I. J. Computer Network and Information Security, **2015, 8, 30-38**
Published Online July 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2015.08.04

An Extended Approach for Efficient Data Storage in Cloud Computing Environment

Fatemeh shieh
Department of Computer Engineering, Mahallat Branch, Islamic Azad University, Mahallat, Iran
Email: Fatemeh.shieh@gmial.com

Mostafa Ghobaei Arani
Department of Computer Engineering, Parand Branch, Islamic Azad University, Tehran, Iran
Email: mostafaghobaei@piau.ac.ir

Mahboubeh Shamsi
Department of Computer Engineering, Qom Branch, University Of Technology Qom,Iran
Email: shamsi@qut.ac.ir

Abstract—In recent years, the advent of online data storage services has been enabled users to save their data and operational programs in cloud databases. Using an efficient and intelligent management helps to optimize quality of provided services. Also it is possible to increase throughput of services by eliminating repeated data. In following article we have offered a completely dynamic approach to detect and eliminate duplicated data which exist in shared storage resources among virtual machines. Results of simulation show that proposed approach, compared to the similar approaches, will save the storage space substantially by reducing usage of CPU, RAM, also will increase rate of de-duplication data up to 23 %.

Index Terms—Cloud computing, Virtual machine, Data storage system, De-duplication.

I. Introduction

Cloud computing technology is currently one of the popular and developing technologies and a successful example of distributed computing. Cloud computing is a model for an easy provision of network access, based on demand, for a shared storage of configurable computing resources (i.e. networks, servers, applications, services, etc.), which is capable of being provided and released very quickly with minimal management efforts, and minimal interaction with the service provider [1,2].

Changing business requirements and outburst of digital data have been launched huge demands for efficient high volume data storage. Due to the limited financial resources and the increasing cost of storing electronic data, people often tend to storage their data in the context of cloud [3].

Cloud Computing technology enables users to transfer their operational data and programs to the web then operate the programs without commitment to have any special physical infrastructure [4,5]. All cloud services that have been offered, allow their users to halt problems using two important aspects of Dependability and Elasticity. One other important aspect is the use of virtualization technology through cloud services [4,6,7,8]. Virtual machines (VMs) make possible to increase services, transferring applications in cloud. It is easier and faster to deploy a new virtual machine or move it to another physical server in the comparison of deploying a new physical server. Virtualization also makes possible having more control over cloud resources such as disk, network and computing power. Hence, these resources can be distributed in accordance to the requirements of the applications. Using virtual machines is an important factor to achieve elasticity. Both Cloud services and Online support services access lots of data which are required continuously to storage data, consequently, a large number of duplicate data will be among them [9].

One of the useful techniques is removing duplicate (De-duplication) data that simplifies and improves the management of storage. De-duplication technique detects and eliminates duplicated data and store only one copy of the data, so that reduces the space required for data storage. So it is clear that the removal of duplicated data helps to decrease size of storage memory in databases [10, 11, 12]. De-duplication process performed in four steps: in the first step, files are divided to smaller parts. In second step a new part will be generated. Control of similarity in the contents of data is performed by secure hash algorithm (SHA-1) (other methods can be performed too). In third step the structure of metadata will be updated. And in fourth step rest of data remained after De-duplication process, will be saved on the common storage resource [13]. In this article we have offered a dynamic approach to save data which has less overhead I/O read and write requests. The proposed approach has increased the rate of De-duplication to 23% so the rate reaches to78 % and there will be substantial decrease in CPU and RAM usage.

This paper is continued as follows: In the second part, the related work will be reviewed. In the third section, we

describe the proposed approach. Evaluate the effectiveness of the proposed approach will be described in Section IV, and finally Section V is devoted to conclusions and future work.

II. Related Works

This section provides an overview of the various researches to detect and remove duplicated data, and the advantages and disadvantages of them will be briefly examined.

Jin-Yong Ha et al [14], proposed a scheme called "Chunking aware of the content of the block (BLK - CAC)" to increase the rate of De-duplication in the solid state drives (SSD). In this scheme, each block is divided into several Chunks according to its content. They reviewed the results of related simulation, and concluded that rate of de-duplication in method of BLK-CAC is higher than the other similar methods. Also the method BLK-CAC, can more effectively serve large size files.

Meyer et al [15], proposed a method of calculating the total hash amount of the file content. They assessed the effectiveness of this strategy in the same part of the data. If two files have the same hash, then they will consist of duplicate content. Method were place at the lowest position in the rankings compared to a fixed size block and Chunking. It causes decreasing the throughput substantially because of computing huge amount of data for updating and repeated calculating SHA-1 digest.

F.Chen et al [14, 16] in 2011 invented two methods including "Chunking aware of the content of the file" (FILE-CAC) and "Chunking with the fixed size block" (BLK - FSC). Their studies showed that it was difficult to achieve high de-duplication rate in these two methods, Because the methods were not able to detect duplicated chunks at the time of inserting or removing a Chunk among others, hence there would be a relatively low de-duplication rate.

George Bebis et al. invented "Super-fingerprint method" to detect similar data in 2009 [17]. A Super-fingerprint is a group of fingerprints belonging to different parts of a file. as a Super-fingerprint is taken from several files, so that files with one or more similar super fingerprint will be similar.

"Simple de-duplication approach" [18], was promoted by Mr. João Tiago in 2009. The approach detected and destroyed duplicated data on servers in which multiple virtual machines were running. Virtual machines stored their pictures on a shared storage. This process partially reduced the amount of used memory of CPU and RAM. However this approach has a weak and low de-duplication rate whenever shared blocks need to be updated.

Another method was founded by U. Manber in 2008 [19, 20], which was in fact, a combination of Rabin Fingerprint and Chunking methods. Whenever a file was changed, this method had better performance compared to the other methods were introduced previously because it required computing signatures only for changed chunks whereas the other approaches needed to compute signature of all blocks of that file. The very high cost is one of the major disadvantages of this method.

RSYNC [1] method by Policroniades et al [21], was offered in 2004 to reduce the using bandwidth and to update two files (were used on separate computers) with the same content. By this solution, receiver separates files inside blocks and calculates hash functions for each block. The sender receives hash blocks and compares them with hashed file blocks. So RSYNC sends data only to those blocks whose receiver has lost data. Also it sends data to the other files or blocks whose receivers are present there.

J.Lavoie, J.M.Tracey [20] in 2004, proposed a method named "fixed size block" which can find duplicates in the block. Using this method, updates which changes part of file, cancels the other SHA-1 digest for the other blocks. As a result, the reference counter of the block is reduced and the SHA-1 digest is calculated for the new block. Using this approach increases the amount of storage space.

J.M.Tracey et al [22, 23], proposed REBL method in 2004, which was in fact a combination of compression and Chunking. It is able to detect and eliminate duplicated data like other approaches but low performance power and high costs are its disadvantageous.

Fred Douglis et al [24, 25], proposed "Delta encoding technique" to reduce the redundancy of similar files. The mentioned technique detects similar files then decreases the copied information. This technique can be used to compress several files also to reduce redundancy across multiple files.

III. De-Duplication In Cloud Computing Environment

The sharp growth of users' demands on cloud storage services, led us to study on de-duplication methods in order to find their pros and cons also to suggest an approach which will be able to solve low rate of de-duplication and high I/O overhead rate optimizing use of CPU and RAM spaces. On the other hand, most of the de-duplication approaches proposed so far has been static, so that after updating shared data, no longer they are not able to detect and remove duplicated data, hence, always have suffered from low de-duplication rate. So we're going to offer a new dynamic approach to improve sharing modules in a simple approach [18] which will be able to detect and remove duplicated data after updating part of shared data. In the proposed approach, de-duplication rate increases, the overhead of I/O reading and writing request and usage of CPU and RAM are greatly reduced. In the following, we will examine the proposed approach in detail.

A. proposed approach

In this paper, the framework of de-duplication consists of 3 main modules including: I/O Interception Module, Share Module and Garbage Collector Module. It should

be noted that due to better memory management, we have used a fixed block size of 4KB. The data structures needed for de-duplication in the proposed approach and performance of each has been shown in Table 1.

Table 1. Data structures needed to de-duplication in the proposed approach

Table	Description
L2P interpretation table	The virtual addresses related to read and write I/O requests are mapped to physical addresses. Each VM has its own L2P table.
Dirty address table	This table includes all virtual addresses. Following the introduction of a writing request, the writing operation is done normally without considering contents of request related to its duplication. Also address of virtual machine is registered. Then share module checks it. Any virtual machine has its own Dirty address table.
DHT Table (Distribution Hash Table)	The table is used to check duplications
P2LExplanation table	The table includes physical addresses which have retrieved by GC module and I/O interception module offers them to the addresses needing them.
Storage table Fingerprint	This table contains all the unique Chunks with their reference field. This field indicates the number of L2P elements, which have shared a single Chunk. Hence, each entry in the table contains a Fingerprint of Chunk, the number of L2P in a chunk and the reference field related to it.

It should be noted that Dirty address table uses to reduce the overhead of writing requests, but in this case we will require more disk space. Also, there should be a balance (trade-off) between the space consumed and the amount of I/O overhead writing operations. The purpose of the mechanism of Copy-on-write (COW), is the physical address which will be shared by more than one virtual addresses. Whenever the contents of a COW block needs to be updated, it does not change the block directly, but a new version will be written that the updated data will be corrected on.

- **I/O Interception Module:** I/O interception Module is responsible to intercept the request of I/O which are sent to the virtual disks on the blocks by the virtual machines. There is only one interception module I/O for any VM. This module only needs to check the L2P table in reading requests so that it will be able to convert virtual addresses to physical addresses, but for writing requests it needs Dirty addresses table in addition to L2P address table.
- **Share Module:** This module firstly checks the Dirty addresses. The addresses are not shared at the table immediately, because there might be some addresses among them which their blocks have been changing constantly. So there is no advantage in sharing them. If the block is shared, constantly changing or modifying, it leads increasing use of COW mechanism and respectively overflow of writing requests increases too. Therefore young and old sets will be used to avoid the storage of this type of blocks. It should be noted that Share Module performs the task concurrently for all VMs.

By selecting the virtual addresses which are ready to share, each of them can be processed independently and this is done by examining the entries in the L2P interpretation table. To share blocks, firstly virtual addresses existed in the L2P interpretation table should be updated according to new physical addresses in the entry of DHT table. After that a copy of duplicate block shared and duplicate content removed then the relevant physical block will be released.

Finally, it should be added one unit to number of fields relevant to the shared virtual addresses which are describing the physical blocks. Otherwise the physical address will be added to DHT table as a new entry and the block will be dedicated to the COW mechanism.

Whenever Share Module is working to share a physical address, simultaneously an update request for a block or blocks of addresses to be issued by user, it means a small piece of data like a chunk has been removed from set of chunks of a block or a new chunk will be inserted among chunks of the block. (When a data among Chunks of a sharing block has been written, the number of Chunks and pointer of L2P table to an element are shared in the relevant entry of L2P table). In this case, for inserting or removing a chunk, all the chunks which are located after

changed chunk, will be shifted so that de-duplication of these chunks will not perform well, provided the system has fixed size Chunks.

So in the proposed approach we will consider various size of chunks as a result we will be able to remove content of duplicated chunks which are located after position of inserted or removed chunk easily. Also we will be able to prevent their storage so the rate of de-duplication will be increased. Therefore we consider three phase of Chunking, Fingerprinting and Adjustment for this part of our approach. So after receiving an update request for a block or blocks of sharing physical address, we enter in Chunking phase. In Chunking phase, every block will be divided to many chunks in various size and contents. We use Rabin, Fingerprint method to find out content of Chunks. Also we suppose 12 existed chunks inside any block to improve management of memory also to prevent losing space.

Chunking phase continues until observing the boundaries of block, then we enter in Fingerprinting phase. In Fingerprinting phase, using the SHA-1 hash function, will be dedicated a 160-bit a Fingerprint for the entire Chunks of this block. Then at the adjustment phase fingerprints of the Chunk and Fingerprints on the table of fingerprint storage will be compared. By confirming the adjustment between fingerprints and existed fingerprints on the storage table, the content of a copy of duplicated chunks will be shared on the fingerprint storage table and the other contents of duplicated chunk will be removed. Then one unit will be added to the reference number of chunks on the fingerprint storage table. By completion of de-duplication, the entire of block included unique chunks will be shared by Share Module. In addition the block will be dedicated to the COW mechanism also will be added to the DHT table as a new entry (Fig.1).

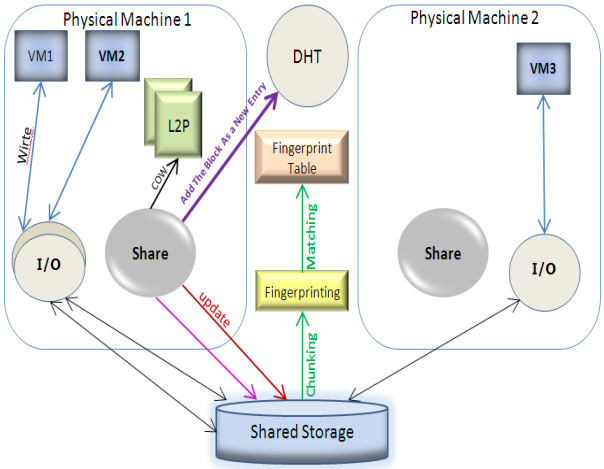

Fig.1. Share Module performance (after an update request)

- **Garbage Collector Module:** the entire of unused physical addresses keep in a queue called free line COW. Finally the GC module accesses them. This module also calculates approved content of any physical block for any input table DHT, and then the value of relevant reference field is considered which actually represents the number of shared virtual addresses. Provided the value of this field is zero, then the physical block can be added to the queue of free blocks because it can be used with any other virtual address. But if the value of the field is higher than zero, then the physical block cannot be added to the queue of free blocks and it shows the physical block is being used by another virtual address or addresses at the moment (Fig.2).

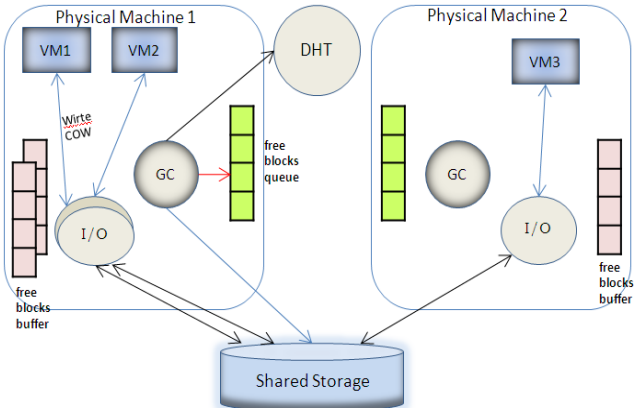

Fig.2. Performance of GC Module (first stage)

In the proposed approach, there is only one GC module works simultaneously for the entire of VMs. GC module in parallel and equally with Share Module accesses to DHT table (Fig.3). The general process of proposed algorithm has been shown on Fig.4.

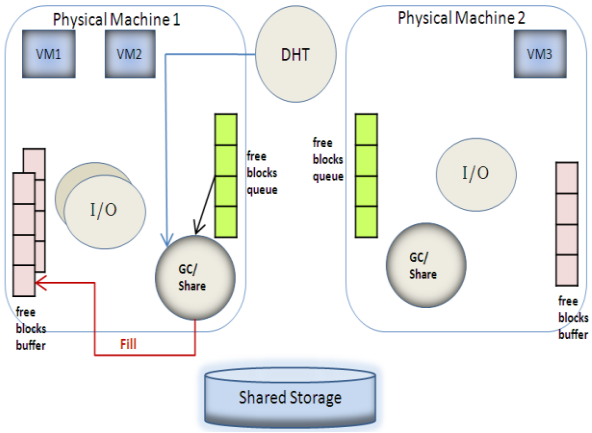

Fig.3. Performance of GC Module (second stage)

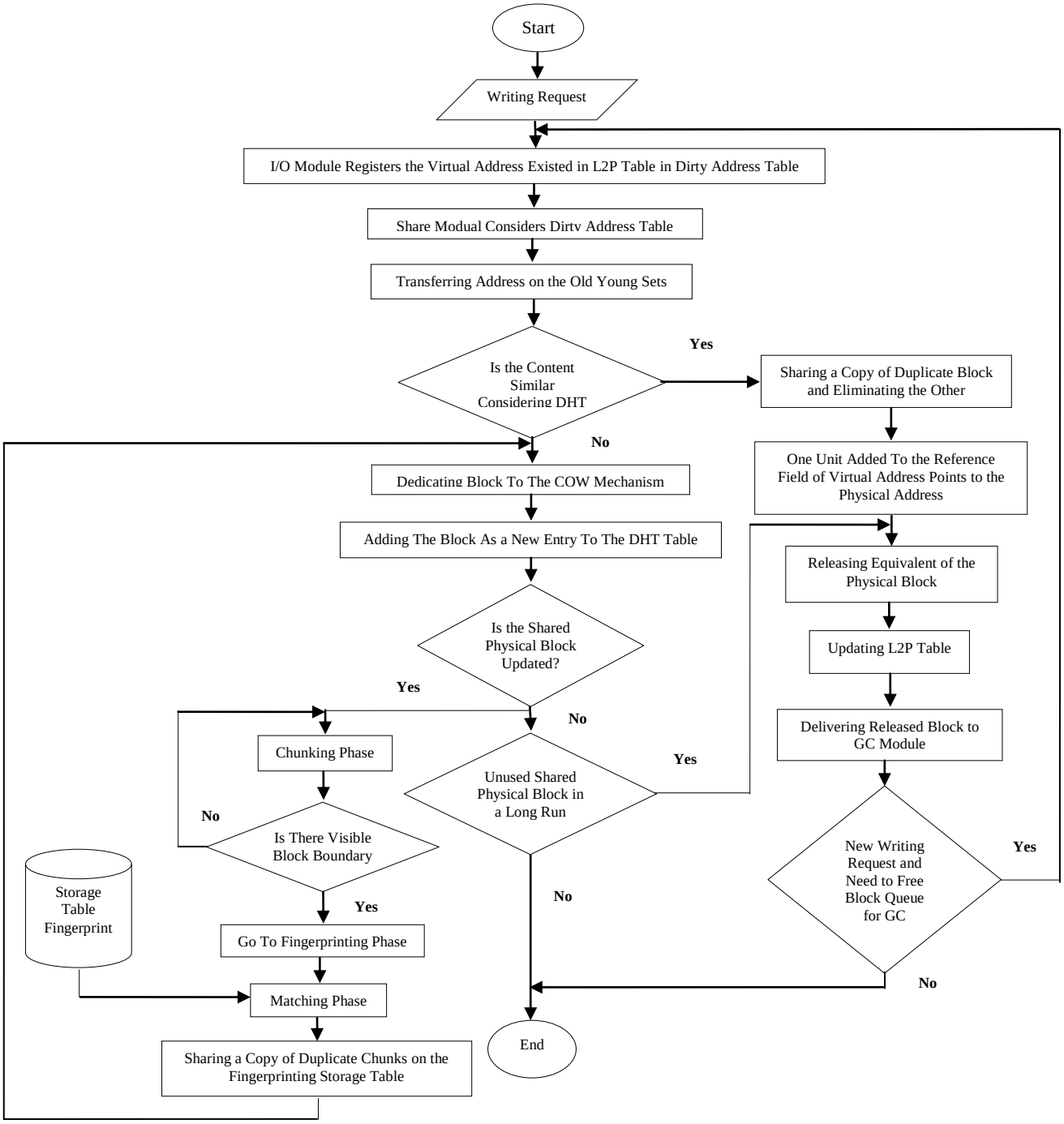

Fig.4.Diagram of proposed approach

IV. Performance Evaluation

The evaluation of the proposed approach has been carried out using three different measures then the results of simple approach [11] and the proposed one will be compared. Specifications of a test context for the implementation of the proposed approach are presented in Table 2. Also there has been used a 500 GB partition manufactured by HP storage works company of virtual designs (EVA 4400) accompanying RAID 0 for tests.

Furthermore, we have used data sets of San Diego [26] in simulation tests which show the amount of server used by different users. So simulation tests are developed by applying 3 benchmarks which actually present practical tests for de-duplication in a virtual scenario (table 3). Practice will be done by measuring the power of VMs I/O requests and used CPU and RAM in DOM 0 also the amount of shared data.

Table 2. Specification of test context for implementation of proposed approach

Parameters	Value
Utilized processor in server	COREi7-720QM 2.8 GHz
RAM in server utilized	6GB
the number of VMs in server	3
Type of kernel used	Ubuntu 9-Xen
RAM in VMs used	256MB
Volume of relevant disks for any VMs	10GB
used operational system for DOM 0	Linux 2.6.32
The type of Hypervisor	Xen 3.3.0
number of used measures to evaluate proposed approach performance	3
The programming language used for testing and implementing measures	C++
Duration of each VM	30 min

Table 3. benchmarks used in simulation of proposed approach

Metric	Description	Objective
Bonnie++	Measuring throughput of I/O reading and writing requests by different sets of tests	Measuring the overhead of I/O reading and writing requests , rate of sharing and the amount of used CPU and RAM
Writing	Evaluating the amount of overhead in I/O writing requests in a VM	
Reading	Evaluating the amount of overhead in I/O reading requests in a VM	

Continued the optimality of the proposed approach against simple approach will be evaluated in three scenarios and then we interpret the results.

A. First Scenario

This scenario evaluates result of simulation of the proposed approach using standard Bonnie ++. It should be noted that a type of bonnie ++ is executed for each of 3 applied VM in proposed approach. Table 4 shows the generated overhead for both simple and proposed approaches. Results of the table represent that there are 4 tests which are generating more overhead. Also comparing diagrams related to before (Fig.5) and after (Fig.6) applying proposed approach, it shows clearly that there is 23 percent increase in de-duplication rate in suggested approach.

Table 4. Result of I/O throughput by Bonnie ++

	Simple Approach	Proposed Approach	Overload
Put-c Test	73,285 KB / sec	60,500 KB / sec	7 %
Write Block Test	162,483 KB / sec	125,500 KB / sec	22 %
Rewrite Test	39,064 KB / sec	32,000KB / sec	16 %
Get-c Test	35,549 KB / sec	26,000KB / sec	20 %
Read Block Test	106,119 KB / sec	83,000 KB / sec	20 %
Random Seeks Test	304 KB / sec	280 KB / sec	4 %

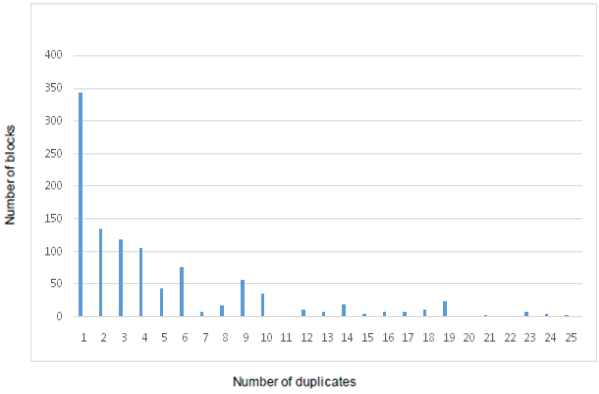

Fig.5. redundancy results of San Diego data sets for blocks included duplications less than 25. (Before applying proposed approach)

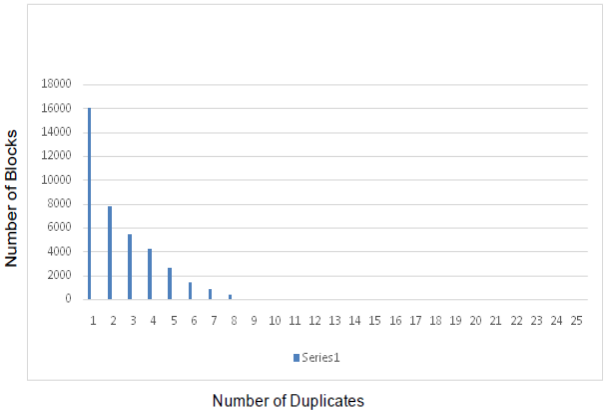

Fig.6. redundancy interpretation for blocks included duplications less than 25. (After applying proposed approach)

Fig.7 compares throughput of our proposed and simple approaches.

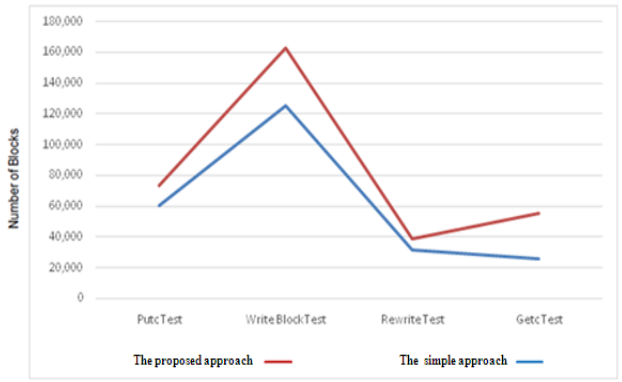

Fig.7. throughput comparison between proposed and simple approaches

Table 5 shows the amount of shared data. Our proposed algorithm shares 41 percent of data according to Bonnie++ benchmark.

Table 5. Redundancy results with Bonnie ++ benchmark

	Proposed Approach
Storage space	4.00 GB
Writing Space	10 GB
Percent space savings	41 %

Table 6 shows average values for usage of CPU and RAM. It is possible to explain differences between RAM using data structures required for sharing process. Fig.8 represents it explicitly.

Table 6. Result of RAM and CPU by Bonnie++ benchmark

	Simple Approach	Proposed Approach
CPU Usage Average	16.00 %	39.10 %
RAM Usage Average	24.60 MB	276.7 MB

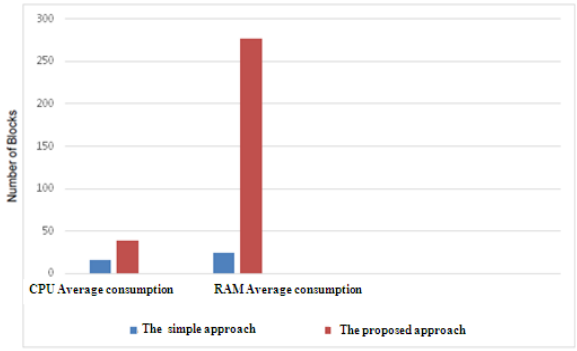

Fig.8. Comparison average used PU and RAM in our proposed approach and the simple one.

B. Second Scenario

The explaining Scenario evaluates results of simulation of our proposed approach using the writing benchmark. We run writing benchmark 30 minutes for any VMs. Table 7 shows the amount of computing power, delay of writing requests, RAM and CPU usage in simple approach and our proposed one. The results show that the average amount of CPU usage, in our explaining Scenario evaluates results of simulation of our proposed approach using the writing metric. We run writing benchmark 30 minutes for each VMs. Table 7 shows the amount of computing power, delay of writing requests, RAM and CPU usage in simple approach and our proposed one that. The results show that the average amount of CPU usage, in our approach is very small. As it is possible the proposed approach consists of series of batch requests to buffer released blocks.

Table 7. Results of writing benchmark

	Proposed Approach
Storage Space	2 GB
Writing Space	3 GB
Percent space savings	68 %

Proposed approach has been optimized completely and the average amount of CPU usage is acceptable compared to the simple approach result. Higher usage of RAM is due to special data structure and de-duplication process on chunks which have been inserted after updating a shared block. The introduced overhead for our proposed approach reported less than 4 percent according to the last study.

Table 8 shows the amount of shared data by both simple and our proposed approaches. Our proposed approach shares 30 percent of written data.

Table 8. Redundancy results for writing benchmark

	Simple Approach	Proposed Approach
Amount Shared Block	1,128,817	1,300,748
Storage Space	4.31 GB	5.00 GB
Writing Space	15.23 GB	20.00 GB
Percent Space Savings	28 %	30%

C. Third Scenario

This scenario evaluates the obtained results of proposed approach simulation using reading benchmark. Again we run reading benchmark for 3 VMs. 4 processes have been produced in any of VMs and data is read on a 2GB file or is written on it. The written result obtains from 10 minutes operation. This benchmark is applied for both of approaches. Table 9 shows throughput of reading and usage of CPU and RAM. The value of RAM is related to the entire of benchmark operation but the value of CPU is only related to the reading process. Our proposed approach does not produce a substantial overhead for I/O reading requests in a VM and it increases usage of CPU by 2 percent. Table 10 represents that our approach shares 68 percent of data. Value resulted by this metric is higher than the other 2 benchmarks, because this benchmark writes less content than other 2 benchmarks.

Table 9. Results Of Reading benchmark

	Simple Approach	Proposed Approach
Write Throughput	882	860
Write Delay	13.5 ms	13ms
RAM Usage Average	2 MB	218 MB
CPU Usage Average	9 %	11 %

Table 10. Redundancy Results For Reading benchmark

	Simple Approach	Proposed Approach
Write Throughput	2529	2437
Write Delay	3.8ms	4.0ms
RAM Usage Average	2.67 MB	242.15 MB
CPU Usage Average	20.00 %	30.00 %

V. Conclusion And Future Work

As mentioned above, the aim of all de-duplication techniques is detection and elimination of redundant data to save required storage space. It also does not cause to produce a lot of overhead in the I/O requests. So the best method among de-duplication approaches will be the dynamic one. It means considering and tracking VM's I/O reading and writing requests on every moment also it will be able to consider optimally all of updating requests for shared data and increase the rate of de-duplication with low cost for storage. In this paper we tried to offer a new dynamic approach to optimize performance of Share Module in simple approach which

will be able to detect and eliminate the sharing duplicated data after being updated. Our proposed approach promoted rate of De-duplication to 78 percent, according to the results of tests. By assessing proposed approach according to offered benchmarks, were presented that our approach will save storage space dramatically by decreasing usage of CPU and RAM. We recommend researching more to find an approach for better management of data structures lead to optimize usage of RAM.

REFERENCES

[1] Monireh Fallah, Mostafa Ghobaei Arani and Mehrdad Maeen. "NASLA: Novel Auto Scaling Approach based on Learning Automata for Web Application in Cloud Computing Environment." *International Journal of Computer Applications* 113(2):18-23, March 2015.

[2] Monireh Fallah, Mostafa Ghobaei Arani "ASTAW: Auto-Scaling Threshold-based Approach for Web Application in Cloud Computing Environment." International Journal of u- and e- Service, Science and Technology (IJUNESST),Vol.8, No.3, pp.221-230, 2015.

[3] Chengzhang Peng, Zejun Jiangb. "Building a Cloud Storage Service System". *Science Direct*, 2011, pp.691-696.

[4] Afife Fereydooni, Mostafa Ghobaei Arani and Mahboubeh Shamsi "EDLT: An Extended DLT to Enhance Load Balancing in Cloud Computing." *International Journal of Computer Applications* 108(7):6-11, December 2014.

[5] Md. Imran Alam, Manjusha Pandey, Siddharth S Rautaray,"A Comprehensive Survey on Cloud Computing", IJITCS, vol.7, no.2, pp.68-79, 2015. DOI: 10.5815/ijitcs.2015.02.09.

[6] R. P. Goldberg. Survey of virtual machine research. Computer, 7(6): 1974, pp. 34_45.

[7] J. E. Smith and R. Nair, "The architecture of virtual machines", Computer, 38(5): 2005, pp.32-38.

[8] Carolan, Jason, Steve Gaede, James Baty, Glenn Brunette, Art Licht, Jim Remmell, Lew Tucker, and Joel Weise. "Introduction to cloud computing architecture." *White Paper, 1st edn. Sun Micro Systems Inc* (2009).

[9] T. E.Denehy and W. W. Hsu, "Duplicate management for reference data", Technical report, IBM Research, 2003.

[10] Hovav Shacham, Brent Waters: Compact Proofs of Retrievability. ASIACRYPT 2008, pp. 90-107.

[11] B. Zhu, K. Li, and H. Patterson, "Avoiding the Disk Bottleneck in the Data Domain De-duplication File System," Proc. FAST '08: Sixth USENIX Conf. File and Storage Technologies, 2008, pp. 1-14.

[12] Indu Arora, Dr. Anu Gupta. "Opportunities, Concerns and Challenges in the Adoption of Cloud Storage", *(IJCSIT) International Journal of Computer Science and Information Technologies, vol 3(3)*, 2012, pp. 4543-4548.

[13] Deepak Mishra , Dr.Sanjeev Sharma ,"Comprehensive Study Of Data de-duplication" ,International Conference On Cloud, Big Data and Trust , Nov 13-15, RGPV,2013.

[14] Jin-Yong Ha, Young-Sik Lee, and Jin-Soo Kim , "De-duplication with Block-Level Content-Aware Chunking for Solid State Drives (SSDs)", IEEE International Conference on High Performance Computing and Communications & IEEE International Conference on Embedded and Ubiquitous Computing, 2013 ,pp.2.

[15] Meyer, Dutch T., and William J. Bolosky. "A study of practical de-duplication." *ACM Transactions on Storage (TOS)* 7, no. 4 (2012): 14.

[16] A. Gupta, R. Pisolkar, B. Urgaonkar, and A. Sivasubramaniam, "Leveraging value locality in optimizing NAND flash-based ssds," in Proc. USENIX Conference on File and Storage Technologies, 2011, pp. 7–7.

[17] Uz, Tamer, George Bebis, Ali Erol, and Salil Prabhakar. "Minutiae-based template synthesis and matching for fingerprint authentication." *Computer Vision and Image Understanding* 113, no. 9 (2009): 979-992.

[18] Joao Tiago, Medeiros Paulo , Escola De Engenharia , Mestrado Em Engenharia ,"Efficient Storage Of Data In Cloud Computing" , Journal Of ACM Computing Surveys(CSUR), July 2009 , pp.3-7.

[19] Youjip Won, Jongmyeong Ban, Jaehong Min , Jungpil Hur, Sangkyu Oh, Jangsun Lee, " Efficient index lookup for De-duplication backup system" , National ResearchLab at Hanyang University, 2008, pp2-3.

[20] P. Kulkarni, F. Douglis, J. LaVoie, and J. M. Tracey," Redundancy elimination within large collections of files", In ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference, USENIX Association, 2004, pp.5-5.

[21] Policroniades, Calicrates, and Ian Pratt. "Alternatives for Detecting Redundancy in Storage Systems Data." In *USENIX Annual Technical Conference, General Track*, pp. 73-86. 2004.

[22] M. Szeredi, "File system in user space. http://fuse.sourceforge.net/", accessed 5th October, 2014.

[23] Purushottam Kulkarni, Fred Douglis, Jason LaVoie, John M. Tracey, "Redundancy Elimination Within Large Collections of Files", In *Proceedings of the 2004 USENIX Annual Technical Conference*, Boston, MA, June 2004, pp.7-10.

[24] Fred Douglis and Arun Iyengar. Application-specific delta-encoding via resemblance detection. In *Proceedingsof 2003 USENIX Technical Conference*, June 2003.

[25] Philip Shilane, Grant Wallace, Mark Huang, Windsor Hsu," Delta Compressed and De-duplicated Storage Using Stream-Informed Locality", Journal of Backup Recovery Systems Division EMC Corporation, 2012, pp.3.

[26] http://ucsdnews.ucsd.edu/archive/newsrel/supercomputer/2011_09cloud.asp.

Authors' Profiles

Fatemeh Shieh received the B.S.C degree in Software Engineering from University Bandarabbas, Iran in 2009, and M.S.C degree from Azad University of mahallat, Iran in 2014, respectively. Her research interests include Cloud Computing, Distributed Systems and Vehicular Cloud Computing.

Mostafa Ghobaei Arani received the B.S.C degree in Software Engineering from IAU Kashan, Iran in 2009, and M.S.C degree from Azad University of Tehran, Iran in 2011, respectively. He is a PhD Candidate in Islamic Azad University, Science and Research Branch, Tehran, Iran. His research interests include Grid Computing, Cloud Computing, Pervasive Computing, Distributed Systems and Software Development.

Mahboubeh Shamsi is Associate Prof. Qom University of Technology. She's received the B.S.C degree in Mathematics from Isfahan University, Iran in 2003, and received the M.S.C degree from Azad University of Isfahan, Iran in 2006, and also received the PhD degree in Software Engineering from UTM in 2011. Her research interests include Image Processing, Design and Implementation of Persian/Arabic OCR, Design and Implementation of Biometric Authentication System, Design and Test of Reliable Software, Databases, Software Engineering, UML, ERD, DFD, ERP.

How to cite this paper: Fatemeh shieh, Mostafa Ghobaei Arani, Mahboubeh Shamsi,"An Extended Approach for Efficient Data Storage in Cloud Computing Environment", IJCNIS, vol.7, no.8, pp.30-38, 2015.DOI: 10.5815/ijcnis.2015.08.04

