Improve Speed Efficiency and Maintain Data Integrity of Dynamic Big Data by using Map Reduce

Sapna R. Kadam
PG Student
MBESociety's of college of engineering,Ambajogai
Maharashtra, India

B.M. Patil
Professor
MBESociety's of college of engineering,Ambajogai
Maharashtra,India

V.M. Chandode
Associate professor
MBESociety's of college of engineering,Ambajogai
Maharashtra,India

ABSTRACT
Cloud computing has rapid growth globally cause of the facet provided by the service not only scalability but also capacity management that subject to storage huge amount of data. Major issue will going to arrived at the time of storing this much bulky data on a cloud because data integrity may lost at the time of data retrieval.First, Anyone canister to challenge in the intention to verification of data integrity of certain file so that appropriate authentication process will going to miss between cloud service provider and third party auditor(TPA). Second, as the BLS signature obligated for fully dynamic updates of data over data blocks of fixed sized which causes re-computation and updating for an entire block of authenticator which origin not only higher storage but also communication overheads. In order to keep security as a vital issue because malicious party may scarf data at the time of data flows this can be addressed by means of symmetric key encryption. Similarly, in order to increase the speed and efficiency at the time of data retrieval for huge amount of data MapReduce plays vital role and the because of replication over the HDFS maintain data integrity with the full support of dynamic updates.

Keywords
Cloud computing, authorized auditing, big data, Hadoop, provable data possession, fine-grained updates

1. INTRODUCTION
Cloud computing is new invention dispersed computing platform that awfully valuable not only for big data storage but also for processing[1]. Cloud computing fetch wonderful advantages as compare to traditional distributed system. Cloud computing is the converging technology as it alias backbone to get rid of the big data connected problems. Especially scalability and elasticity [2] make cloud the supreme platform for processing big data streams also for managing big data appliance complexities. Datasets are always dynamic in big data hence security is the major distress. Many big data appliance have been drifted into cloud. 'X as a Service'(XaaS),in same way Infrastructure-as-a-Service (IaaS), and including Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) are the nucleus concept of cloud which means not only individual but also enterprise users can utilize IT service as pay-as-you-go model fashion[3].

1.1 Security and Confidentiality aspect in cloud
Security is the primary concern regarding utilization of cloud computing[4]. As data is not having the power over user's direct control, as they are averse to move their important data over the cloud especially the public cloud with its highly merge and multi-tenancy[5]. Also, from an efficiency aspect, querying and retrieving from cloud server need lot of efforts than in local server. The stored and maintained data should be novel is the main focus on integrity of data. The vigorous research area are the integrity and defense of data the problems regarding this area have been studied at past .Integrity can violate by oblivious malicious attack.

1.2 Dynamic big data public auditing
Especially with the intension of integrity assurance, problem in the recent year is public auditing of cloud data[6]. The datasets are not in origin means it is out of reach for cloud user's which is going to store on cloud storage server(CSS) auditing by client or a third party auditor is a ordinary request doesn't matter how strong server-side mechanism are stated [7].

.

Majority datasets in application of big data are dynamic hence public auditing has a crucial importance to be scalable and competent to prop up dynamic or vibrant updates for data. The current work concentrate on data integrity which is concerned with ensuring that data is stored and maintained in its original form in efficient way with fine grained updates.

2. RELATED WORK
In this latest epoch, enlargement of distributed system or as an alternative cyber infrastructures has been crucial platform for processing the huge amount of data. To get rid from all these big data problems the cloud presently regard as most powerful effective and lucrative platform. Privacy and security are the two sides of one coin even though both of them aspire for protection and integrity.

Wang et al. [6] offer one scheme which is based on BLS signature which supports public auditing by third party auditor. Latest work on public auditing of data with full data dynamics support. Though, this scheme not having complete support for both fine-grained updates as well as authorized auditing. Newest work has proposed by Wang et al. [7] added a random masking scheme on top of [6] to make sure the TPA cannot conclude the raw data file from a sequence of integrity proof.

Authenticity of data nothing but integrity of data has fascinated research concern. Jules et al. [8] was the founder of Proofs of irretrievability (POR). Unfortunately hitch of POR model is it will support only static data storage as like archive and library. Ateniese et al. [9] is the originator of 'provable data possession' (PDP) schemes which tender mainly 'blockless verification' which used to work in terms of verification as like verifier be able to validate outsourced file data integrity of a proportion just by checking combination of pre-computed tags of file which mainly known as homomorphic verifiable tags (HVTs) otherwise homomorphic linear authenticators (HLAs) and need entire file as a proof.

Ateniese further analysed remote data checking by using PDP scheme which gives proof for file stored by TPA called 'spot checking' that permit server to access small portion of file[10] and also ropes large data sets in widely-distributed storage systems. Later Ateniese et al.[11] invent enhanced PDP scheme when the server is low it oppeses the linear value in the data size also supports only partial data dynamics and predefined number of challenges. Curtmola et al. [12] presents the MR-PDP shemes which is able to prove integrity for the original data file along with multiple replicas but it does not support verification of the dynamic data updates.

In 2009, Erway et al. [13] has projected pioneer integrity verification scheme '''Dynamic provable data possession' (DPDP)which has used authenticated data structure that is rank based authenticated skip list for verification updates. On the other hand, public auditing ability as well as variable-sized blocks of file not giving support by default[13].Ateniese have present how to renovate a mutual identification protocol to a PDP scheme. In 2012, Zhu et al. [14] have present method which permits different service provider in a hybrid cloud to kindly prove data integrity to the data owner. Afterward problem has found for this scheme as sharing of cloud data is occurred in many scenarios.

Shacham [15] have studied on PDP and POR unified scheme and proposed initial public verification scheme which is based on BLS signature. Moreover to the basic reliability of digital signature, this scheme has a greatly condensed signature length, but also bigger overheads due to the computationally classy paring operations. When wielding the similar security strength (80-bit security) and BLS signature (160 bit) are much shorter than an RSA signature (1024 bit), which affects a shorter proof size for a POR scheme. They have also give recovery in POR model with 'stateless verification'[16].

Yao [17] have been proposed encryption-based data security protection move toward for cloud storage services in which original data input needs to be processed on the cloud side. Lot of big data application users data which stored on the cloud for small-sized frequent updates. A classic illustration is Twitter[18], where each peep is restricted to 140 characters long. They can add up to a total of 12 terabytes of data per day. Furthermore, cloud users may perhaps need to split large-scale datasets into smaller chunks before uploading to the cloud for privacy-preservation [19] or efficient scheduling [20]. In this regard, efficiency in processing small updates will affect the performance of many big data applications. Moreover this scheme supports limited types of updates.

To address big data problems, cloud computing is believed to be the most potent platform. In Australia, big companies such as Vodafone Mobile and News Corporation are already moving their business data and its processing tasks to Amazon cloud - Amazon Web Services (AWS)[21]. Email systems of many Australian universities are using public clouds as the backbone as similarly it generates big data continuously. As security point of view this big data generation need to protect data from malicious attack as in our paper inspired how to provide security with for dynamic big data with public auditing and to protect data against malicious attack.

3. PROPOSED WORK

3.1 Authenticated key exchange in cloud

To present server side key exchange schemes which intend for supporting competent proof generation in the public auditing of cloud data. key exchange for big data which is based on randomness-reuse strategy and Internet Key Exchange (IKE) scheme commonly known as CCBKE stated by Liu et al. [22]. CCBKE scheme used for efficient as well secure data transfer in the background of the cloud for client side.

To provide security for big data in cloud Yang et al. [23] have proposed Hierarchical Key Exchange for Big data in Cloud (HKE-BC) where iterative key exchange strategy for two-phase layer-by-layer approach in use to attain more competent AKE without sacrifice the level of data security. These key exchange schemes will also help other security method which generally involve symmetric encryptions, such as security-aware scheduling still there is chances to drop security because of generation dynamic big data. Liu [24] have shown data security may get upset because of malicious TPA.

In this paper TPA getting least information on client data during auditing. There is always big potential to address security threats from other malicious user. Also acquainted with how to increase processing speed at the time data retrieval from server while other side there id generating huge amount of data.

3.2 Public Auditing Of Verifiable Fine-Grained Updates

BLS-Signature scheme design is inherently incompatible to hold variable-sized blocks, in spite of their remarkable advantage of shorter integrity proofs.In fact it supports only insertion, deletion or modification of one or multiple fixed-sized blocks, which known as 'coarse-grained' updates. This scheme is not able to handle modification and deletion in a size lesser as compare to block. CSS will create new block every time at every insertion scheme. On the other hand, when there are a huge number of small upgrades especially at the time of insertions, the amount of wasted storage of blocks will be huge.

To resolve this problem new-fangled perception come in mind which will support of fine grained updates which resolve problem of wastage of block. Also it gives support for fine grained updates for dynamic big data.

3.3 Authoritative public auditing

Figure 1 illustrate relations between participating parties i.e., both CSS and TPA only semi-trust the client three parties. In the old model, the challenge message is uncomplicated so anyone will be able send a challenge to CSS for the evidence of a certain set of file blocks, which is permit malicious in practice. In order to cause additional overheads on CSS and congestion to its network connections malicious user can commence distributed denial-of-service (DDOS) attacks by sending so many challenges from multiple clients at a time. Because of this an adversary may get privacy sensitive information from integrity proof which are compared with client selected data blocks returned by CSS. At the end, traditional PDP models unable to meet the security necessities of 'auditing-as-a service', although they support public verifiability

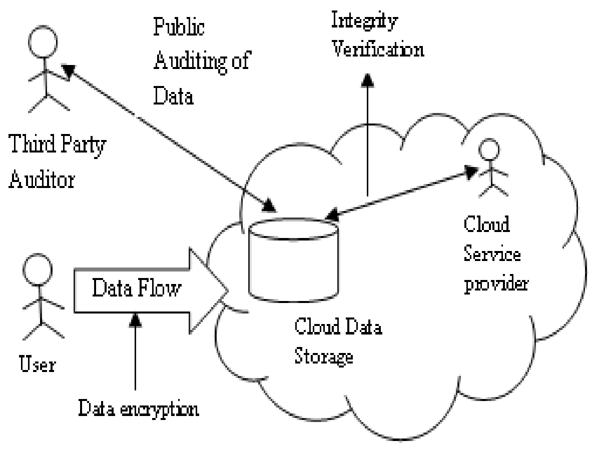

Figure 1: Relationship between the participating component in a public auditing scheme.

3.4 Security against suspicious server

Participating parties in public auditing scheme do not trust fully each other.Authenticated data structure as like MHT facilitate other parties to check content as well as updates of data blocks. The verification for a block is skilled with the data node itself and also its auxiliary authentication information (AAI) which is builded with node values on or near its verification path. Without authentication of block indices, a dishonest server can easily take another intact block and its AAI to fake a proof that might pass authentication. First,the proof of updates are no longer consistent. A deceitful server be able to store a original data block anywhere, only if it transfers back reliable pair of hash $H(m_i)$ and AAI that is capable of using to computed the correct root value. Second, for auditing of dynamic data, $H(m_i)$, the hash value of the block itself, is required in authenticator computation instead of a hash of any value that contains block indices as like $H(i)$ if not an insert/delete will basis changes to authenticators of all the following blocks, which will be terrible , especially if the client is the only one who can compute authenticators.

To overcome this problem use MHT among top-down leveling instead of RASL(Rank-Based Authenticated Skip List) to intend the new ADS(Authenticated Data Structure). To ensure server is not cheating with client with the help of another node require both client and verifier to memorize the total number of blocks as well as confirm the block index as of both directions.

4. FRAMEWORK AND ANALYSIS

All the work done in this paper has been conducted our experiments on U-Cloud .U-cloud is a cloud computing atmosphere situated in University of Technology Sydney (UTS). The computing amenities of this system are situated in several labs in the Faculty of Engineering and IT, UTS. Resting on hardware and Linux OS, also installed KVM Hypervisor [25] which virtualizes the infrastructure and permit it to provide combined computing and storage resources. Upon virtualized data centers, Hadoop [26] is installed to ease the MapReduce programming model as well as distributed file system. In addition installed OpenStack open source cloud platform [27] which is accountable for international management, resource scheduling, task distribution and interaction with users. Table 1 shows simulation parameter for the implementation.

Table 1: Simulation parameter for implementation

Simulation parameter	Values
CPU Cores	2 cores
RAM Used	4GB
Default block size	64KB
Default replication factor	3
HDFS size	23.73GB
Heap size	992.31MB
Security	80 bit

4.1 Module of implementation

Figure 2 depicted complete overview of the integrity verification module for the outsourced data which is nothing but lifecycle that generally work among two notion as like verification framework and also cloud data storage for verification.

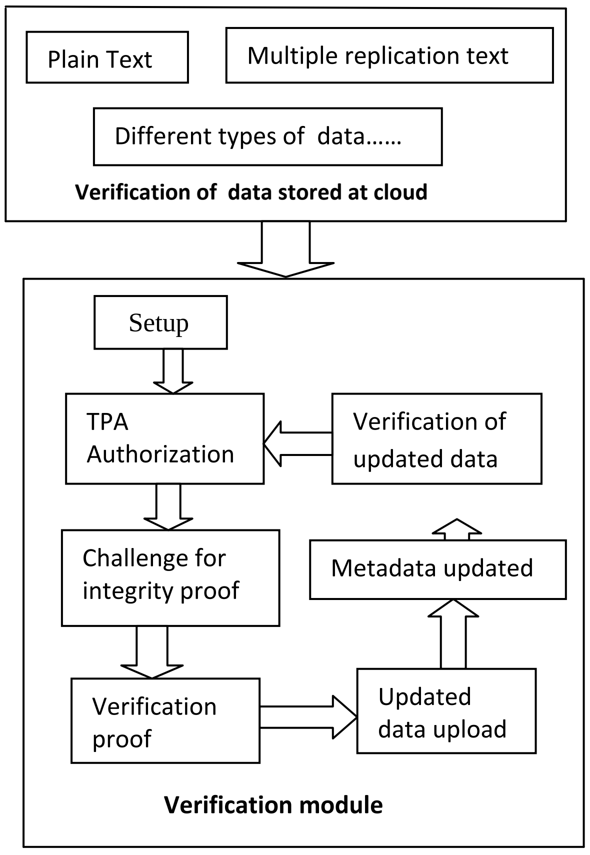
Plain Text
Multiple replication text
Different types of data……
Verification of data stored at cloud
Setup
TPA Authorization
Verification of updated data
Challenge for integrity proof
Metadata updated
Verification proof
Updated data upload
Verification module

Figure 2: Integrity verification lifecycle for cloud data framework

In cloud, generally user data will going to remotely store on CSS. At that time metadata will going to upload along with original dataset. There will be always security risk in plaintext extraction if third party will ask many time for verification over certain part of data.AT the time of Challenege nad verification of data integrity verification will fulfilled. After verification dynamic data will going to occur client require to perform updated data uploaded already which will stored on blocks. Without retrieving all the data stored or re-running the entire set up in order to keep data storage stay verifiable ,client will update verification of metadata.

As the CSS is not completely trusted ,client require verification of data update process to check if the updating of both user data as well as verification metasat has been performed successfully or not to ensure verification of updated data.

4.2 Algorithm

The algorithm will work on the basis of lifecycle of integrity verification as shown in Figure 2. shows stepwise execution how it will work for method of implementation

1. User data will store on CSS and prepare for authentication metadata.
2. Metadata will be uploaded along with original dataset.
3. CSS will perform the *UpdateReq* also parse it as {PM,i , o, m_{new}} for gathering sectors CSS will use {o,|m_{new}|} which are not added in this update
4. Then CSS will send proof of update to the client
5. After getting proof client will compute R using $H(m_i)$, Ω_i
6. Then client will check *sig* and compute m_i and it will compute R with{$H(m_i')$, Ω_i}
7. Then TPA will verify $e(sig,g) = e(H(R),v)$ also $e(\sigma,g) = e(w,v)$
8. If equation is true it will hold otherwise it will return false
9. Uploaded data at HDFS will create replicas at each node.
10. Malicious user will detect before storing the data
11. CSS will combine the cluster.
12. Heap size will vary for each file observed by client.

Description:
On CSS user data will going to store remotely. At that time client need to plan verification of metadata as HLA or HVT.
Depend upon Homomorphic signature metadata will be uploaded. CSS will perform *UpdateReq* for updating the newly uploaded data which will going to mention sectors for uploaded data for just gathering the sectors. Then client will prepare TPA authorization. The client will check updated data with the original data file for the purpose of verification. The client will send challenge message to server then server will compute response over pre-stored data. This response computation will depend upon the all the message blocks namely ‘proof of integration’. Client will perform updates to some of cloud data storage also replicas verification will be done. When data is dynamic and auditing will done by third party auditor, malicious server may swindle the client along with other blocks when challenge is tainted. For that purpose developed scheme which will do verification over pre-stored data and it will going to discover malicious user. Because of that processing speed efficiency will going to increase.

5. RESULT ANALYSIS

In Figure 3 depicted as the number of sector s are rising per block is one most significant metrics for overall performance s_i=s_{max}. Here s_{max} will only come to a decision total number of blocks is a constant according to assured file interfere and a certain success rate detection hence number of audited block as initial variable of dimension. So it can conclude that proof of size decreases when s_{max} rises, the reason for that average depth of leaf node m_i reduced when s_{max} rises to a certain level especially when right after the initial uploading of file it effected like storage at css will also going to reduced with the rise in value of average number of block. For that reason, comparatively large s_{max} is proposed in our dynamic setting.

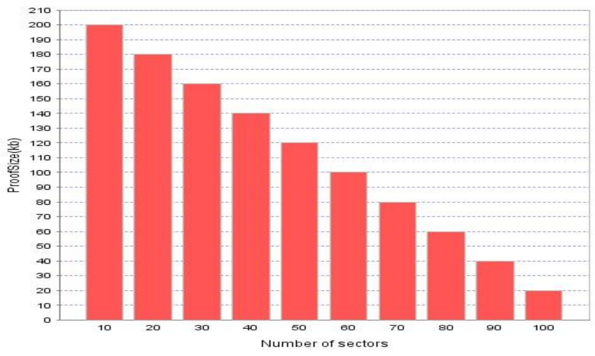

Figure 3: Communication overhead under different s_{max}.

Second storage head for small insertion with no sustain for fine grained update, creation of complete new block and update of associated MHT node cause due to every small insertion our schemes has support for dynamic updates by compare it with public auditing schemes. To get rid from all this problem storage swapping notion has inspired. It can support communication for large file block with multiple sectors each.

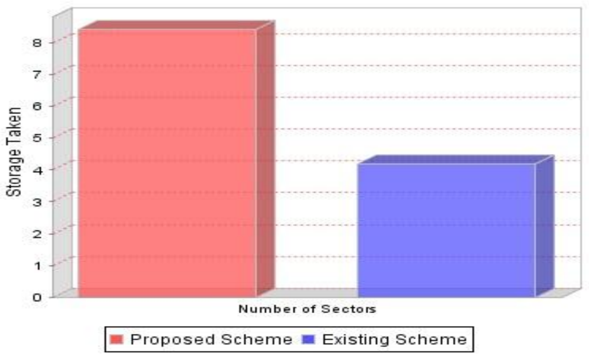

Figure 4: Storage Overhead for a particular block

The updates chosen for each sector conclude processing speed and storage taken at each sector as an efficiency see Figure 4 which shows different operation of storage according to communication overhead as per storage on data for one file on one sector. After testing first sample it bring to a close how S_{max} can influence the size of proof p which is not present in former schemes as depicted in Figure 4 by taking value as for a particular file value as total load was 8023425 for proposed scheme. After testing different size of file and uploaded on hdfs on different sectors as depicted in table 2 and shown in Figure 6.

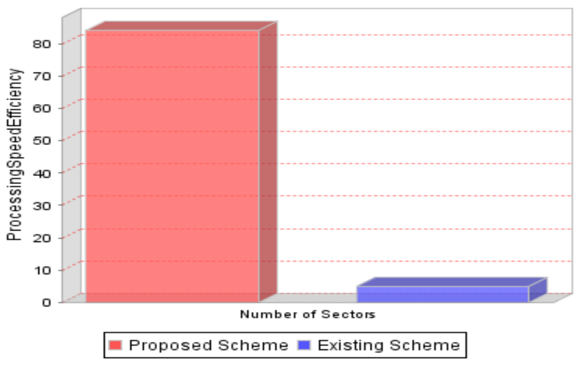

Figure 5: processing speed efficiency at the time of data retrieval for a particular sector

Processing speed efficiency will be concluded at the time of data retrieval as concern to the total load will come to server at the time of file uploaded to the DFS. Total record will arrive at the time of load, before only dfs will check for malicious TPA so processing speed efficiency will be more. Processing speed will increase because of for large amount of data are going to use MapReduce method. With the intention of testing has done on one sample at that time data load on server was 8023425 is shown in Figure 5.

If numbers of blocks are less processing speed at the time of data retrieval will be more as per sectors. First simply all the records of log file which will going to update every time will going to combine as it will combine the cluster after that it will going to store on blocks as shown in equation number. Processing speed is inversely proportional to the number of blocks. In Existing system processing speed was low because they were not using MapReduce method and also authentication was not provided for pre-stored data. scheme.

$$Processing\ speed = \frac{\text{Total number of records in log file}}{\text{Number of Blocks} * 10^3}$$

From Table 2 only shows the tested values for efficiency of processing speed represent dynamic verifiable updates for the file as per existing scheme it was constant in processing speed because it was not supporting dynamism so at the time of data retrieval testing has done for different file size of data as shown in Table 2.

Table 2: Comparative analysis for storage taken and processing speed efficiency

File Size	Storage Taken		Processing speed efficiency	
	Existing scheme	Proposed Scheme	Existing Scheme	Proposed Scheme
100KB	4.2	2.3	5	2.3
52.7KB	4.2	5.1	5	5.1
42.1KB	4.2	5.8	5	5.8
34.1KB	4.2	5.9	5	5.9
20.9	4.2	7.1	5	7.1

By taking different values for storage overhead over the large dynamic data updates as given values in Table2. According to number of sectors Proof of Size will going to increase but in existing scheme it was constant or low which was hitch of existing scheme because it will going to affect to the fine grained support. Existing scheme used to support to the coarse grained updates. But our scheme supports fine grained dynamic updates see Figure 6.

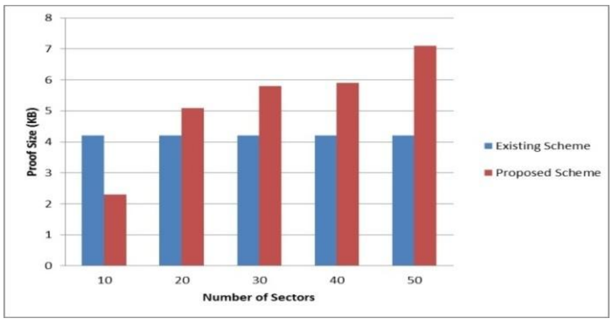

Figure 6: Comparative analysis for total storage overhead

Third investigation is processing speed efficiency and performance improvement because as the huge amount of data

are going to store on cloud big data will going to generate for that hadoop has implemented. At the time of the data retrieval from hdfs processing speed should be matter because large number of data will be there, in order boost processing speed to amplify processing speed efficiency of data retrieval MapReduced method is going to be used so it will give better processing as compare to existing scheme. So it is concluded that by taking different sample of size as shown in Table 2 and depicted in Figure 7. In existing scheme they were not using MapReduce method and However data storage on cloud was huge in amount to process the big data used Mapreduce method in our scheme see Figure 7.

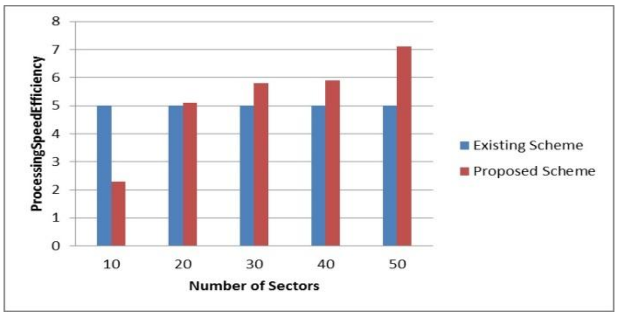

Figure 7: Comparative analysis for processing speed efficiency

Fourth, the performance upgrading the modification by testing 3 pieces of random data with size 100 bytes,140 byte and similarly 180 bytes to update quite a few blocks. Data retrieval is a mean of communication overheads in verifiable update phase. For every update, by tracing sum of data retrieval for our modified schemes also for basic schemes also as depicted in Figure 8.

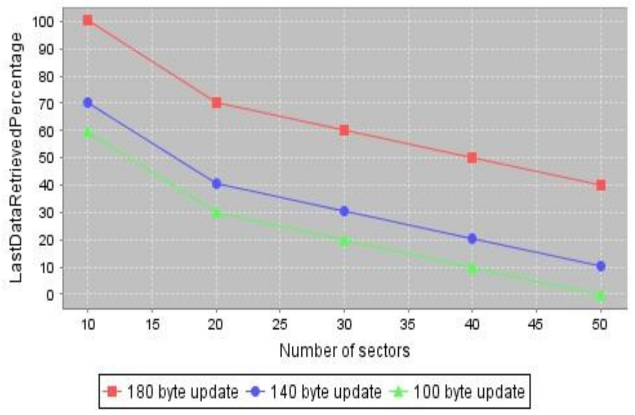

Figure 8: Percentage of communication overhead in data retrieval

From Figure 8 easily it can conclude that modified scheme has always enhanced with respect to data-retrieval-invoked communication overheads, and more advantage is for huge updates..On the other side, for an update of the same size, the benefit will diminish with the increase in S_i where huge number of sectors in novel file is required to be retrieved. So block size require keeping low if less communication in verifiable updates is highly required as depicted in Figure 8.

Along with the processing speed efficiency it is also important to see cluster summary when hadoop framework id going to be implemented. For this reason Table 4 is depicting the parameter that all used in our experiment. File size is the size of file that uploaded on HDFS so it will show how much DFS file has used and also heap size. Number of blocks will shows the number of block required for particular file as shown in Table 4.

Table 4: Percentage of cluster summary of storage of data details

No. of Block	24	7	6	5	4
File Size	100Kb	52.7KB	42.1KB	34.9KB	20.9KB
DFS Used	160.63KB	104KB	83.05KB	76KB	66.15KB
Heap Size	6.13KB	6.13KB	6.13KB	6.13KB	6.13KB

As file is going to upload on HDFS file will going to store in the block as by default size used by hadoop is 64 MB per block. For better efficiency number of blocks should be less for large file also like number of block should not be waste thatwhich is main intention of our scheme. At the time of data retrieval if number of blocks are more it will affect to the processing speed and as it gives high availability by providing replication factor by default 3. According to file size DFS usage will be conclude and heap Size also for checking our proposed scheme as performance of dynamic data generation while using hadoop framework and processing speed efficiency as depicted in Figure 9.

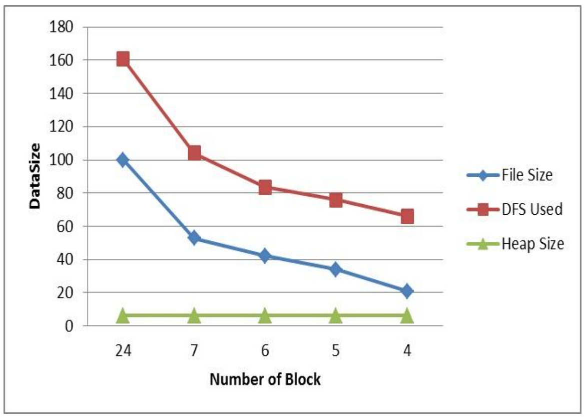

Figure 9: Cluster summary during time of storage overhead with different size of block

At the time of setting memory option for individual job there will be control over heap size. When a map task is run, the node manager will allocate a 1,024 MB container (decreasing the size of its pool by that amount for the duration of the task) and launch the task JVM configured with a 800 MB maximum heap size. In existing scheme JVM process will have a larger memory footprint than the heap size so heap size evaluation is not much good as compare to the proposed scheme. The overhead will depend on such things as the native libraries that are in use, the size of the permanent generation space, and so on. If a container utilize more memory than it has been allocated than it may be terminated by the node manager and marked as failed. So in our scheme fixed memory size for JVM heap size have been used.

Table 5: Comparative analysis for heap size used by different block

Number of block	Heap Size For File Storage	
	Existing Scheme	Proposed Scheme
24	6.13	3.51
7	6.13	2.19
6	6.13	2.05
5	6.13	1.90
4	6.13	1.46

As per testing different sample see Table 5 heap size per block will be concluded like it should be less as compare to the total heap size available in HDFS also in Figure 10 depicted the value of testing in the form of graph. Existing scheme was not prop up dynamic data access now consistent with proposed system before storing data on cluster, it will be verifying malicious user by using key so that no need not to worry about the data integrity but at all aspect heap size will be matter because at the time of storage of data on HDFS number of blocks will be used as per file size as 150 byte so every time it heap size should be less.In our proposed scheme heap size will be conclude according to following equation..

Heap size = (total number of directories + files)*150 byte

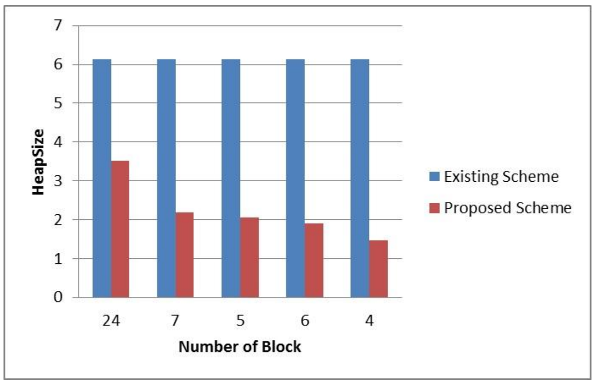

Figure 10: Comparison Storage Overhead on DFS according to different number of block.

6. CONCLUSION AND FUTURE SCOPE

To overcome the problems of public data auditing of data which will going to store on cloud in huge amount and to address the security for data and processing speed to retrieve the large amount of data to get rid from this problems development of project has done in hadoop and similarly in this paper we have shown the security series schemes also algorithms for the processing speed and best efficiency. Specially, focused on authenticated key exchange at cloud internal, dynamic fine grained update supports Third Party Auditor (TPA) for authorization ,index verification and one more combining the cluster by using labeling and commit for address the malicious user ,because of this only total load on cluster will going to reduce and efficiently processing speed will also rise efficiently. Some of the directions for future work are as follows:

1. Auditing of shared data
2. Auditing for streaming data

7. REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, ‘‘Cloud Computing and Emerging IT Platforms: Vision, Hype, Reality for Delivering Computing as the 5th Utility,’’ Future Gen. Comput. Syst., vol. 25, no. 6, pp. 599-616, June 2009.

[2] M.Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.Katz, A.Konwinski,G. Lee, D. Patterson, A. Rabkin, I. Stoica, andM. Zaharia, ‘‘AView of Cloud Computing,’’ Commun. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[3] Customer Presentations on Amazon Summit Australia, Sydney, 2012, accessed on: March 25, 2013.

[4] J. Yao, S. Chen, S.Nepal,D. Levy, and J. Zic, ‘‘TrustStore: Making Amazon S3 Trustworthy With Services Composition,’’ in Proc. 10th IEEE/ACM Int’l Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2010, pp. 600-605.

[5] D. Zissis and D. Lekkas, ‘‘Addressing Cloud Computing Security Issues,’’ Future Gen. Comput. Syst., vol. 28, no. 3, pp. 583-592, Mar. 2011.

[6] Q. Wang, C.Wang, K. Ren,W. Lou, and J. Li, ‘‘Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847-859, May 2011.

[7] C. Wang, Q. Wang, K. Ren, and W. Lou, ‘‘Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,’’ in Proc. 30st IEEE Conf. on Comput. and Commun. (INFOCOM), 2010, pp. 1-9.

[8] A. Juels and B.S. Kaliski Jr., ‘‘PORs: Proofs of Retrievability for Large Files,’’ in Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS), 2007, pp. 584-597

[9] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, ‘‘Scalable and Efficient Provable Data Possession,’’ in Proc. 4th Int’l Conf. Security and Privacy in Commun. Netw. (SecureComm), 2008, pp. 1-10.

[10] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, ‘‘Remote Data Checking Using Provable Data Possession,’’ ACM Trans. Inf. Syst. Security, vol. 14, no. 1, May 2011, Article 12.

[11] G.Ateniese, R.B. Johns,R. Curtmola, J.Herring, L. Kissner,Z. Peterson, and D. Song, ‘‘Provable Data Possession at Untrusted Stores,’’ in Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS), 2007, pp. 598-609.

[12] R. Curtmola, O. Khan, R.C. Burns, and G. Ateniese, ‘‘MR-PDP: Multiple-Replica Provable Data Possession,’’ in Proc. 28th IEEE Conf. on Distrib. Comput. Syst. (ICDCS), 2008, pp. 411-420.

[13] C. Erway, A. Ku¨ pc¸u¨ , C. Papamanthou, and R. Tamassia, ‘‘Dynamic Provable Data Possession,’’ in Proc. 16th ACM Conf. on Compute. and Commun. Security (CCS), 2009, pp. 213-222.

[14] G. Ateniese, S. Kamara, and J. Katz, ‘‘Proofs of Storage From Homomorphic Identification Protocols,’’ in Proc. 15th Int’l Conf. on Theory and Appl. of Cryptol. and Inf. Security (ASIACRYPT), 2009, pp. 319-333.

[15] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, ‘‘Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage,’’ IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 12, pp. 2231-2244, Dec. 2012.

[16] H. Shacham and B. Waters, ‘‘Compact Proofs of Retrievability,’’ in Proc. 14th Int’l Conf. on Theory and Appl. of Cryptol. and Inf. Security (ASIACRYPT), 2008, pp. 90-107.

[17] S. Nepal, S. Chen, J. Yao, and D. Thilakanathan, ‘‘DIaaS: Data Integrity as a Service in the Cloud,’’ in Proc. 4th Int’l Conf. on Cloud Computing (IEEE CLOUD), 2011, pp. 308-315.

[18] E. Naone, ‘‘What Twitter Learns From All Those Tweets,’’ in Technology Review, Sept. 2010, accessed on: March 25, 2013. [Online]. Available: http://www.technologyreview.com/view/420968/what-twitter-learns-from-all-those-tweets/

[19] Y. He, S. Barman, and J.F. Naughton, ‘‘Preventing Equivalence Attacks in Updated, Anonymized Data,’’ in Proc. 27th IEEE Int’l Conf. on Data Engineering (ICDE), 2011, pp. 529-540.

[20] X. Zhang, L.T. Yang, C. Liu, and J. Chen, ‘‘A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud,’’ IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 363-373, Feb. 2014.

[21] S.E. Schmidt, ‘‘Security and Privacy in the AWS Cloud,’’ presented at the Presentation Amazon Summit Australia, Sydney,Australia,May 2012, accessed on: March 25, 2013. [Online]. Available: http://aws.amazon.com/apac/awssummit-au/

[22] C. Liu, X. Zhang, C. Yang, and J. Chen, ‘‘CCBKEVSession Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing,’’ Future Gen. Comput. Syst.,vol. 29, no. 5, pp. 1300-1308, July 2013.

[23] C. Liu, N. Beaugeard, C. Yang, X. Zhang and J. Chen, “HKE-BC: HierarchicalKey Exchange for Secure Scheduling and Auditing of Big Data in Cloud Computing, Concurrency and Computation” Practice and Experience, accepted on 3 October, 2014.

[24] C., Chen, J., Yang, L. T., Zhang, X., Yang, C., Ranjan, R. & Ramamohanarao, K. 2014b “Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-grained Updates” IEEE Transactions on Parallel and Distributed Systems, 25, 2234 -244.

[25] KVM Hypervisor, accessed on: March 25, 2013. [Online]. Available: www.linux-kvm.org/

[26] Hadoop MapReduce. [Online]. Available: http://hadoop.apache.org

[27] OpenStack Open Source Cloud Software, accessed on: March 25, 2013. [Online]. Available: http://openstack.org/