to read its document, this information will be attached to the original file.

Figure 3 shows another important part of the application code. This is part of the attach function. In this process, the Microsoft Excel document is read and copied in a list in order to be accessed during program execution. Then the list is read and for each saved position, the highlight attribute is applied to the word in that position. This part is going to be executed whenever the user decides to access the document.

```
// read position data from list
foreach (FormattingInfo information in info)
{
    start = information.start; // start point of word
    end = information.end; // end point of word

    // set the cursor on obtaining positions
    // in order to highlight that word or set of words
    doc.Range(ref start, ref end).HighlightColorIndex =
        Microsoft.Office.Interop.Word.WdColorIndex.wdYellow;
} // end foreach
```

Figure 3: Part of the code that attaches formatting data to its relative word document

4. EVALUATION

The results are investigated for two cases: Worst-case and average-case. Worst-Case is defined to be the situation in which all of the words in the document are highlighted, underlined or the likes. Average-Case 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 highlighted normally. We obtain the average case by averaging the results acquired from all users. The algorithm is evaluated for 50 users taking four different courses. The criterion for choosing courses is their lecture size. We consider four different lectures from low (96 KB) to average (2509 KB) sizes.

Table 1 shows size of the files, which have been used for the experiment and the results gathered from the traditional approach. The first column is the original file size of four different courses from small to medium. The second column is the total size of files that must be saved in data center for 50 users. It is clear that in the traditional approach the total file size becomes 50 times bigger than the original file size.

Original file size	Total file size for 50 users
2509KB	125450KB
1855KB	92750KB
112KB	5600KB
96KB	4800KB

Table 1: Traditional Approach

Table 2 shows the results of using the proposed approach in the average-case scenario. The first column is size of the original file and the second column is the intermediate file size. The third column is size of the original file plus 50 intermediate files for 50 users.

Table 3 shows the results of applying the proposed approach in the worst-case scenario. The columns are like Table 2. Of course the worst-case is rare and unrealistic, but even in this case, the results are acceptable for large files. However for small-size files, applying the proposed approach can be inefficient.

Original file size	Intermediate file size	Total file size
2509KB	22KB	3609KB
1855KB	9KB	2305KB
112KB	9KB	562KB
96KB	9KB	546KB

Table 2: Proposed Approach - Average-Case

Figure 4 Results of comparing the traditional approach and the proposed approach. Figure 4 compares three different cases. The size of the intermediate file is almost the same as the original file when the size of document becomes smaller. This implies that the algorithm works well for medium size files in both cases, while working well for small size files just in the average-case. Also the results for large data files imply that a large amount of storage can be saved by the proposed approach.

Original file size	Intermediate file size	Total file size
2509KB	440KB	24509KB
1855KB	113KB	7505KB
112KB	54KB	2812KB
96KB	96KB	4896KB

Table 3: Proposed Approach - Worst-Case

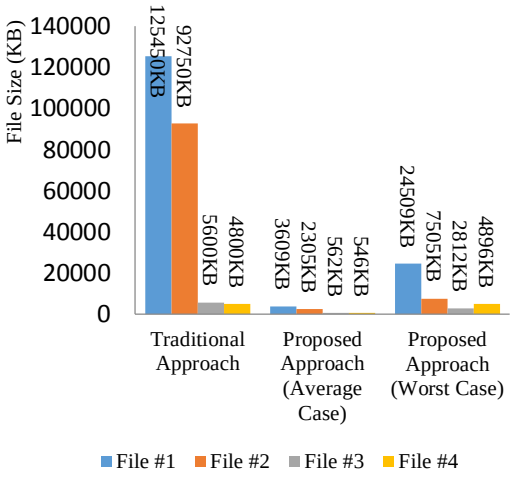

Figure 4: Comparison of Traditional Approach and the Proposed Approach

In Table 4 second and third columns show the efficiency of proposed approach with respect to the traditional approach in both cases of average-case and worst-case. We define efficiency as follows:

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

where PA is the proposed and TA is the traditional approach.

The results show when the size of the file becomes larger, the efficiency of the proposed approach rises.

Original file size	Efficiency in average-case	Efficiency in worst-case
2509KB	0.971	0.804
1855KB	0.975	0.919
112KB	0.899	0.497
96KB	0.886	-0.02

Table 4: Results of Comparison in terms of Efficiency

5. CONCLUSION

Cloud computing has recently gained a large impact on industry and its users increases. This leads to a large and significant growth of data storage by cloud server. The proposed approach aims to provide an efficient approach to store generally-shared and individually-customized text files. The algorithm is simple and easy to implement. For a file that is shared through Internet among millions of people and only its content wants to be customized by users, the total size can be enormous. Hence, using the proposed approach can save significant disk space in a cloud server.

6. REFERENCES

[1] Q. Zhang, L. Cheng, R. Boutaba, "Cloud Computing: state-of-the-art and research challenges", Journal of Internet Services and Applications, vol. 1, pp. 7-18, May 2010

[2] Microsoft OneDrive, https://onedrive.live.com, 19 April 2014

[3] Cloud computing on Wikipedia, en.wikipedia.org/wiki/Cloud computing, 25 April 2014
[4] W. Voorsluys, J. Broberg, R. Buyya"Introduction to Cloud Computing". In R. Buyya, J. Broberg, A.Goscinski. Cloud Computing: Principles and Paradigms. New York, USA: Wiley Press. pp. 1–44. ISBN 978-0-470-88799-8.
[5] A. Berl et al., "Energy-Efficient Cloud Computing", the Computer Journal, vol. 53, pp. 1045-1051, August 2009
[6] A. Beloglazov, J. Abawajy, R. Buyya "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing", Future Generation Computer Systems, Elsevier, vol. 28, pp 755–768, May 2012
[7] R. Buyya, A. Beloglazov, J. Abawajy "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges", Future Generation Computer Systems, vol. 28, pp. 755-768, May 2012
[8] C. Vecchiola, S. Pandey, and R. Buyya "High-Performance Cloud Computing: A View of Scientific Applications", in: Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, ACM, pp. 4-16