IJCSET – Volume 3, Issue 3 –MARCH 2017 ISSN: 2455-9091 Pages: 1-6

International Journal of Computer Science Engineering & Technology
APPROVED BY
NATIONAL SCIENCE LIBRARY (NSL)
NATIONAL INSTITUTE OF SCIENCE COMMUNICATION AND INFORMATION RESOURCES (NISCAIR)
COUNCIL OF SCIENTIFIC AND INDUSTRIAL RESEARCH (CSIR)- NEW DELHI INDIA .
ISSN :2455-9091

# CLOUD STORAGE OPTIMIZATION USING EFFICIENT CODING APPROACH

[1] M.DHINESH KUMAR, [2] U. GOKUL KANNAN, [3] M.ASHOK KANNA, [4] P. ANATHA PRABHA
[1, 2, 3, 4] DEPARTMENT OF CSE
[1, 2, 3, 4] SRI KRISHNA COLLEGE OF TECHNOLOGY

**ABSTRACT:** To minimize storage overhead, cloud file systems are transitioning from replication to erasure requirements. Cloud storage generally provides different redundancy configuration to users in order to maintain the required the balance between performance and threshold. Data availability is critical in distributed storage systems, especially when node failures are prevalent in real world. This kind of process has revealed new dimensions on which to evaluate the performance of different coding schemes. This paper introduces a novel cloud storage management which optimally combines storage space resources from multiple providers. Redundancy, security and other impractical properties can be adjusted adequately to the needs of the storage space service consumer. Furthermore, this paper presents an user friendly storage controller implementation with adaptable overhead which operates on and integrates into typical consumer environments as a central part of an overall storage system. Its optimality claims are validated in real-world situations with several commercial on the internet and cloud storage providers.

## 1. INTRODUCTION

Cloud computing has gained tremendous popularity in recent years. One of its main features is how easy it is to increase computing power at a moment's notice, and on the other hand to give it up when it's not needed anymore. Little spare capacity should beheld around for times during the high resource usage, lyingnonproductive in between. Only the resources used are paid for. These cloud resources come by means of virtual machines, which comein most sizes, ranging from less powerful than a cheap smart phone to monsters with dozens of cores and hundreds of gigabytes of memory. Depending on circumstances resources can be modified either by modifying specific virtual machines with the addition of or removing cores or memory space (vertical scaling) or by spinning up or down entire machines (horizontal scaling). The reason for the limitation is the traditional maxim of storage systems, no data mustat any time be lost. According to this promise,all bytes written to everlastingstorage are of equal and extremely important. However, modern applications don't always behave determined by this principle. They use durable storage media for increased performance, rather than durability. This shift is driven by hardware developments: Larger disks allow softwaredevelopers to think of creative waysto work with the extra space, sturdy state drives can function as a cheaper option to RAM. As a Consequently, much of the data stored by applications on supplementary storage is volatile data it does not fit inRAM

MEMORY; usually, it might be thrown away on a reboot (e. g., swap files), reconstructed via computation over other data (e. g., more advanced MapReduce or Dryad data files, image thumbnails, memorized results of computations, desktop search indices, and inflated types of compressed files), or fetched over the network from all other systems (e. g., browser and package management caches). In addition, strength|may well not be complex for some data either because new applications(such as big data analytics) provides useful answers despite lacking data, or because the data may be replicatedacross multiple locations. This kind of project exposed an ineffectivedynamic between storage systems and applications: Applications opportunistically use any spaceavailable for nonessential data, as the storage systems struggle to make certainnone of the data is lost. The applications do this for valid reasons, to enhance their own efficiency. However, there is certainly little or no cooperation between applications and hard devices fill up with data. This paper, therefore like to build a system that has more oversight over data on the system. When free space is scattered the device should be able to identify less important data and remove it.However, the system should be able to go in the other direction and reconstruct removed data when space becomes plentiful.

## 2. LITERATURE SURVEY

One of the biggest challenges in developing storage systems providing the reliability and availability that users expect. Once their data is stored, users expect it to be persistent forever, and constantly available. Unfortunately, in practice there are quantity of problems that, if not handled, can cause data loss in storage space systems. One primary cause of data loss is disk drive unreliability. Distinct combinations of matrices and schedules, there is a huge gap in the number of XOR operations, and no oneblend performs best for all redundancy configurations.The amount of data necessary for recovery and degradedsays, may limit performance more than CPU overhead. Encoding as well as decoding in Caco, calculation of delay in network performance overhead.

L. N. Bairavasundaram et al[1]., 2008 proposed and analyzed that an important threat to effective storage space of data is noiseless data corruption. To be able to develop desired protection mechanisms against data corruption, one will need to understand its characteristics based on the error and eliminate noiseless data corruption to avoid data storage corruption. It includes checksum mismatches within the samedisks and also across different disks which leads to data corruption. J. L. Hafner et al[2]., 2008 proposed that a suitable desiredunit provides the capability to analyze the danger posed by Undetected Disk Errors(UDE) in specific system and the threshold coverage provided by recommended mitigation techniques. An easyconditional model could fail to account for important emergent trends in UDE outward exhibition for a givenwork load. D. Ford et al[3]., 2010 proposed thatquantifying the impact of failures and prioritizing hardware and software subsystem improvements in datacenter environment avails better availability of distributed system.B. Calder et al [4]., 2011provides cloud storage space in the form of Blobs (user files), Tables(structured storage), and Lines (message delivery). These three data abstractions provide the overall storage and work for many applications with greater effectivity.J. Kubiatowiczet al[5]., 2000envisions a supportive utility model in which consumers pay a regular monthlycost in return for gain access to persistent storage. Many of these utility should be highly-available from anywhere in the network, and it employsprogrammed replication for disaster restoration.B. Fan et al[6]., 2009 proposedan adjustment of the Hadoop distributed file system (HDFS) to asynchronously replace multiple copies of information blocks with RAID 5 and RAID 6 encodings. Because this replacement is asynchronous, it can be delayed where ever spare capacity is available. K. D. Bowers et al[7]., 2009 proposed a distributed cryptographic system that permits some collection ofweb servers to prove to a customer that a stored record is intact and retrievable. A High-availability and integrity layer (HAIL) strengthens, formally unifies, and streamlines distinct techniques from the cryptographic and distributed-systems communities to provide integrity to greater availability.

## 3. PROPOSED SYSTEM

Cloud storage systems always use different redundancy configurations, depending on the desired balance between performance and fault tolerance. This paper presents an improvement to Cauchy maintenance based on Hilbert Curve Algorithm and for finding duplicate redundancy based on Rabin Hash Function coding. Perform degraded reads more efficiently than all known codes, but otherwise inherit the reliability and performance properties of Hilbert Curve codes. This approach is desired to be able to identify the optimal coding scheme in the current state of the art, for an arbitrary given redundancy configuration. The design of CaCo not only proves efficacious in theimprovement of performance, but also has good scalability.First, CaCo incorporates all existing matrix and scheduleheuristics so that it is able to discover a comparable solution.Compared to the enumeration algorithm, CaCo hasmuch lower complexity. Second, if there are new heuristicsfor generating Cauchy matrices or scheduling to be derivedin the future, it can be easily added to CaCo. If some othermatrices are later found better for scheduling, it can be addedto the set.Our approach helps in reducing the cost of the cloud computing by making the predictions.

### 3.1 File Uploading

This module is helping to register the user details on the application. In registration get username, email address, password, etc. Each user needs the login credential to develop a web interface to upload and download files in cloud storage. The different file uploading links are open. The user can choose the link which they want to upload on cloud. User can upload the file on cloud such as doc file, video, mp3, etc. The user can upload the file after the login of the application. User with help of the server to view details and upload files with the secured manner.

### 3.2 Block SplittingOf File

A block is a fragment of information, each chunkshad a header which indicates some parameters (e.g. the type of chunk, comments, size etc.) In the middle there is a variable area containing data which are decoded by the program from the parameters in the header. Chunks may also be fragments details which are downloaded or managed by P2P programs. In distributed computing, a chunk is a collection of data which are delivered to a processor or one of the parts of |your computer for control.After uploading the data files in the applying, clients store their data at the remote server, they can dynamically update their data at later times. This kind of system splits the records in several portions then encodes and stores it on different cloud. Meta data necessary for decrypting and moving a file will be stored in metadata management server. File can club with another record. At the block level, the key functions are block attachment, block modification and block out deletion.

### 3. 3 Redundancy Check

Redundancy check(RC), which is in common use currently. This method can discover and appropriate errors in sequences of bits and can therefore be used in data transmission as well as in data storage to protect files from problems. Cauchy Redundancy Check is in utilization in many applications and specifications, and is easily included in hardware. The method can be described by polynomial division, but since its property seem to be obscure to a lot ofpeople will give a explanations of its properties and exactly how it can demonstrate them. Absolute goal of this paper is to expose a new and simple theory to characterize the types of errors that can be detected or corrected by this method.

### 3.4 Cache File Maintenance

Reed-Solomon codes are based on a limited field, often called Galois field. When encoding data using RS codes, to implement a Galois submitted arithmetic procedure (addition or multiplication)requires many computations, so the performance is often unsatisfactory. Whereas Cauchy Reed-Solomon codesimprove RS codes and give two improvements. First, CRSuses a Cauchy matrix rather than a Vander monde matrix. Second requirements convert Galois field copies into XOR operations. In a storage system using erasure codes, data are encoded to obtain data redundancy when data are written. Therefore, to increase the overall efficiency of a system, one should reduce the expense of erasure coding, i.e., the number of XOR operations. it is impractical to enumerate them. Therefore it is challenging to determine which one of the matrices will produce better schedules. In the CaCoapproach, it chooses only a certain number of them for scheduling.

## 4. METHODOLOGY

### Hilbert Curve Algorithm

Direct volume rendering is concerned with the visualization of volumetric datasets, described as 3D or 4D arrays of scalar or vector (RGBA) values. These types of datasets are being ubiquitously produced by medical scanners, scientific simulations, and engineering design. They even occur in mainstream entertainment applications, for example in the technique of amorphous effects, such as smoke, gas, or fluid. A number of paradigms have been described for the volume rendering of regular grids, with the most popular being ray casting, splatting, shear-warp, and cell projection. These methods have received significant attention and, due to this widespread effort, have been well developed at this point. While all of these algorithms have their strengths and weaknesses, ray-based approaches provide the highest flexibility to conveniently realize non-linear light propagation effects, as occurring in refracting and diffracting media. The modeling of global scattering and shadows are likewise best handled with ray-based approaches. Our cloaking algorithm consists of two steps; cloaking region generation by the Hilbert value and expansion cell selection by considering the locality. The algorithm iterates above two steps until satisfying k-anonymity.

### Definition 1. Grid cells ordered by Hilbert curve

The set of cells ordered by Hilbert curve is defined as H = {C00, C01, C02, ... Cij, ..., C(N-1)(N 1)}, where i and j are the (x,y)-coordinates of a grid and N is the number of grid cells in one dimension. Next, it stores the information of adjacent cells being not connected by Hilbert curve. Because the number of objects is required to store along with the adjacent cells' information, it searches neighbor cells for each cell. If the difference of Hilbert value is greater than '1', it inserts the neighbor cell into our data structure. The definition of the adjacent cell is below.

### Definition 2. Adjacent cell without sequential Hilbert curve's order

The set of adjacent cells being not connected by Hilbert curve is defined as

$AC = \{ ac_{ij} \| ac_{ij} - c_{xy} | > 1, ^{¥}ac_{ij}, ^{¥}c_{xy} € H\}$

Where $0 \leq i,j \leq N$ and $i = x \pm 1$, $j = y \pm 1$

By using our data structure, it can reduce a cloaking area generated and can decrease processing time for finding k-1 users.

```
Algorithm Cloaking (int cId, int k, int threshold)
Input: cId //cell identifier which user located
k // k anonymity value
threshold // threshold for retrieve adjacent cell
Output: CR(Cloaking Region)
1. Find the Cell Ci with cId
2. do
3. Retrieve Ci
4. if(Ci->count >0)
5. add Ci to CR and Count+=Ci ->count
6. if(Count >= k) return Cover(CR)
7. Retrieve the set HC={Ci-T,Ci-T+1, ..., Ci-1, Ci+1, ..., Ci+T}
8. for each hci of the set HC
9. if(Ci->count > 0)
10. add hci to CR and Count+=hci->count
11. if(Count >=k) return CR
12. Insert the set AC of hci into PQ //PQ: Priority
13. Remove Ci from the Priority Queue
14. while(Ci = NULL)
        End Algorithm
```

## 5. EXPERIMENTAL RESULT

In this paper, the system explored the performance of the popular hash operate function ofHilbert curve's Algorithm. Then it checks the algorithm's quality in two aspects: procedure and area complexity. Then it checks the protection aspects of Hilbert curve's Algorithm. The output result of the process is defined below in the table and in figure 1.

| Techniques | Accuracy | Throughput | Delay |
|---|---|---|---|
| SHA | 78% | 80% | 70% |
| Rabin | 82% | 86% | 62% |
| Hilbert Curve | 90% | 92.3% | 51% |

**Table - 1 Performance measure table**

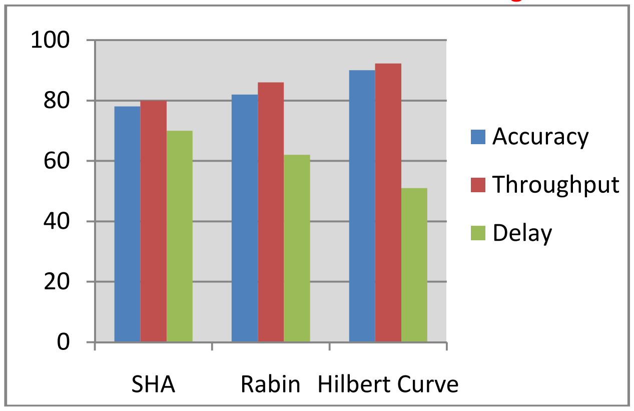

**Figure - 1 Performance Graph**

In Figure 1, SHA performance is low when compared to rabin and Hilbert curve. In the all, Hilbert curve has the greater accuracy, throughput and lesser delay with better result. But with integrity and consideration of all this approach it produces balanced performance and greater stability of results.

## CONCLUSION

A systematic procedure for obtaining optimality in cloud storage services along the provider's and customer's iterations of the service lifecycle. Constructionafter our analysis of existing distributed cloud storage space techniques and theweaknesses, it creates an even moreuniversal and extensible architecture which serves as blueprint for building optimal cloud storage space controllers encompassing a superset of the main existing features. In this paper, it presents an improvement to Cauchy maintenance based on Hilbert Curve Algorithm and for finding duplicate redundancy based on Rabin Hash Function coding. This strategy isplanned to be able to identify the optimumcoding scheme.

## REFERENCES

[1] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci- Dusseau, G. R. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," Trans. Storage, vol. 4, pp. 8:1–8:28, Nov. 2008.

[2] J. L. Hafner, V. Deenadhayalan, W. Belluomini, and K. Rao, "Undetected disk errors in raid arrays," IBM J. Res. Develop., vol. 52, pp. 413–425, Jul. 2008.
[3] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in Proc. 9th USENIX Symp. Oper. Syst. Des. Implementation, 2010, pp. 61–74.
[4] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. F. u. Haq, M. I. u. Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas, "Windows Azure storage: A highly available cloud storage service with strong consistency," in Proc. 23rd ACM Symp. Oper. Syst. Principles, New York, NY, USA, 2011, pp. 143–157.
[5] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," SIGPLAN Notices, vol. 35, pp. 190–201, Nov. 2000.
[6] B. Fan, W. Tantisiriroj, L. Xiao, and G. Gibson, "Diskreduce: Raid for data-intensive scalable computing," in Proc. 4th Annu. Workshop Petascale Data Storage, New York, NY, USA, 2009, pp. 6–10.
[7] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in Proc. 16th ACM Conf. Comput. Commun. Security, New York, NY, USA, 2009,
pp. 187–198.
[8] G. Xu, F. Wang, H. Zhang, and J. Li, "Redundant data composition of peers in p2p streaming systems using Cauchy Reed-Solomon codes," in Proc. 6th Int. Conf. Fuzzy Syst. Knowl. Discovery Volume 2, Piscataway, NJ, USA, 2009, pp. 499–503.
[9] J. S. Plank, "A tutorial on Reed-Solomon coding for faulttolerance in raid-like systems," Softw. Pract. Experience, vol. 27, pp. 995–1012, Sept. 1997.
[10] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, 1960.
[11] J. Bloomer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, and D. Zuckerman, "An xor-based erasure-resilient coding scheme," International Computer Science Institute, Berkeley, California, USA, 1995.
[12] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," in Proc. 7th Conf. File Storage Technol., Berkeley, CA, USA, 2009, pp. 253–265.
[13] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in raid architectures," IEEE Trans. Comput., vol. 44, no. 2, pp. 192–202, Feb. 1995.
[14] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in Proc. 3rd USENIX Conf. File Storage Technol., Berkeley, CA, USA, 2004, pp. 1–1.
[15] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," IEEE Trans. Inform. Theory, vol. 45, no. 1, pp. 272–276, Jan. 1999.
[16] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-code: A new raid-6 code with optimal properties," in Proc. 23rd Int. Conf. Supercomput., New York, NY, USA, 2009, pp. 360–369.