International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2019 IJSRCSEIT | Volume 5 | Issue 3 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT19539

Storage Preservation Using Big Data Based Intelligent Compression Scheme

Ramya. S[1], Gokula Krishnan. V[2]

[1]ME-Computer Science and Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamil Nadu, India

[2]Assistant Professor, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamil Nadu, India

ABSTRACT

Big data has reached a maturity that leads it into a productive phase. This means that most of the main issues with big data have been addressed to a degree that storage has become interesting for full commercial exploitation. However, concerns over data compression still prevent many users from migrating data to remote storage. Client-side data compression in particular ensures that multiple uploads of the same content only consume network bandwidth and storage space of a single upload. Compression is actively used by a number of backup providers as well as various services. Unfortunately, compressed data is pseudorandom and thus cannot be deduplicated: as a consequence, current schemes have to entirely sacrifice storage efficiency. In this system, present a scheme that permits a more fine-grained trade-off. And present a novel idea that differentiates data according to their popularity. Based on this idea, design a compression scheme that guarantees semantic storage preservation for unpopular data and provides scalable data storage and bandwidth benefits for popular data. We can implement variable data chunk similarity algorithm for analyze the chunks data and store the original data with compressed format. And also includes the encryption algorithm to secure the data. Finally, can use the backup recover system at the time of blocking and also analyze frequent login access system.

Keywords : Data Chunks, Similarity Matching, Parallel Processing, Data Security, Data Compression

I. INTRODUCTION

Now a day there is growth in information. With infinite storage space provide by cloud service provider users tend to use as much space as they can and vendors constantly look for techniques aimed to minimize redundant data and maximize space savings. Users will access information according to their needs and most users access same information again and again, the cost of computation, application hosting, content storage and delivery is reduced significantly. The cloud makes it possible for you to access your information from anywhere at any time. Cloud provides benefits such as, flexibility, disastracter, recovery, software updates automatically, pay- per-use model and cost reduction.[3] While a traditional computer setup requires you to be in the same location as your data storage device, the cloud takes away that step. The cloud removes the need for you to be in the same physical location as the hardware that stores your data. Each provider serves a specific function, giving users more or less control over their cloud depending on the type. Your cloud needs will vary depending on how you intend to use the space and resources associated with the cloud. Cloud computing refers to the use of computers which access Internet locations for computing power, storage and applications, with no need for the individual access points to maintain any of the infrastructure. Data deduplication is a technique for

reducing the amount of storage space an organization needs to save its data. In most organizations, the storage systems contain duplicate copies of many pieces of data. For example, the same file may be saved in several different places by different users, or two or more files that aren't identical may still include much of the same data. Along with low ownership costs and flexibility, users require the protection of their data and confidentiality guarantees through encryption. To make data management scalable deduplication we are use Encryption for secure deduplication services.[1] Unfortunately, deduplication and encryption are two conflicting technologies. While the aim of deduplication is to detect identical data segments and store them only once, the result of encryption is to make two identical data segments in distinguishable after being encrypted. This means that if data are encrypted by users in a standard way as like shared authority, the cloud storage provider cannot apply deduplication since two identical data segments will be different after encryption. On the other hand, if data are not encrypted by users, confidentiality by cannot be guaranteed and data are not protected against curious cloud storage providers. There are two types of deduplication in terms of the size: (i) file-level deduplication, which discovers redundancies between different files and removes these redundancies to reduce capacity demands, and (ii) blocklevel deduplication, which discovers and removes redundancies between data blocks. The file can be divided into smaller fixed-size or variable-size blocks. Using fixedsize blocks simplifies the computations of block boundaries, while using variable-size blocks. A technique which has been proposed to meet these two conflicting requirements is Tag generation and AES Scheme whereby the encryption key is usually the result of the hash of the data segment. Although encryption seems to be a good candidate to achieve confidentiality and deduplication at the same time, it unfortunately suffers from various well-known weaknesses. The confidentiality issue can be handled by encrypting sensitive data before outsourcing to remote servers. Along with low ownership costs and flexibility, users require the protection of their data and confidentiality guarantees through encryption. In this paper, we address the a for mentioned privacy issue to propose a shared authority to the files which Deduplicted based privacy preserving authentication for the cloud data storage, which realizes authentication and authorization without compromising a user's private information. The basic data chunk similarity is shown in fig 1.

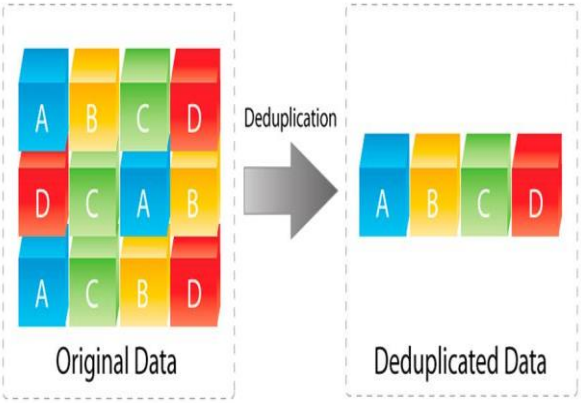

Fig 1 : Data chunk similarity

II. RELATED WORK

L. Wang,et.al,…[1] proposed an innovative public cloud usage model for small-to medium scale scientific communities to utilize elastic resources on a public cloud site while maintaining their flexible system controls, i.e., create, activate, suspend, resume, deactivate, and destroy their high-level management entities—service management layers without knowing the details of management. Second, we design and implement an innovative system—DawningCloud, at the core of which are lightweight service management layers running on top of a common management service framework. The common management service framework of DawningCloud not only facilitates building lightweight service management layers for

heterogeneous workloads, but also makes their management tasks simple. Third, we evaluate the systems comprehensively using both emulation and real experiments.

B. Li,et.al,…[2] took a step towards bringing the many benefits of the MapReduce model to incremental one-pass analytics. In the new model, the MapReduce system reads input data only once, performs incremental processing as more data is read, and utilizes system resources efficiently to achieve high performance and scalability. The goal is to design a platform to support such scalable, incremental one-pass analytics. This platform can be used to support interactive data analysis, which may involve online aggregation with early approximate answers, and, in the future, stream query processing, which provides near real-time insights as new data arrives. We argue that, in order to support incremental one-pass analytics, a MapReduce system should avoid any blocking operations and also computational and I/O bottlenecks that prevent data from "smoothly" flowing through map and reduce phases on the processing pipeline.

R. Kienzler,et.al,…[3] propose an incremental data access and processing approach for data-intensive cloud applications that can hide data transfer latencies while maintaining linear scalability. Similar in spirit to pipelined query evaluation in traditional database systems, data is accessed and processed in small increments, thereby propagating data chunks from one stage of the data analysis task to another as soon as they are available instead of waiting until the whole dataset becomes available. This way we can process data mostly in memory (hence, reduce time-consuming I/O to local disk and cloud storage, and avoid storage costs) as well as achieving pipelined parallelism (in addition to the existing partitioned parallelism), leading to a reduction in overall task completion time.

C. Olston,et.al,…[4] proposed a system for Building and updating a search index from a stream of crawled web pages. Some of the numerous steps are deduplication, link analysis for spam and quality classification, joining with click-based popularity measurements, and document inversion. Processing semi-structured data feeds, e.g. news and (micro-)blogs. Steps include de-duplication, geographic location resolution, and named entity recognition. Processing along these lines is increasingly carried out on a new generation of flexible and scalable data management platforms, such as Pig/Hadoop. Hadoop is a scalable, fault-tolerant system for running individual map-reduce processing operations over unstructured data files. Pig adds higher-level, structured abstractions for data and processing. Despite the success of Pig/Hadoop, it is becoming apparent that a new, higher, layer is needed: a workow manager that deals with a graph of interconnected Pig Latin programs, with data passed among them in a continuous fashion. Given that Pig itself deals with graphs of interconnected data processing steps, it is natural to ask why one would layer another graph abstraction on top of Pig.

K.H. Lee, et.al,…[5] implemented The programming model is inspired by the map and reduces primitives found in Lisp and other functional languages. Before developing the MapReduce framework, Google used hundreds of separate implementations to process and compute large datasets. Most of the computations were relatively simple, but the input data was often very large. Hence the computations needed to be distributed across hundreds of computers in order to finish calculations in a reasonable time. MapReduce is highly efficient and scalable, and thus can be used to process huge datasets. When the MapReduce framework was introduced, Google completely rewrote its web search indexing system to use the new programming model. The indexing system produces the data structures used by Google web search. The parallelization doesn't necessarily have to

be performed over many machines in a network. There are different implementations of MapReduce for parallelizing computing in different environments. Phoenix is an implementation of MapReduce, which is aimed at shared-memory, multi-core and multiprocessor systems, i.e. single computers with many processor cores.

data deduplication types

In this chapter, we can discuss the various data deduplication types as follows

File-level de-duplication

It is commonly known as single-instance storage, file-level data de-duplication compares a file that has to be archived or backup that has already been stored by checking all its attributes against the index. The index is updated and stored only if the file is unique, if not than only a pointer to the existing file that is stored references. Only the single instance of file is saved in the result and relevant copies are replaced by "stub" which points to the original file.

Block-level de-duplication

Block-level data deduplication operates on the basis of sub-file level. As the name implies, that the file is being broken into segments blocks or chunks that will be examined for previously stored information vs redundancy. The popular approach to determine redundant data is by assigning identifier to chunk of data, by using hash algorithm for example it generates a unique ID to that particular block. The particular unique Id will be compared with the central index. In case the ID is already present, then it represents that before only the data is processed and stored before .Due to this only a pointer reference is saved in the location of previously stored data. If the ID is new and does not exist, then that block is unique. After storing the unique chunk the unique ID is updated into the Index. There is change in size of chunk as per the vendor. Some will have fixed block sizes, while some others use variable block sizes

Variable block level de-duplication

It compares varying sizes of data blocks that can reduce the chances of collision, stated Data links in cloud framework. The difference between deduplication schemes are shown in fig 2.

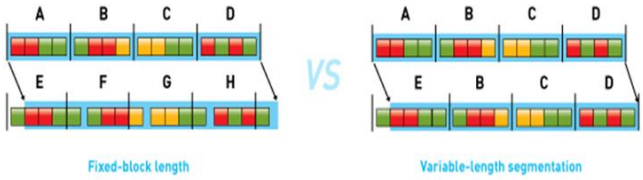

Fig 2 : Deduplication Schemes

Secure Deduplication Algorithms

The main objective of this paper to analyze various encryption algorithms with deduplication schemes. The basic algorithms are shown as follows:

Traditional Encryption algorithm:

Although it is known that data deduplication gives more benefits, security and privacy concerns arise because the users sensitive data is susceptible to both the outsider as well as insider attacks. So, while considering the traditional encryption techniques to secure the users sensitive data there are many issues are associated. Traditional encryption provides data confidentiality but it is not compatible with deduplication. As in traditional encryption different users encrypt their data with their own keys. Thus, the identical data of the different users will lead to different ciphertext which is making the data deduplication almost impossible in this traditional approach. The basic step of the algorithm as shows:
KeyGenSE: k is the key generation algorithm that generates κ using security parameter I

EncSE (k, M): C is the symmetric encryption algorithm that takes the secret κ and message M and then outputs the ciphertext C;

DecSE (k, C): M is the symmetric decryption algorithm that takes the secret κ and ciphertext C and then outputs the original message M.

Convergent Encryption algorithm: The convergent encryption techniques are those which provide the data confidentiality to the users outsourced data stored on the public clouds. These techniques while providing the confidentiality to the data are also compatible with the data deduplication process. In this algorithm the encryption key is itself derived from the message. So it supports data deduplication also, because the same file will give the same encryption key so it will generate the same ciphertext irrespective of users which makes data deduplication possible.

KeyGenCE(M) → K is the key generation algorithm that maps a data copy M to a convergent key K;
EncCE(K,M) → C is the symmetric encryption algorithm that takes both the convergent key K and the data copy M as inputs and then outputs a ciphertext C;
DecCE(K,C) → M is the decryption algorithm that takes both the ciphertext C and the convergent key K as inputs and then outputs the original data copy M; and
TagGen (M) → T (M) is the tag generation algorithm that maps the original data copy M and outputs a tag T (M).

Block cipher algorithm:
In cryptography, a block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key. Block ciphers operate as important elementary components in the design of many cryptographic protocols, and are widely used to implement encryption of bulk data. Iterated product ciphers carry out encryption in multiple rounds, each of which uses a different sub key derived from the original key. One widespread implementation of such ciphers is notably implemented in the DES cipher. Many other realizations of block ciphers, such as the AES, are classified as substitution-permutation networks. The publication of the DES cipher was fundamental in the public understanding of modern block cipher design. It also influenced the academic development of cryptanalytic attacks. Both differential and linear cryptanalysis arose out of studies on the DES design. There is a palette of attack techniques against which a block cipher must be secure, in addition to being robust against brute force attacks. Even a secure block cipher is suitable only for the encryption of a single block under a fixed key. A multitude of modes of operation have been designed to allow their repeated use in a secure way, commonly to achieve the security goals of confidentiality and authenticity. However, block ciphers may also feature as building-blocks in other cryptographic protocols, such as universal hash functions and pseudo-random number generators.

One important type of iterated block cipher known as a substitution-permutation network (SPN) takes a block of the plaintext and the key as inputs, and applies several alternating rounds consisting of a substitution stage followed by a permutation stage—to produce each block of cipher text output. The non-linear substitution stage mixes the key bits with those of the plaintext, creating Shannon's confusion. The linear permutation stage then dissipates redundancies, creating diffusion. A substitution box (S-box) substitutes a small block of input bits with another block of output bits. This substitution must be one-to-one, to ensure invertibility (hence decryption). A secure S-box will have the property that changing one input bit will change about half of the output bits on average, exhibiting what is known as the avalanche effect—i.e. it has the property that each output bit will depend on every input bit.

Variable chunk similarity:

It requires more processing power than the file deduplication, since the number of identifiers that need to be processed increases greatly. Correspondingly, its index for tracking the individual iterations gets also much larger. Using of variable length blocks is even more source-intensive. Moreover, sometimes the same hash number may be generated for two different data fragments, which is called hash collisions. If that happens, the system will not save the new data as it sees that the hash number already exists in the index. The algorithm steps as follows.

BlockTag(FileBlock) - It computes hash of the File block as file block Tag;

DupCheckReq(Token) - It requests the Storage Server for Duplicate Check of the file block.
FileUploadReq(FileBlockID, FileBlock, Token) – It uploads the File Data to the Storage Server if the file block is Unique and updates the file block Token stored.

FileBlock Encrypt(Fileblock) - It encrypts the file block with Convergent Encryption, where the convergent key is from SHA Hashing of the file block;
TokenGen(File Block, UserID) – the process loads the associated privilege keys of the user and generate token.

FileBlockStore(FileBlockID, FileBlock, Token) - It stores the FileBlock on Disk and updates the Mapping. The variable chunk similarity level deduplication is shown in fig 3.

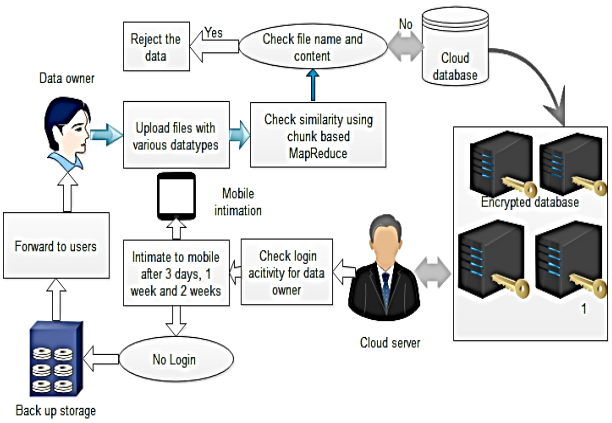

Fig 3 : Variable chunk similarity backup server

The mathematical model as follows:
Let S be the system object.
It consist of following S={U,F,CSP}
U= no of users U={u1,u2,u3,…..un}
F= no of files F={f1,f2,f3,…..fn}
B=no of blocks. B{B1,B2,…,Bn}
CSP={C,PF,V,POW}
C=challenge
PF =proof by CSP
V= verification by TPA
POW= proof of ownership
CSP= Cloud Service provider
CSP={PF,F} PF=proof F=files

COMPRESSION SCHEME

LZW compression is the compression of a file into a smaller file using a table-based lookup algorithm

- Takes each input sequence of bits of a given length and creates an entry in a table for that particular bit pattern, consisting of the pattern itself and a shorter code.
- As input is read, any pattern that has been read before results in the substitution of the shorter code, effectively compressing the total amount of input to something smaller.

Psuedo code for variable data compression:

Step 1: User profiling: Client registration and log-in
Step 2: Client initiates file transfer (upload/download).
Step 3: File-upload: check for duplicate
Step 4: If duplicate at any of file name and file content, create file pointer and store in CSP
Step 5: If no duplicate found, compress the file.
Step 6: File-download: Data owner to decompress and download file.
Step 7: Server sync with client and completes file upload/download process
Step 8: Check Login time of user after 3 days, 1 week, 2 week and 3 Weeks
Step 9: If no login means, send mobile intimation to user
Step 10: Recover the files and forward to alternative mail
The proposed work modules are listed as follows

FRAMEWORK CONSTRUCTION

Big data and storage solutions provide users and enterprises with various capabilities to store and process their data in either privately owned, or third-party data centers that may be located far from the user–ranging in distance from across a city to across the world. Big data storage system relies on sharing of resources to achieve coherence. In this framework, we can have two types of users such as Gmail user and Gmail server. The person or organization that legally owns a service is called a Gmail server. The Gmail user can be the consumer, or the user that owns the storage within which the Gmail service resides. Service provider provides the storage space to the users. Storage space can be shared by multiple data owners. Data owners can be upload the files in storage system for future use.

FILE COMPRESSION

In this module, Data owner files are read by server. Server can generate the dictionary to predict the redundant data. Update the index files to eliminate the redundant files to preserve the storage. A particular LZW compression algorithm takes each input sequence of bits of a given length (for example, 12 bits) and creates an entry in a table (sometimes called a "dictionary" or "codebook") for that particular bit pattern, consisting of the pattern itself and a shorter code. As input is read, any pattern that has been read before results in the substitution of the shorter code, effectively compressing the total amount of input to something smaller. Unlike earlier approaches, known as LZ77 and LZ78, the LZW algorithm does include the look-up table of codes as part of the compressed file. The decoding program that uncompresses the file is able to build the table itself by using the algorithm as it processes the encoded input.

FILE ENCRYPTION:

Encryption is the most effective way to achieve data security. To read an encrypted file, you must have access to a secret key or password that enables you to decrypt it. Unencrypted data is called plain text ; encrypted data is referred to as cipher text. There are two main types of encryption: asymmetric encryption (also called public-key encryption) and symmetric encryption. We can implement symmetric encryption for encrypt the data files using single key approach. Symmetric key algorithms are algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of cipher text. The keys may be identical or there may be a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. Encrypted data can be stored in cloud server.

SIMILARITY CHECKING

In computing, data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Related and somewhat

synonymous terms are intelligent (data) compression and single-instance (data) storage. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the compression process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. In this module, we can check the files using file name with file contents. The uploaded files are spilted into chunks. Service provider checks the chunks at the time of uploading files. Data owner only upload original file so save storage space in cloud system. We can compression in text file, document file and image files and video files

ALERT SYSTEM

In this system, we can design application for alert system for every week. After four weeks completed, if there is no access means the files are automatically sent to alternate mail and mobile which are stored at the time of registration. Server can save huge amount of storage and provide to other users.

BACKUP RECOVERY APPROACH

Admin can check access time for each user login. If user login to the system means, activity is registered in storage. And also monitor each user access. If the user access is paused more than 3 days means, admin automatically send alert to user based on registered mobile numbers. Finally if there is no access in storage system means, backup is generated. And flush the storage space and save storage for server for future use.

III. CONCLUSION

We proposed the distributed compression systems to improve the reliability of data while achieving the confidentiality of the users and also shared authority outsourced data with an encryption mechanism. Four constructions were proposed to support file-level and block-level data compression. The security of tag consistency and integrity were achieved. We implemented our compression systems using the secret sharing scheme and demonstrated that it incurs small encoding/decoding overhead compared to the network transmission overhead in regular upload/download operations. In this work, we have identified a new privacy challenge during data accessing in the cloud computing to achieve privacy-preserving access authority sharing for similarity files. Authentication is established to guarantee data confidentiality and data integrity. Data anonymity is achieved since the wrapped values are exchanged during transmission. User privacy is enhanced by access requests to privately inform the cloud server about the users access desires. The backup recovery scheme is to improve the recovered scheme to avoid the blockages and also refund the amount to unused spaces in cloud system.

IV. REFERENCES

[1]. L. Wang, J. Zhan, W. Shi and Y. Liang, “In cloud, can scientific communities benefit from the economies of scale?” IEEE Transactions on Parallel and Distributed Systems 23(2): 296-303, 2012.

[2]. B. Li, E. Mazur, Y. Diao, A. McGregor and P. Shenoy, “A platform for scalable one-pass analytics using mapreduce,” in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), 2011, pp. 985-996.

[3]. R. Kienzler, R. Bruggmann, A. Ranganathan and N. Tatbul, “Stream as you go: The case for incremental data access and processing in the cloud,” IEEE ICDE International Workshop on Data Management in the Cloud (DMC'12), 2012

[4]. C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V.B.N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell and X. Wang, "Nova: Continuous pig/hadoop workflows," Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), pp. 1081-1090, 2011.

[5]. K.H. Lee, Y.J. Lee, H. Choi, Y.D. Chung and B. Moon, "Parallel data processing with mapreduce: A survey," ACM SIGMOD Record 40(4): 11-20, 2012.

[6]. X. Zhang, C. Liu, S. Nepal and J. Chen, "An Efficient Quasiidentifier Index based Approach for Privacy Preservation over Incremental Data Sets on Cloud," Journal of Computer and System Sciences (JCSS), 79(5): 542-555, 2013.

[7]. X. Zhang, T. Yang, C. Liu and J. Chen, "A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization using Systems, in MapReduce on Cloud," IEEE Transactions on Parallel and Distributed, 25(2): 363-373, 2014.

[8]. N. Laptev, K. Zeng and C. Zaniolo, "Very fast estimation for result and accuracy of big data analytics: The EARL system," Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE), pp. 1296-1299, 2013.

[9]. T. Condie, P. Mineiro, N. Polyzotis and M. Weimer, "Machine learning on Big Data," Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE), pp. 1242-1244, 2013.

[10]. Aboulnaga and S. Babu, "Workload management for Big Data analytics," Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE), pp. 1249, 2013

Cite this article as :

Ramya. S, Gokula Krishnan. V, "Storage Preservation Using Big Data Based Intelligent Compression Scheme", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 5 Issue 3, pp. 92-100, May-June 2019. Available at doi : https://doi.org/10.32628/CSEIT19539
Journal URL : http://ijsrcseit.com/CSEIT19539