ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

Efficient Cloud Storage Management Using DHT Mechanism

D.K.Karthika*, G.Sudhakar, D.Sugumar

*PG Student, Department of CSE, Ranganathan Engineering College, Coimbatore, India

HOD, Department of CSE, Ranganathan Engineering College, Coimbatore, India

PG Student, Department of CSE, Ranganathan Engineering College, Coimbatore, India

ABSTRACT: Cloud computing enables a wide range of users to access distributed over the Internet. During the centralized cloud management to satisfy user needs and data backup becomes a critical concern. The proposed method solves this issue by focuses on the following factors, user provisioning cost, security, load balancing and redundancy .To carry out this process perform local and global de-duplication process, chunking of data, hash function, load balancing. An intelligent data chunking and Hash functions is performed to chunks of data. Load balancing function is used to balance the load while storing the files into the resources of cloud. Propose a fully distributed scheme which works in a constant number of rounds and achieves optimal balance with high probability. The optimal centralized algorithm is used for generating the migrations in cloud storage. Perform experiments to demonstrate that proposed scheme can significantly higher efficiency over the state-of-the-art methods.

KEYWORDS: Cloud computing, personal storage, source deduplication, deduplication efficiency, application awareness, load balancing, distributed hash table.

I. INTRODUCTION

The fast growth of data volumes leads to an increased demand for online storage services, ranging from simple backup services to cloud storage infrastructures. Remote backup services give users an online system for collecting, compressing, encrypting, and transferring data to a backup server that is provided by the hosting company. Cloud storage refers to scalable and elastic storage capabilities that are delivered as a service using Internet technologies with elastic provisioning and use-based pricing that does not penalize users for changing their storage consumption without notice [1-2]. The term data deduplication refers to techniques that store only a single copy of redundant data, and provide links to that copy instead of storing other actual copies of this data.

With the transition of services from tape to disk, data deduplicationhas become a key component in the backup process. Bystoring and transmitting only a single copy of duplicate data,deduplication offers savings of both disk space and networkbandwidth. For vendors, it offers secondary cost savings inpower and cooling achieved by reducing the number of diskspindles [3].

Data deduplication strategies can be categorized according to the basic data units they handle. In this respect there are two main data deduplication strategies: (1) File-level deduplication, in which only a single copy of each file is stored. Two or more files are identified as identical if they have the same hash value. This is a very popular type of service offered in multiple products [4]; (2) Block-level deduplication, which segments files into blocks and stores only a single copy of each block. The system could either use fixed-sized blocks [5] or variable-sized chunks [6]. The discussion in this paper may be applied to both strategies. In terms of the architecture of the deduplication solution, there are two basic approaches. In the target-based approach deduplication is handled by the target data-storage device or service, while the client is unaware of any deduplication that might occur. This technology improves storage utilization, but does not save bandwidth. On the other hand, source based deduplication acts on the data at the client before it is transferred. Specifically, the client software communicates with the backup server (by sending hash signatures) to check for the existence of files or blocks. Duplicates are replaced by pointers and the actual duplicate

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

data is never sent over the network. The advantage of this approach is that it improves both storage and bandwidth utilization.

The existing source deduplication strategies can be divided into two categories: local source deduplication [7] that only detects redundancy in backup dataset from the same device at the client side and only sends the unique data chunks to the cloud storage, and global source deduplication [8] that performs duplicate check in backup datasets from all clients in the cloud side before data transfer over WAN. The former only eliminates intra-client redundancy with low duplicate elimination ratio by low-latency client-side duplicate data check, while the latter can suppress both intra-client and inter-client redundancy with high deduplication effectiveness by performing high-latency duplication detection on the cloud side.

Inspired by Cloud4Home [9] that enhances data services by combining limited local resources with low latency and powerful Internet resources with high latency, local-global source deduplication scheme that eliminates intra-client redundancy at client before suppression inter-client redundancy in the cloud, can potentially improve deduplication efficiency in cloud backup services to save as much cloud storage space as the global method but at as low latency as the local mechanism. In cloud computing environment, the random arrival of tasks with random utilization of CPU service time requirements can load a specific resources heavily, while the other resources are idle or are less loaded [10]. Hence, resource control or load balancing is major challenging issue in cloud computing. Load balancing is a methodology to distribute workload across multiple computers, or other resources over the network links to achieve optimal resource utilization, maximize throughput, minimum response time, and avoid overload. This research work is based on simulation technique and it uses the cloud simulator, CloudSim [11].

In this paper, propose a novel distribute hash table (DHT) function for load balancing of user files which is chunked from deduplication methods for personal user. So before performing the load balancing task first need to perform ALG-Dedupe deduplication, that not only exploits application awareness, but also combines local and global duplication detection, to achieve high deduplication efficiency by reducing the deduplication latency to as low as the application-aware local deduplication while saving as much cloud storage cost as the application-aware global deduplication. To gain maximum profits with optimized load balancing algorithms, it is necessary to utilize resources efficiently,so DHT is designed to solve load balancing problem.Observe that there is a significant difference amongdifferent types of applications in the personal computingenvironment in terms of data redundancy, sensitivity todifferent chunking methods, and independence in the deduplicationprocess. Solution to the load balancing problem is to choose a used hash function uniformly at random to perform load balancing task after the local and globalduplicate check processes is completed to significantly improve thededuplication efficiency , reduce the system overhead and highly balanced load .

The paper is organized as follows. In Section II, review related works. In Section III, describe proposed methods for deduplication and load balancing with use distributed hash functions. In Section IV, simulation results evaluate the performance of the algorithms. The conclusion and issues of the current work is drawn in Section VI

II. BACKGROUND KNOWLEDGE

At a cloud cluster node, each instance of a guest operating system runs on a virtual machine, accessing virtual hard disks represented as virtual disk image files in the host operating system. For VM snapshot backup, file level semantics are normally not provided. Snapshot operations take place at the virtual device driver level, which means no fine-grained file system metadata can be used to determine the changed data. Backup systems have been developed to use content fingerprints to identify duplicate content [12]. Offline deduplication is used in [13] to remove previously written duplicate blocks during idle time. Several techniques have been proposed to speedup searching of duplicate fingerprints. For example, the data domain method [14] uses an in-memory Bloom filter and a prefetching cache for data blocks which may be accessed. An improvement to this work with parallelization is in [15-16].

The approximation techniques are studied in [17] to reduce memory requirement with a tradeoff of the reduced deduplication ratio. In comparison, this paper focuses on full deduplication without approximation. Additional inline

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

deduplication techniques are studied in [13]. All of the above approaches have focused on such inline duplicate detection in which deduplication of an individual block is on the critical write pathrequests.

Sub file hashing (SFH) [18] is appropriately named. Whenever SFH is being used, it means the file is broken into a number of smallersections before data de-duplication. The number of sections depends on the type of SFH that is beingused. The two most common types of SFH are fixedsize chunking and variable-length chunking. In a fixed-size chunking approach, a file is divided up into a number of fixed-size pieces called "chunks". In a variable-length chunking approach, a file is broken up into "chunks" of variable length. Some techniques such as Rabin fingerprinting [19] are applied to determine "chunk boundaries". Each section is passed to a cryptographic hash function (usually MD5 or SHA-1) to get the "chunk identifier". The chunk identifier is used to locate replicate data. Both of these SFH approaches find replicate data at a finer granularity but at a price.

In [20] file replication is performed through multiple hash functions, which are organized in a tree. This results in the replication servers being organized into a tree. In our case, the replication servers have no topological relationship. FDR is implemented on dedicated servers called request redirectors, which maintain information on server availability for each object and server load. Such information may not be accurate at all time and may not be consistent among the redirectors, which may cause unnecessary overload on some servers. On the other hand, there is no need of dedicated servers with our algorithm. The only information each client has is the set of hash functions. Server availability can be found by random binary search and server load is monitored by the server itself, which is always accurate. Overall, our algorithms are highly decentralized and keep minimum state information without loss of accuracy of needed information, and therefore, should be more robust against failures and other contingencies.

III. PROPOSED LOAD BALANCING AND LOCAL-GLOBAL SOURCE DEDUPLICATION

The proposed distribute hash table (DHT) based load balancing schema and the proposed Application-aware Local-Global source (ALG) schema for deduplication schema is motivated in personal cloud computing environment in Section 2, is designed to meet the requirement of load balancing and deduplication efficiency with high deduplication effectivenessand low system overhead. The main idea of the proposed ALGDedupeis 1) exploiting both low-overhead local resourcesand high-overhead cloud resources to reduce the computationaloverhead by employing an intelligent data chunking scheme and an adaptive use of hash functions based onapplication awareness, and 2) to mitigate the on-disk indexlookup bottleneck by dividing the full index into smallindependent and application-specific indices in an application- aware index structure. 3) The main idea of the proposed DHT based load balancing schema is to perform the load balancing task for stored files in the ALGDedupe phase ,where the proposed method the task are efficiently balanced. The storage nodes are structured as a network based on distributed hash tables (DHTs), e.g., discovering a file chunk can simply refer to rapid key lookup in DHTs, given that a unique handle (or identifier) is assigned to each file chunk. The proposed results improve the detection results of the data redundant files with low system overheadon the client side and highly balanced system result in the personal cloud computing environment.

An architectural overview of proposed ALG and DHT is illustrated in Fig.1, where tiny files are first filtered out by file size filter for efficiency reasons, and backup data streams are broken into chunks by an intelligent chunker using an application aware chunking strategy. Data chunks from the same type of files are then deduplicated in the application-aware deduplicator by generating chunk fingerprints in hash engine and performing data redundancy check in application-aware indices in both local client and remote cloud.DHTs enable nodes to self-organize and Repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. The chunk servers in our proposal are organized as a DHT network.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

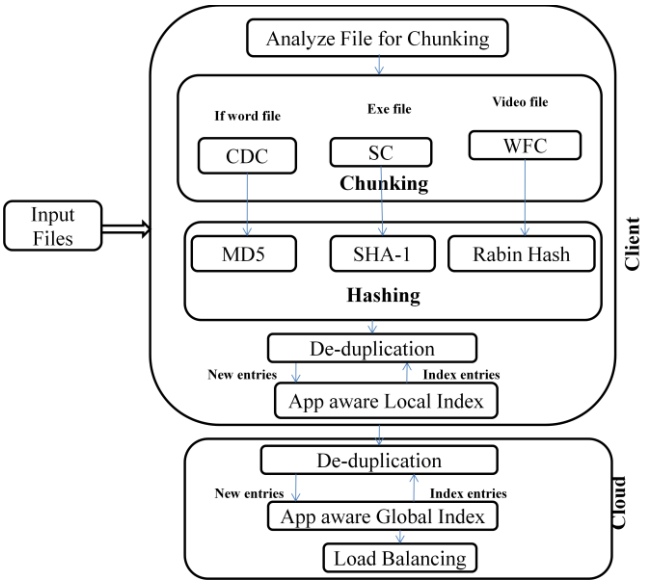

Fig.1. Architectural overview of the proposed ALG deduplication and DHT for load balancing design

A. *Analysis the files for chunking:*
Investigate how data redundancy, space utilization efficiency of popular data chunking methods and computational overhead of typical hash functions change in different applications of personal computing to motivate our research. Perform preliminary experimental study on datasets collected from desktops in our research group, volunteers' personal laptops, personal workstations for image processing and financial analysis, and a shared home server. To the best of our knowledge, this is the first systematic deduplication analysis on personal storage.

To verify the data redundancy in the compressed files, carried out chunk-level deduplication using two popular methods: Static Chunking (SC) [21] of 4 KB chunk size and TTTD based Content Defined Chunking (CDC) [22] of 4 KB average chunk size (Min: 2 KB, Max: 16 KB) after file level deduplication in about 2.6TB data of typical PC applications using compression, respectively.Here, according to the function of applications, we group file types using compression into application groups: video, audio, image, Linux-AC for compressed archive file types in Linux, Mac-AC for compressed archive file types in Mac OS X, Windows-AC for compressed archive file types in Windows.
Stronger hash functions for fine-grained chunks are an effective way to reduce the computational overhead and RAM usage for local deduplication in PC clients. To evaluate the computational overhead of typical hash functions on datasets measured the 4-thread parallel computing throughputs of the Rabin hash, Message digest (MD5) and secure hash algorithm (SHA)-1 hash algorithms respectively in user space on a laptop with 2.53 GHz Intel Core 2 Duo for fingerprinting data chunks obtained, respectively, from the Whole FileChunking (WFC), which uses an entire file as the basis for duplicate detection [21], the SC-based chunking with 4 KB fixed chunk size, and the CDC-based chunking with 4 KB average chunk size. Rabin hash is a rolling hash function with lower computational overhead than cryptographic hash functions SHA-1 and MD5.

B. *Intelligent Data Chunking*
The deduplication efficiency of data chunking scheme among different applications differs greatly .Depending on whether the file type is compressed or whether SC can outperform CDC in deduplication efficiency, divide files into three main categories: compressed files, static uncompressed files, and dynamic uncompressed files. The dynamic files are always editable, while the static files are uneditable in common. To strike a better tradeoff between duplicate elimination ratio and deduplication overhead, deduplicate compressed files with WFC, separate static uncompressed files into fix-sized chunks by SC with ideal chunk size, and break dynamic uncompressed files into variable-sized chunks with optimal average chunk size using CDC based on the Rabin fingerprinting to identify chunk boundaries

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

C. *Application-Aware Deduplicator*

After data chunking in intelligent chunker module, data chunks will be deduplicated in the application-aware deduplicator by generating chunk fingerprints in the hash engine and detecting duplicate chunks in both the local client and remote cloud. ALG-Dedupestrikes a good balance between alleviating computation overhead on the client side and avoiding hash collision to keep data integrity. Employ an extended 12-byte Rabin hash value as chunk fingerprint for local duplicate datadetection and a MD5 value for global duplicate detection of compressed files with WFC. In both local and globaldetection scenarios, a SHA-1 value of chunk serves as chunk fingerprint of SC in static uncompressed files and a MD5 value is used as chunk fingerprint of dynamic uncompressed files since chunk length is another dimension for duplicate detection in CDC-based deduplication.

D. *Application-Aware Index Structure*

An application-aware index structure consists of an in- RAM application index and small hash-table based on-disk indices classified by application type. According to the accompanied file type information, the incoming chunk is directed to the chunk index with the same file type. Each entry of the index stores a mapping from the fingerprint (fp) of a chunk or with its length (len) to its container ID(cid).As chunk locality exists in backup data streams [23], a small index cache is allocated in RAM to speedup index lookup by reducing disk I/O operations. The index cache is a key-value structure, and it is constructed by a doubly linked list indexed by a hash table. When the cache is full, fingerprints of those containers that are ineffective in accelerating chunk fingerprint lookup are replaced to makeroom for future prefetching and caching. The current cache replacement policy in ALG-Dedupe is Least-Recently-Used (LRU) on cached chunk fingerprints. Furthermore, a periodical data synchronization scheme is also proposed in ALG-Dedupe to backup the application-aware local index and file metadata in the cloud storage to protect the data integrity of the PC backup datasets.

E. *Load balancing using distributed hash table (DHT)*

The goal of this is to provide a scheme for balancing the load in the personal cloud computing which is categorized into several number of chunks which, for every cloud server i, returns an estimate n_i of the total number of chunk files for each cloud server, so that each n_i is within a constant factor of n, with high probability. A DHT network is an over lay on the application level. The logical proximity abstraction derived from the DHT does not necessarily match the physical proximity information in reality. That means a message traveling between two neighbours in a DHT over lay may travel along physical distance through several physical network links. In the load balancing algorithm, light nodes n may rejoin as a successor of are heavy node j. Then, the requested chunks migrated from j to i need to traverse several physical network links, thus generating considerable network traffic and consuming significant network resources (i.e., the buffers in the switches on a communication path for transmitting a file chunk from a source node to a destination node). We improve our proposal by exploiting physical network locality. Basically, instead of collecting as in vector per algorithm i around, each light node i gathers NV vectors. Each vector is built using the method introduced previously. From the NV vectors, the light node I seeks NV heavy nodes by invoking Algorithm1 (i.e., SEEK) for each vector and then selects the physically closest heavy node based on the message round-trip delay.

Each node n estimates the size of the cloud environment a as follows. It sets initially $l := l_n$ which is the length of its interval and $m := m_n$ which is the number of 2 markers its interval stores. As long as $m < log\frac{1}{l}$, the next not yet contacted successor is contacted and both l and m are increased by its length and the number of markers, respectively. After that, l is decreased so that m = $log\frac{1}{l}$. This can be done locally using only the information from the last server on our path.Will call the intervals of length at most$\frac{4}{l \cdot n_i}$ short and intervals of length at least $12 \cdot u/l^2 \cdot n_i$long. Intervals of length between $\frac{4}{l \cdot n_i}$and $12 \cdot u/l^2 \cdot n_i$will be called middle. Notice that short intervals are defined so that each middle or long interval has length at least $4/n_i$. On the other hand, long intervals are defined so that halving long interval we never obtain a short interval. The algorithm will minimize the length of the longest interval. Therefore, before begin the routine, force all the intervals with lengths smaller than $\frac{1}{2.l.n_i}$. By doing this, assure that the length of the shortest interval in the cloud computing environment for personal user will be bounded from below by $\frac{1}{2.l.n_i}$.

First of all, it is possible that remove a huge fraction of the nodes for each chunk of data files for cloud computing environment . It is even possible that a very long interval appears, even though the cloud computing environment was

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

balanced before. This is not a problem, since the algorithm will rebalance the system. Besides, if this algorithm is used also for new nodes at the moment of joining, this initialization will never be needed. The number of nodes $n$for each chunk data files of user and the removed files from user will later act as though they were simple short intervals. Each of these nodes can contact the cloud computing environment through its cloud server. Our algorithm works in rounds. In each round find a linear number of short intervals which can leave the cloud computing environment without introducing any new long intervals and then use them to divide the existing long intervals.The routine works differently for different nodes, depending on the initial cloud server's interval's length. The middle intervals and the short intervals which decided to stay, help only by forwarding the contacts that come to them. The pseudocodes for all types of intervals are depicted in Fig.1.

IV. PSEUDO CODE

Pseudocode 1: Load balancing using distributed hash table (DHT)

Short the number of files in the chunk with number of user
State:=staying
If(predecessor is short) // number of files and task is short
With probability ½ change state to leaving
If (state =leaving and predecessor .state =staying)
{
P:=random(0.1)
P:=the node responsible for P
Contact consecutively the node P and its 6.log(u.n_i) successor on the ring
If (a node R accepts)
Leave and rejoin in the middle of R
}
At any time ,if any node contacts reject imbalanced
Middle
At any time ,if any node contacts reject imbalanced
Long
Wait for contacts
Id any node contacts accept the current load task and balanced

In distributed file systems (e.g., Google GFS and Hadoop HDFS), a constant number of replicas for each file chunk are maintained in distinct nodes to improve file availability with respect to node failures and departures. Our current load balancing algorithm does not treat replicas distinctly. It is unlikely that two or more replicas are placed in an identical node because of the random nature of our load rebalancing algorithm. More specifically, each under loaded node samples a number of nodes, each selected with a probability of 1/n, to share their loads (where n is the total number of storage nodes).

V. EXPERIMENTATION RESULTS

In this section we have evaluated the advantages of our design over the state-of-the-art source deduplication based cloud backup services in terms of deduplication efficiency, energy consumption and system overheads by feeding the real-world datasets in a personal computing device. The following evaluation subsections will show the results, beginning with a description of the experiment platform with PC backup datasets we use as inputs.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

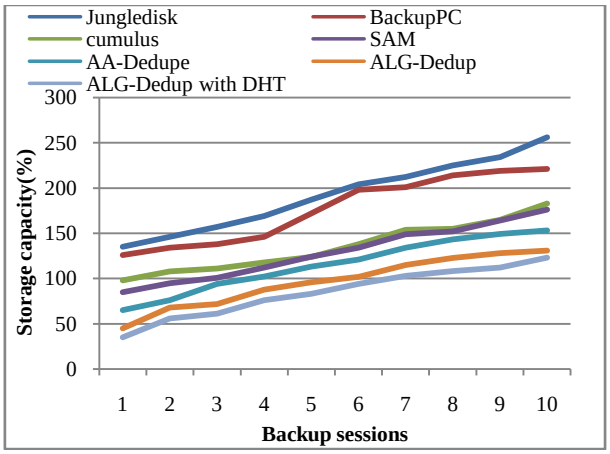

Fig.2.Cloud storage space requirement

The experimental result is shown in the Fig. 2 present the cumulative cloud storage capacity required of the providers at each backup session for individual user with the six cloud backup schemes. Different from source deduplication schemes, Jungle Disk fails to achieve high cloud storage saving due to the fact that its incremental backup scheme cannot eliminate file copies written in different places. In the source deduplication schemes, the coarse-grained method BackupPC cannot find more redundancy than other fine-grained mechanisms.

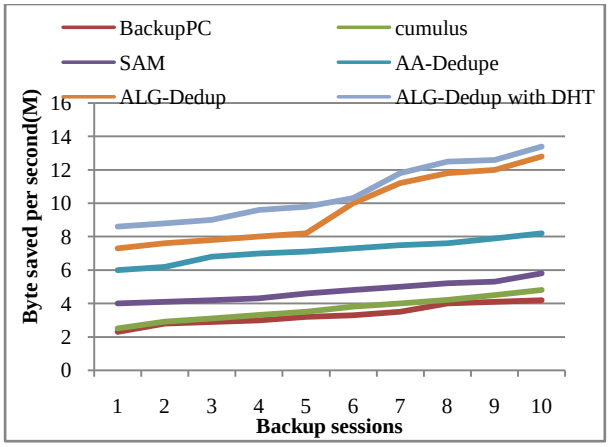

Fig.3. Data deduplication efficiency of the backup datasets

Present a comparison of the six cloud backup schemes in terms of deduplication efficiency in Fig. 3, and employ our proposed new metric of "bytes saved per second", to measure the efficiency of different deduplication approaches in the same cloud storage platform. ALG-Dedupe with DHT performs much better than other backup schemes in the deduplication efficiency measure with a low overhead,since load balancing is done in well organized manner using DHT. This significant advantage of ALGDedupe is primarily attributed to its application awareness and global duplicate detection in the deduplication process.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

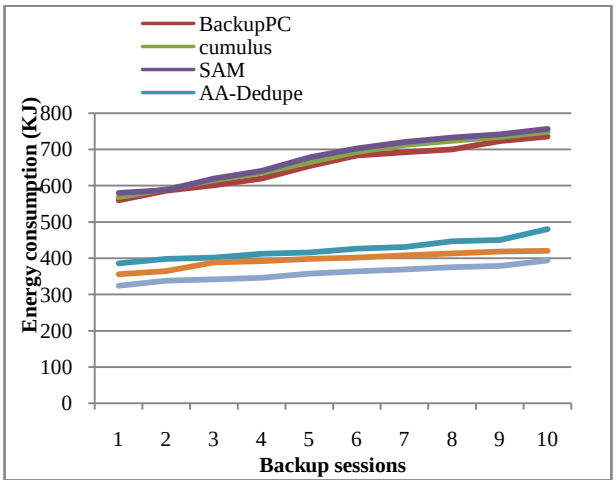

Fig.4. Energy consumption of source deduplication schemes

Fig.4 shows the energy consumptions of the cloud backup schemes as a function of backup sessions. Existing approaches incur high-level power consumptions due to their significant computational overhead during the deduplication process or high data transfer overhead owing to low space saving. ALG-Dedupe with DHT incurs only 52 percent that of SAM, BackupPC and Cumulus by adaptively using application-aware design in deduplication. Furthermore, it achieves almost the same energy consumption as its local scheme AA-Dedupe.

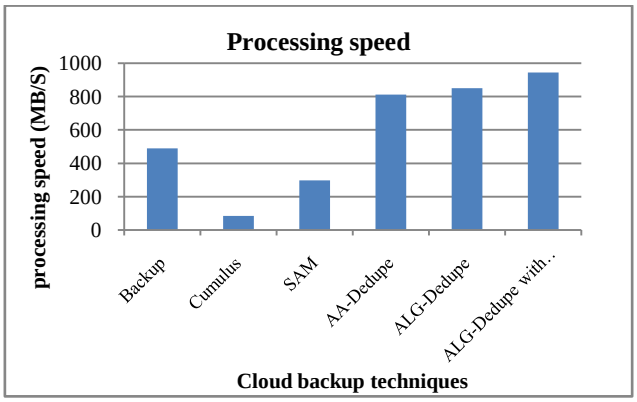

Fig. 5. Speeds of chunking and fingerprinting in PC clients.

As shown in Fig. 5, our ALGDedupe with DHT can achieve more than 1.9 ~ 13 times speed of all application-oblivious schemes perform better than the other methods in processing speed due to the involvement of global deduplication. It is well known that chunk fingerprint index cost the main RAM usage for local deduplication in client.

VI. CONCLUSION AND FUTURE WORK

This paper is based on cloud computing technology which has a very vast potential and is still unexplored. The capabilities of cloud computing are Interminable. Cloud computing provides everything to the user as a service which includes platform as a service, application as a service, infrastructure as a service. One of the major issues of the cloud computing system is that the storage of cloud backup data in personal computing environment and balancing the tasks in the environment, since overloading of a system may lead to poor performance which can make the technology unsuccessful. In order to overcome these problem in the personal computing environment in this work presents a novel

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 3, March 2015

ALG-Dedupe , deduplication scheme for cloud backup in both local and global cloud storage space to exploit file semantics to minimize computational overhead. In order to manage load balancing between the global and local deduplicationfiles in the backup service distributed Hash Table is proposed. The proposed distributed Hash Table scheme works with high probability and that its cost measured in the number of migrated nodes is comparable to the best possible to reduce the cloud storage usage. In our future work we will discuss Simulated Annealing for global search optimization in cloud load balancing and simulation. Some additional algorithms which can help in solving some sub-problems in load balancing which are applicable to cloud computing

.

REFERENCES

1. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, 'Above the clouds: A berkeley view of cloud computing', EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, February 2009.
2. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, 'Reclaiming space from duplicate files in a serverless distributed file system', Distributed Computing Systems, International Conference on, Vol. 0, pp. 617, 2002.
3. D. Russell, Data Deduplication Will Be Even Bigger in 2010, Gartner, February 2010.
4. H. S. Gunawi, N. Agrawal, A. C. Arpaci-Dusseau, R. H. Arpaci- Dusseau, and J. Schindler, 'Deconstructing commodity storage clusters', in ISCA '05: Proceedings of the 32nd annual international symposium on Computer Architecture. Washington, DC, USA: IEEE Computer Society, pp. 60–71, 2005.
5. S. Quinlan and S. Dorward, 'Venti: a new approach to archival storage', in First USENIX conference on File and Storage Technologies, Monterey, CA, 2002.
6. L. L. You, K. T. Pollack, and D. D. E. Long, 'Deep store: An archival storage system architecture', In Proceedings of the 21st International Conference on Data Engineering IEEE, pp. 804–815, 2005.
7. Y. Fu, H. Jiang, N. Xiao, L. Tian, and F. Liu, 'AA-Dedupe: An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment', in Proceedings 13th IEEE International Conference on Cluster Computer, pp. 112-120, 2011.
8. P. Anderson and L. Zhang, 'Fast and Secure Laptop Backups with Encrypted De-Duplication', in Proceedings 24th International Conference LISA, pp. 29-40, 2010.
9. S. Kannan, A. Gavrilovska, and K. Schwan, 'Cloud4HomeV Enhancing Data Services with @Home Clouds', in Proc. 31st ICDCS, pp. 539-548, 2011.
10. Bhasker Prasad Rimal, Eummi Choi, Lan Lump, 'A Taxonomy and Survey of Cloud Computing System', 5th International Joint Conference on INC, IMS and IDC, IEEE Explore, pp. 44-51, 2009.
11. CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services, the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, 2011.
12. S. Rhea, R. Cox, and A. Pesterev, 'Fast, inexpensive content-addressed storage in foundation', In USENIX ATC'08, Berkeley, CA, USA, USENIX Association, pp.143–156, 2008.
13. C. Alvarez, 'NetApp Deduplication for FAS and V-Series Deployment and Implementation Guide', NetApp. Technical Report TR-3505 , 2011.
14. B. Zhu, K. Li, and H. Patterson, 'Avoiding the disk bottleneck in the data domain deduplication file system', In FAST'08, pages 1–14, 2008.
15. J. Wei, H. Jiang, K. Zhou, and D. Feng, 'MAD2: A scalable high-throughput exact deduplication approach for network backup services' In IEEE MSST'10, pages 1–14, 2010.
16. T. Yang, H. Jiang, D. Feng, Z. Niu, K. Zhou, and Y.Wan, 'Debar: A scalable high-performance de-duplication storage system for backup and archiving', In IEEE IPDPS, pp. 1–12, 2010.
17. F. Guo and P. Efstathopoulos, 'Building a high performance deduplication system', In USENIX ATC'11, pp.25–25, 2011.
18. D. T. Meyer, and W. J. Bolosky, 'A study of practical deduplication. ACM Transactions on Storage (TOS)', Vol.7, Issue.4, pp.1-14. 2012.
19. D.Shue, M. J. Freedman and A. Shaikh, 'Performance Isolation and Fairness for Multi-Tenant Cloud Storage', In OSDI, Vol. 12, pp. 349-362, 2012.
20. AnawatChankhunthod, Peter Danzig, Chuck Neerdaels, Michael Schwartz, and Kurt Worrell, 'A Hierarchical Internet Object Cache', In Proceedings of USENIX Annual Technical Conference, San Diego, CA, January 1996.
21. D. Meister and A. Brinkmann, 'Multi-Level Comparison of Data Deduplication in a Backup Scenario,' in Proc. 2nd Annu. Int'l SYSTOR, pp. 1-8, 2009.
22. K. Eshghi, 'A Framework for Analyzing and Improving Content Based Chunking Algorithms', HP Laboratories, Palo Alto, CA,USA, Tech. Rep, 2005.
23. M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, 'Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality', in Proc. 7th USENIX Conf. FAST, pp. 111-123,2009.

BIOGRAPHY

D.K.Karthika received the Bachelor of Technology in Information Technology from the Anna University Chennai in 2012,where she is currently pursuing the Master of Engineering Degree in Computer Science And Engineering in Anna University,Chennai.