International Journal of Research in Engineering, Science and Management
Volume-2, Issue-3, March-2019
www.ijresm.com | ISSN (Online): 2581-5792

Data Deduplication in Cloud by Chunking

R. Saranya[1], S. Vidhya[2], M. Muthumari[3], B. Sangeerthana[4]
[1]*Assistant Professor, Department of Computer Science and Engineering, P.S.R. Rengasamy College of Engineering for Women, Sivakasi, India*
[2,3,4]*UG Student, Department of Computer Science and Engineering, P.S.R. Rengasamy College of Engineering for Women, Sivakasi, India*

***Abstract*: Data deduplication is an emerging technology that introduces reduction of storage utilization and an Efficient way of handling data replication in Secondary Storage. In the deduplication, data are divided into "multiple chunks" and unique hash identifier is identified with every chunks. These identifiers are used to compare the chunks with previously stored chunks and verified for duplication. High throughput hash less chunking method called Rapid The maximum-valued byte is included in the chunk and located at the boundary of the chunk. We propose chunking method called Rapid Asymmetric Maximum (RAM). Rapid Asymmetric Maximum (RAM) which improves the chunking throughput of AE by putting the extreme value at the boundary of the chunk. It has a low computational overhead which makes the algorithm faster than existing CDC algorithms. The low computation overhead of RAM reduces the cost of chunking process which makes chunking more attractive over AE for low performance devices such as mobile devices and IoT.**

***Keywords*: Chunking, Deduplication, Rapid Asymmetric Maximum(RAM), Asymmetric Experimum (AE), Content Defined Chunking(CDC).**

1. Introduction

In hybrid cloud approach for secure authorized deduplication. Data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data, and has been widely used in cloud storage to reduce the amount of storage space and save bandwidth. To protect the privacy of sensitive data while supporting deduplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing. To better protect data security, this paper makes the first attempt to formally address the problem of authorized data deduplication. Different from traditional deduplication systems, the differential privileges of users are further considered in duplicate check besides the data itself. We also present several new deduplication constructions supporting authorized duplicate check in a hybrid cloud architecture. Security analysis demonstrates that our scheme is secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement a prototype of our proposed authorized duplicate check scheme and conduct test bed experiments using our prototype. We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.

2. Background

Chunking is used in many data compression applications. For example, it is used in data deduplication and remote differential compression. Data deduplication works by eliminating duplicate data within the files and between files. In data deduplication, a chunking algorithm is one of the vital parts to achieve high duplicate elimination. By choosing the correct chunking method, we can save time and space. Data deduplication can be applied on cloud storage, virtual disk images, memory, and network traffic. One of the applications of data deduplication is remote differential compression. Remote differential compression does not save space but it saves network bandwidth and time by sending only the parts that are not available to the receiver as stated by Teodosio et. al. Additionally, Ruppin et. al. proposed a data synchronization system that uses chunking for data synchronization across multiple devices. Chunking algorithms can be categorized into two categories: (i) whole file chunking and (ii) block chunking. Whole file chunking means the whole file is treated as one chunk, while block chunking means the file is split into multiple chunks. When chunking a file into blocks or chunks, the chunk size can be fixed-sized or variable-sized. Fixed-sized chunking is fast and not resistant to byte insertion or shifting. When the file is shifted by a byte insertion or deletion, the chunks will become completely different chunks and undetectable by the chunk duplicate search. Content Defined Chunking (CDC) solves this problem by chunking the file into variable-sized chunks. CDC algorithms find the cut point by using internal features of the file. Therefore, when the file is shifted, only several chunks are affected. CDC has a higher probability of eliminating duplicates within the files and between files compared to fixed-sized chunking.

One of the oldest CDC algorithms is Rabin [4] based CDC algorithm. It finds the cut-point by using Rabin rolling hash. Rabin rolling hash uses sliding window and every time the window is moving, a hash value is calculated. When the hash value matches a predefined value, it uses the window position for the hash value as a cut-point. Since the checksum is calculated based on polynomials over a finite field, the old checksum can be used to calculate the new checksum when the window slides.

3. Motivation

CDC offers more benefits than fix-sized chunking. However, CDC process is slightly more time-consuming which limits the use of CDC algorithms on latency-critical applications and on devices with limited processing capability such as mobile devices and Internet of Things (IoT) devices. In our previous work, we used Rabin based chunking algorithm for the deduplication system to eliminate duplicate data. We found out that the main drawback of using CDC algorithms in the mobile application is its large processing time. Before we discuss and compare various CDC algorithms, we would like to state the following criteria that can be used to compare CDC algorithms, which is used by Zhang et. al. in:

- Content dependent: The algorithm should define the cut point based on the internal features of the file, which makes it resistant against byte shifting and allows the algorithm to find duplicate chunks between two or more files.
- Low chunk sizes variance: The chunks produced by the algorithm should have low chunk variance because it might affect the deduplication efficiency. To limit the chunk variance, we can add a limit on the maximum or minimum size of the chunks. However, this will affect the content dependent properties of the algorithm and make the algorithm vulnerable against byte shifting.
- Ability to eliminate low entropy strings: Low entropy strings are strings which consist of repetitive bytes or patterns. When it encounters strings with low entropy or low variance, it is Prefer able for the algorithm to be able to eliminate the redundancy within the string.
- High throughput and duplicate detection: The algorithm should have a good balance between deduplication performance and computational overhead.

Local Maximum Chunking (LMC) [3] is a CDC algorithm that compares bytes with bytes as a number to find the cut point. LMC has a resistance against byte changing and byte shifting. When there is a change in the chunk and the change has a value less than the maximum, it will only affect that chunk. The main drawback of this method is the requirement of rechecking all the bytes within the window when the window slides. This drawback makes Rabin-based CDC algorithms faster than LMC method because when the sliding window of Rabin slides, it only needs to subtract the most left byte and add the new byte into the hash. However, LMC needs all of the bytes in the window every time it slides the window. Rabin based CDC algorithm uses polynomial over a finite field and a sliding window to calculate the hash [4].

Rabin-based CDC algorithms have a few disadvantages due to the use of the hash. It is time-consuming because of the hash calculation, and changing a byte in the chunk has a high probability of changing the cut-point as it might create a different hash value. It also has a large chunk variance because of the higher probability of having a long chunk [3], [5]. In order to limit the chunk variances, we can use a limit on the chunk size. However, this will reduce the resistance of the algorithm against byte shifting.

Fig. 1. Original data

Local Maximum Chunking (LMC) [3] is a CDC algorithm that compares bytes with bytes as a number to find the cut point. LMC has a resistance against byte changing and byte shifting. When there is a change in the chunk and the change has a value less than the maximum, it will only affect that chunk. The main drawback of this method is the requirement of rechecking all the bytes within the window when the window slides. This drawback makes Rabin-based CDC algorithms faster than LMC method because when the sliding window of Rabin slides, it only needs to subtract the most left byte and add the new byte into the hash. However, LMC needs all of the bytes in the window every time it slides the window.

AE is similar to the local maximum method because it treats a byte as a number. Treating the chunk as the windows allows AE to have a lower computational overhead than the LMC method. However, unlike the LMC method, AE puts the extreme-valued byte in the middle of the chunk. This makes AE less resistant to byte shifting. When there is a byte inserted at the fixed window, it will affect the chunk and the next chunk and might affect subsequent chunks. If we put the extreme-valued byte at the boundary of the chunk, inserting a byte will not affect the next chunk. Thus, it minimizes the number of affected bytes. AE is capable of eliminating low entropy strings because AE has maximum chunk size. AE reach its maximum chunk size when it processes a long increasing sequence. The maximum chunk size is 256 bit is the length of the fixed window.

```
Algorithm 1: Algorithm for AE chunking
Input: input string, Str, left length of the input string, L;
Output: cut-point I;
Predefined values: window size, w;
function AEChunkning (Str, L)
        i=1;
        while (i<L)
                If Str[i].value<=max.value then
                        if i=max.position+w then
                                return i
                        end if

                else
                        max.value=Str[i].value
                        max.position = i
                end if
                i=i+1
        end while
end function
```

Fig. 2. The pseudo code for AE chunking

4. Rapid Asymmetric Maximum Algorithm (RAM)

Rapid Asymmetric Maximum Algorithm (RAM and analyze the chunking properties of the algorithm. With a goal of achieving low computational overhead and byte shift-resistant algorithm, we proposed a boundary version of AE, called RAM. RAM is similar to AE because it also uses two windows: fixed and variable-sized windows. The placement of the windows is, however different from AE. In RAM, the fixed-sized window is located at the beginning of the chunk and followed by the variable-sized window and the maximum-valued byte the maximum-valued byte is included in the chunk at the end of the chunk in the case of RAM. The algorithm works by searching a byte with the maximum value in the fixed-sized window. If the byte next to the fixed-sized window has larger value than the one in the fixed-sized window, the byte is used as the maximum-valued byte and the cut-point is found. Otherwise, the algorithm moves to the next byte until it finds the larger byte as illustrated in the pseudo window.

The chunking used in our proposed RAM scheme. RAM reduces the computation time by searching the byte that is equal or larger than the current maximum value, while AE process all the bytes smaller or equal than the maximum-valued bytes. Since the probability that the next byte is smaller than the current maximum value is higher than the probability that the next byte is larger than the current maximum value, RAM enters the first condition less frequently than AE. This lowers RAM's overhead.

```
Algorithm 1 RAM Chunking
Input: input string, Str, left length of the input string, L;
Output: cut-point l;
Predefined values: window size, w;
function   NAE Chunking
        i=1;
        while (i<L)
                If Str[i].value >=max.value then
                        if i>w then
                                return i
                        end if

                else
                        max.value=Str[i].value
                        max.position = i
                end if
                i=i+1
        end while
end function
```

Fig. 3. The pseudo code for RAM chunking

A. Low chunk sizes

variance and the ability to eliminate low entropy strings. RAM has a low probability of long chunk as explained in However, low entropy string is a problem for RAM. When the low entropy string starts at the beginning of the fixed-sized window, RAM is able to eliminate the low entropy string because the condition for a cut point is that the maximum-valued byte must be equal to or larger than the maximum in the fixed window. On the contrary, when there is a byte larger than any value in the low entropy string is in the fixed-sized window, the chunk size can become infinite because it cannot find a byte with larger value. To solve this, we can add a limitation on the maximum chunk size. High throughput: RAM has a low performance overhead. To prove that, we use the worst case of RAM, based on the number of comparisons. RAM uses while loop which takes comparisons and two additional conditional branches which add comparisons, where is the length of the input data stream in bytes. comparisons in the worst case scenario. Since the probability of finding a byte larger than the max is smaller than finding a smaller byte, on the average case it uses comparisons. In the application, the probability that the value of the maximum byte in the fixed window being big is higher than the probability that the byte being small. We prove this by assuming that the data entries are random Splitting RAM into two parts: the fixed window part and the variable-sized part, where the fixed window part's length is and the variable. sized part's minimum size is one.

B. Authentication

The process of identifying an individual usually based on a username and password. In security systems, Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual. In authentication module is used to security purpose. Here this module only for user, after registration user enter the username and password. This input is check into the database, whether input is correct or not. If input is correct then allow to next process otherwise consider as a non-authenticated user.

C. Register

In this Module If he is a new user he needs to enter the required data to register the form and the data will be stored in server for future authentication purpose.

D. File uploading:

In this scheme user upload the files in the cloud server. Cloud can store multiple files. Collect several file from the stored in the Cloud Server.

E. Chunking algorithm

Chunking Algorithm In data deduplication, the basic idea is to split a file into blocks and applies hash functions to compute hash values. To check data duplication the client sends the hash key list to the server. The hash key for each chunk is used to determine if that chunk exists in the multiple locations by comparing hash keys. If there are same hash keys on another location, we assume that the chunk is duplicated. Therefore, we can prevent duplicated data blocks to be transferred. Generally, the chunking algorithms are divided into two; fixed length chunking and variable length chunking. The fixed length chunking approach achieves very fast data deduplication result but the performance is not good; because boundary shift problem degrades the deduplication performance. On other hand, variable length chunking achieves high degree of performance while causing high computation overhead and longer processing time.

F. De-duplication

Cloud can store and retrieve file. De-duplication has a

removing duplicate file. Its will find out duplicate file. Deduplication The Admin is the data owner who performs deduplication by checking if the contents of two files are the same and stores only one of them. Here the data owner upload, download and update the files. Then the deduplication is performed by applying the RAM Algorithm

5. User authentication

A. *Registration*

If you are the new user going to login into the application then you have to register first by providing necessary details. After successful completion of sign up process, the user has to login into the application by providing username and exact password.

B. *Login*

The user has to provide exact username and password which was provided at the time of registration, if login success means it will take up to main page else it will remain in the login page itself.

C. *View details*

In this scheme user after the successful login goes to view the no of files in the cloud server. Each service has different set of files. This cloud server has collection of server which uniquely connected with the cloud server.

D. *File downloading*

In this scheme User uses to download the files in the cloud server. Each service has different set of files. User can collect several file by downloading, which are stored in the Cloud Server. This cloud server has collection of server cluster which uniquely connected with the cloud server

6. Performance evaluation

We analyzed the chunking algorithms based on the properties The properties of the resulting chunks from each algorithm are discussed in we present our test results related to the chunking throughput and duplicate data found in the datasets.

The performances of the following algorithms have been evaluated:

- Asymmetric extremum (AE)
- Our proposed algorithm (RAM)
- RAM with limit on maximum chunk size (RAML)
- Local maximum chunking (LMC)
- Rabin based chunking algorithm (Rabin)

As can be seen in the above list, we added RAM with a limit in the test. The purpose of adding RAM with the limit is to show the performance of RAM when a limit is applied to the maximum size of a chunk. Additionally, it also shows the improvement of RAM when the chunk variance is reduced. The performance comparison consists of three datasets. The datasets used in the tests are chosen to represent the use cases of the chunking algorithm. The first dataset is the compilation of multiple Linux distributions which have a lot of duplicate data in different locations in each file. The second dataset consists of 10 H.264 encoded videos of length 23 minutes each to simulate deduplication of media files in cloud storage. Lastly, the third dataset contains TCP dump files from to represent deduplication network traffic. The dumps contains 15 GB of data. However, we only used 9 GB of the data because of the limitation of our test system. The chunks metadata consumes a lot of memory and causes the program to stop working when the total number of chunks went over 10 million of chunks. We did not optimize the chunks management because our focus in this performance evaluation is the chunking performance.

A. *Convergent encryption technique*

A user derives a convergent key from each original data copy and encrypts the data copy with the convergent key. The key generation algorithm that maps a data copy to a convergent key. The symmetric encryption algorithm that takes both the convergent key and the data copy as inputs and then outputs a ciphertext. The decryption algorithm that takes both the ciphertext and the convergent key as inputs and then outputs the original data copy and the tag generation algorithm that maps the original data copy and outputs a tag.

B. *Component diagram*

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components. An assembly connector is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port. When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components

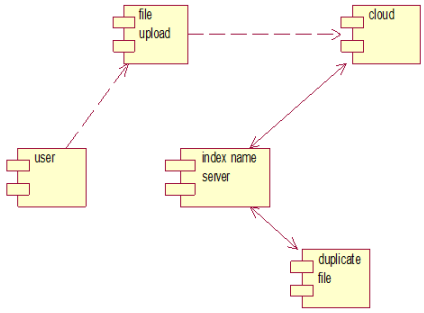

Fig. 4. Collaboration diagram

A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction. The collaboration diagram can be a decomposition of a class, class

diagram, or part of a class diagram. It can be the decomposition of a use case, use case diagram, or part of a use case diagram. The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

An object diagram in the Unified Modeling Language (UML) is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time. An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time. Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.

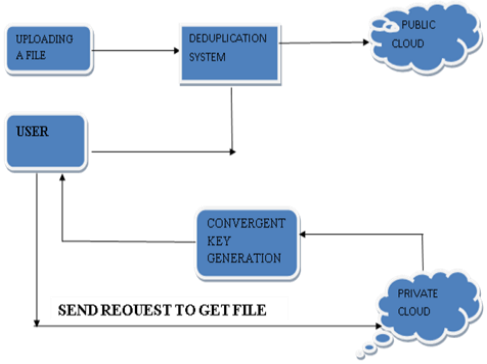

Fig. 5. System architecture

7. Application

A. *CTRL's real cloud*

The CtrlS Real Cloud has a multi-layered management model. The cloud controller server enables everything, from system architecture to VM root access, to be managed via the user interface and API. Real Cloud enables you to put up applications and manage them, all remotely and with utmost ease.

B. *Cloud layer services*

Discover the promise of cloud, not the compromises. Cloud Layer includes virtual servers, remote storage and a robust content delivery network that leverage our core advantages and longtime leadership in automated, on-demand, self-managed infrastructure.

8. Conclusion

In this paper, the notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. In which the duplicate-check tokens of files are generated by the private cloud server with private keys.

References

[1] Y.-M. Huo, H.-Y. Wang, L.-A. Hu, and H.-G. Yang, "A cloud storage architecture model for data-intensive applications," in Proc. Int. Conf. Comput. Manage., May 2011, pp. 1–4.
[2] L. B. Costa and M. Ripeanu, "Towards automating the configuration of a distributed storage system," in Proc. 11th IEEE/ACM Int. Conf. Grid Comput., Oct. 2010, pp. 201–208.
[3] H. Ohsaki, S. Watanabe, and M. Imase, "On dynamic resource management mechanism using control theoretic approach for wide-area grid computing," in Proc. IEEE Conf. Control Appl., Aug. 2005, pp. 891–897.
[4] H. Dezhi and F. Fu, "Research on self-adaptive distributed storage system," in Proc. 4th Int. Conf. Wireless Commun. Netw. Mobile Comput., Oct. 2008, pp. 1–4.
[5] J. Wang, P. Varman, and C.-S. Xie, "Avoiding performance fluctuation in cloud storage," in Proc. Int. Conf. High Performance Comput., Dec. 2008, pp. 1–9.
[6] C.-Y. Chen, K.-D. Chang, and H.-C. Chao, "Transaction pattern based anomaly detection algorithm for IP multimedia subsystem, IEEE Trans. Inform. Forensics Security, vol. 6, no. 1, pp. 152–161, Mar. 2011.
[7] G. Urdaneta, G. Pierre, and M. Van Steen, "A survey of DHT security techniques," ACM Comput. Surveys (CSUR), vol. 43, no. 2, pp. 8:1–8:49, Jan. 2011.
[8] H. He and L. Wang, "P&P: A combined push-pull model for resource monitoring in cloud computing environment," in Proc. IEEE 3rd Int. Conf. Cloud Comput., Jul. 2010, pp. 260–267.
[9] X. Sun, K. Li, and Y. Liu, "An efficient replica location method in hierarchical P2P networks," in Proc. 8th IEEE/ACIS Int. Conf. Comput. Inform. Sci., Jun. 2009, pp. 769–774.
[10] T.-Y. Wu, W.-T. Lee, and C. F. Lin, "Cloud storage performance enhancement by real-time feedback control and de-duplication," in Proc. Wireless Telecommun. Symp., Apr. 2012, pp. 1–5.