COURSED AND HIGH PERFORMANCE BIG-FILE CLOUD STORAGE BASED ON KEY-VALUE STORE

ERUGURALLA SATHISH BABU, Assistant Professor,

Dept of Computer Science and Engineering,

VIVEKANANDA INSTITUTE OF TECHNOLOGY & SCIENCES, KARIMNAGAR,TS, INDIA.

ABSTRACT: This research proposes a new Big File Cloud (BFC) with its architecture and algorithms to solve difficult problems of cloud-based storage using the advantages of key-value stores. There are many problems when designing an efficient storage engine for cloud-based storage systems with strict requirements such as big-file processing, lightweight meta-data, low latency, parallel I/O, deduplication, distributed, high scalability. Key value stores have many advantages and outperform traditional relational database in storing data for heavy load systems. This paper contributes a low-complicated, fixed-size meta-data design, which supports fast and highly-concurrent, distributed file I/O, several algorithms for resume able upload, download and simple data deduplication method for static data. This research applies the advantages of ZDB - an in-house key-value store which was optimized with auto-increment integer keys for solving big-file storage problems efficiently. The results can be used for building scalable distributed data cloud storage that support big-files with sizes up to several terabytes.

Key words: Cloud computing, Servers, Distributed databases, Metadata, Mobile communication, Google, Complexity theory.

1. INTRODUCTION

Cloud-based breaking point benefits regularly serve a significant number of clients with confine compel concerning every client can reach to two or three gigabytes to terabytes of information. Individuals utilize scattered limit with respect to the well ordered requests, for instance going down information, sharing records to their sidekicks by strategies for social relationship, for example, Face book 14, Zing Me 4. Clients likewise presumably trade information from a broad assortment of sorts of contraptions, for example, PC, telephone or tablet. Beginning now and into the not so distant, they can download or share them to others. Structure stack in an appropriated storing is commonly to an awesome degree significant. Along these lines, to ensure an OK nature of association for clients, the framework needs to go up against different troublesome issues and necessities: Serving power information advantage for unending without bottle-neck; Storing, recovering and overseeing enormous records in the structure satisfactorily; Parallel and resumable trading and downloading; Data deduplication to diminish the mishandle of storage room caused by securing a similar static information from various clients. In standard file structures, there are various difficulties for advantage build while overseeing interminable record: How relating framework for the bewildering change of information; The inconveniences in passing on information in a broad number of focuses; Effective methodology to emulate information for stack altering and acclimation to inside frustration; How to store oftentimes got to information for smart I/O, control the dormancy, and so on. In many Distributed File Systems and Cloud Storages, an ordinary method for managing these issues is part tremendous record into different more minor knocks, securing them on plates or scattered focus focuses and after that administering them utilizing a meta-information structure 7, 6,29,1. Securing pieces, meta-information sufficiently and organizing a lightweight meta-information are essential issues that dispersed amassing suppliers need to confront. After quite a while of inquisitive about, we appreciated that power passed on limit associations have a complex meta-information structure; at any rate the level of meta-information is prompt to the record gage. Subsequently, the space multifaceted plan of these metadata framework is O(n) . For enormous documents which have sizes of two or three Gigabytes or Terabytes can prompt epic meta-information sizes. Also, it isn't well adaptable for gigantic records.

In this examination, we propose another huge record passed on limit building and a transcendent reaction for diminish the space flexible nature of meta-information. Key-Value stores have various awesome conditions for advantageously securing information in information concentrated associations. They a great part of the time beat conventional social databases in the cutoff of overwhelming weight and expansive scale frameworks. Beginning late, key-respect stores have an exceptional progression in both instructive and mechanical fields. They have low-dormancy reaction time and mind blowing flexibility with little and medium key-respect facilitate gage. Current key-respect stores are not proposed for plainly securing huge respects or gigantic records for our situation. We executed two or three trials in which we put entire chronicle information to key-respect store, the framework did not have magnificent execution as run of the mill for a couple of reasons: promptly, the inactivity of put/get operation for massive respects is high, along these lines it impacts distinctive simultaneous operations of key-respect store advantage and different parallel gets to various respect finish restricted. Moreover, when the respect is more important, there is no more space to spare various difficulties in basic memory for smart access operations. At long last, it is hard relating out structure when the measure of clients and information broadened. This examination is executed to manage those issues while securing huge respects or epic document utilizing key-respect stores. It brings various focal motivations behind key-respect store in information association to outline a passed on accumulating structure called Big File Cloud (BFC).

These are our contributions in this research:

- Propose a light-weight meta-information format for giant record. Each record has about a tantamount size of metadata. BFC has O(1) space diserse nature of metadata of a report, while the cross of meta-information of a record in Dropbox7, HDFS1 has space adaptable nature of O(n) where n is size of the important record. See Fig. 13
- Propose a genuine touching abnormality id for piece accumulations of records. That makes it less hard to dissipate information and scale-out the breaking point framework.
- Bring the benefits of key-respect store into goliath record information store which isn't fortified clearly for colossal respect.
- Study the key-respect stores to locate the most proper key-respect store for BFC.

These obligations are acknowledged and assessed in Big File Cloud (BFC) that serves aggregating for Zing Me 4 clients. Circle Image records of CSM Boot-Diskless † frameworks are secured in Big File Cloud. Whatever is left of this paper is shaped as take after: Section 2 is about related works examination and some examination among them and BFC. Region 3 introduces Big-File Cloud appear, detail planning, plan of BFC, information unsurprising association and estimations for parallel trade and download immense reports. District 4 assesses to locate the best fit key-respect store for proposed passed on limit building. By at that point, it looks and other doubtlessly comprehended appropriated amassing frameworks to feature the upsides of the proposed metadata design. It also has benchmarks to examine BFC and other open-source course of action, for example, HDFS, Blob of MySQL. At last, zone 5 shows the entire of this examination.

II. RELATED WORKS

CEPH29 is an appropriated accumulating stage expected that would store contrast, piece and report. Ceph is depended upon to give noteworthy execution, committed quality and adaptability. Ceph clears record assigning tables and replaces them with passing on limits, so it limits metadata and data operations. This is a principal change. Since this draws in it to fitting the confusing data get to operations, revive serialization, replication and determined quality, bewilderment domain, and recovery. CEPH uses unprecedented reason data spread work called CRUSH 30, CRUSH is a pseudo-unusual data scattering count. Its use is that it can direct take in the zone of any test by any get-together in a tremendous structure. In addition, the measure of metadata is tied in with nothing and all things considered static unless devices are intertwined or removed. CEPH breaks records into 8 MB irregularities and stores them.

Hadoop Distributed File System (HDFS) 25 is an appropriated record structure proposed to continue running on ease thing gear. The basic favored viewpoint of HDFS is charge tolerant most remote point. As a rule, a HDFS case contains hundreds or thousands concentrations which to store a to a brain boggling degree far reaching measure of data. There are two sorts of center point in a HDFS case: NameNode and DataNode which are the parts to produce a master/slave plot. A HDFS gather merges a lone NameNode and differing DataNodes. The essential manages the record system namespace and serves the archive discover the chance to request from clients. It executes record structure namespace operations, for instance, opening, closing or renaming a report or registry. Data Nodes are responsibility concerning securing application data and serving record structure read/shape requests. Decisively when a client makes another record in a HDFS event, that report will be spited to a couple of squares with a configurable size for each document (generally 64MB). Starting their ahead, HDFS copies those squares and stores them to Data Nodes. All of metadatas and trade logs will be secured at Name Node. Right when a client request to explore a record in the report structure, it send the request to Name Node to get archive metadata, by then get all bits of that document from Data Nodes. Each square of a record has a checksum regard, these regard are secured in another covered archive in HDFS namespace, when client recoups a piece, it will check this checksum regard, if not mastermind, if will find another duplicate of that piece at another Data Node. This is the way HDFS keep up an essential parcel from corrupted data from plate dissatisfaction. Near these mind blowing conditions, HDFS has a lone clarification behind disappointment - the Name Node. It has only a single Name Node in a get-together, if it crashes, the whole structure will down. Records in HDFS are make once and have completely one creator at whatever point. This obliges the game-plan throughput while applications can make data in parallel.

Punch database (ZDB) 19 is a first class key-regard store that is redesigned for auto increment Integer-key. It has a regular memory level once-finished for speedy inquiring about position of key-regard segments in data reports. ZDB invigorates steady makes, subjective read. ZDB is served in

ZDB Service using thrift custom and diffuse data using dependable hash system. In BFC, both record id and piece id are auto increment entire number keys, so it is incredible to use ZDB to store data. The upside of ZDB is lightweight memory record and execution for immense data. Absolutely when data grow, in any case it has a low inaction for read and make operations. Distinctive specific inspects attempt to enhance perceived data structures, for instance, B+tree 15 on SSD, HDD or cross breed assembling contraption. It is similarly useful for building key-regard stores on these data structures. With the diagram and plan of BFC, the chunkId of a report has a coterminous entire number range; ZDB is 'before the best to store piece data.

Circumnavigated Storage Systems (DSS) are limit structures foreseen that would handle manage condition including Local Area Network (LAN) and the Internet. In DSS, data is spread to various servers with ability to serve a significant number of customers 22. There are different sort of Distributed Storage System which can be depicted by its abilities, outline. As showed by 22, DSS can have these useful essentials: Archival, General reason File system, File Sharing, Performance, and so forth. Systems under recorded depiction give customer reinforce and recoup data limits. BFC totally sustains these purposes of imprisonment. DSS in like way give benefits as an altogether steady record structure, for instance, NFS 24 or other Distributed File System (DFS, for instance, GFS 10,11. BFC is a driving forward nonvolatile circumnavigated putting away, so it can give this most remote point in Linux by using FUSE27 and BFC client tradition. Applications store data on BFC can take awesome conditions of its five star and parallel overseeing limits.

III.BIG FILE CLOUD MODEL AND ARCHITECTURE

3.1. General Big File Model

Subsequent to inspecting numerous prevalent distributed storage framework, for example, Drop Box‡ , Google Drive§ , One Drive¶ . We can demonstrate the general enormous document framework (BFS) as take after:

BFS = {F,C,M,Aw,Ar} where:

- BFS: Big archive structure
- F : remarkable record of customer in System. It can be recognized by a fileId. BFS has ability to store gigantic number of record F.
- C = (c0, c1,...,cn) : bumps split from exceptional report F , pieces are consistently secured in key-regard store or a constant accumulating. Each protuberance can be perceive by a chunkId.
- M : Metadata that delineated the relationship of F by its pieces. We need to consider space diserse nature of this Metadata.
- Aw : Algorithm to create record F into BFS. • Ar: Algorithm to recoup substance of F from BFS by fileId.
- Using this model we can easily analyze pros and cons of a specific architecture of big file system.

3.2. Architecture Overview

The BFC structure wires four layers: Application Layer, Storage Logical Layer, Object Store Layer and Persistent Layer. Each layer contains a few made segments. They are completely appeared in Fig 2. Application Layer includes close-by programming on desktop PCs, cell phones and web interface, which engage client to trade, download and share their own particular records.

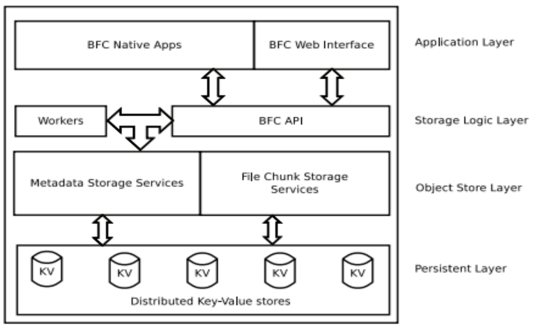

Fig. 1. BFC Architecture

Farthest point Logical Layer includes different covering associations and master associations, ID-Generator associations and every last astute Apus for Cloud Storage System. This layer finishes business legitimization part in BFC. The most essential parts of this layer are trading and download advantage. Moreover, this layer gives a high adaptable association named Cloud Apps Service which serves all customer demands. Right when the measure of customers fulfills a specific constrained ones, we can pass on Cloud Apps Service into more servers for scaling. Customers don't especially demand to Cloud Apps Service, yet through a dispatcher which gives open APIs to customers. The dispatcher checks client session before sending the customer demand to Cloud Apps Service. Besides, the dispatcher in like way checks the measure of relationship from a customer, if there are an extraordinary number of synchronous relationship from a customer, the dispatcher can piece demands from that customer. Farthest point Logical Layer stores and recovers information from Object Store Layer.

Test Store Layer is the most fundamental layer which has duty with respect to securing and sparing articles. This layer coordinates data of all articles in the structure, including client information, report data information, and particularly meta-information. In BFC structure, meta-information portrays a report and how it is managed as a quick overview of little pieces. We finished several moves up to make low-stupefied meta-information. Test Store Layer contains many appropriated backend associations. Two essential associations of Object Store Layer are File Info Service and Chunk Store Service.

Record Info Service stores data of reports. It is a key-respect store mapping information from record ID to File Info structure.

Bunch Store Service stores information pieces which are made by part from the essential records that client traded. The level of each knock is settled (the last piece of a report may have a more minute size). Part and securing an expansive file as an outline of projections in coursed key-respect store bring a great deal of central focuses. Specifically, it is less asking for to store, stream and repeat pieces in key-respect stores. Little projections can be secured beneficially in a key-respect store. It is hard to do this with a colossal file especially in neighborhood record structure. Likewise, this sponsorships trading and downloading record parallel and reusable. All information on this layer are grasped Persistent Layer in context of ZDB 19 key-respect store. There are different ZDB occasions which are sent as a scattered favorable position and can be scaled when information making. Parts in these layer are framed and typically made utilizing Zookeeper 13. Fig 1 displays the chart of BFC Architecture.

3.3. Logical Data layout

Fig 3 demonstrates the design of huge record information. Each document comprises of at least one settled size lumps. Each piece has an one of a kind whole number ID, and all of lumps which were produced from a record have a bordering scope of lump id. This is an alternate point to numerous other Cloud Service, for example, DropBox6 which utilizes SHA-223 of lump as piece ID.

3.4. Chunk Storage

Fundamental data units in the BFC conveyed capacity structure are pieces. A bump is a data area delivered from a report. Exactly when a customer exchanges an archive, if the record assess is more prominent than the outlined size, it will be part into a game plan of knots. All knots which are created from a report beside the last piece have a comparable size (the last chunk of a record may have an equal or tinier size). Starting now and into the foreseeable future, the ID generator will deliver id for the record and the essential piece with auto increment framework. Next piece in the knot set

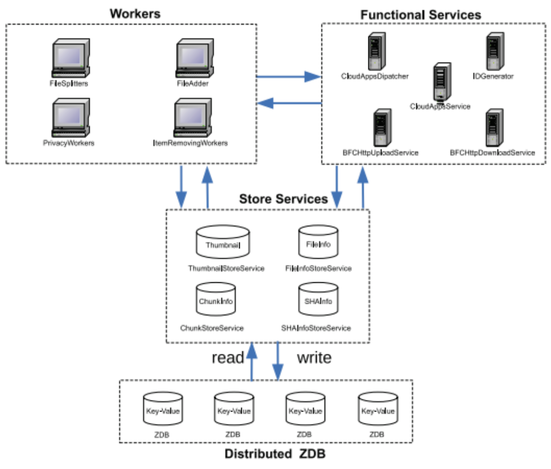

Fig. 2. BFC Main Backend Components

Will be doled out an ID sensibly reached out until the last bulge. A File Info question is made with data, for example, record id, size of the report, id of the key inconsistency, the measure of pieces and set away in a key respect store executed in ZDB Service 19. Fundamentally, the bulge will be secured in key-see store as a record with key is the id of piece and respect is piece information. Bunch amassing is a champion among the most essential procedures of BFC. By utilizing bunches to address a report, we can in actuality build up a scattered record collecting framework advantage with replication, stack altering, denounce tolerant and recuperation. Fig 4 depicts Chunk Storage System of BFC.

3.5. Metadata

Consistently, in the disseminated stockpiling system, for instance, Drop box 6, CEPH 29, the degree of meta-data will independently augment with the traverse of novel record, it contains an once-over of segments; each segment contains information, for instance, protuberance appraise, hash estimation of piece. Length of the once-over is equal to the amount of piece from archive. So it twists up detectably jumbled when the record measure is tremendous. BFC proposed an answer in which the measure of meta-data is free of number of pieces with any size of archive, both a little record and a huge record. The course of action just stores the id of first irregularity, and the amount of pieces which is delivered by extraordinary record. Since the id of piece is continuously allotted from the essential irregularity, we can without a doubt register the I th piece id by the condition:

Chunkid[i] = archive Info. Start Chunk ID+i (1)

Meta-data is basically depicted in File Info structure contain following fields: record Name - the name of report; archive ID: 8 bytes - stand-out conspicuous confirmation of report in the whole structure; sha256: 32 bytes - hash a motivator by using sha-256 computation of record data; re f File ID: 8 bytes - id of record that have past existed in System and have the same sha256 - we view these records as one, re f File ID is generous in case it is more critical than zero; start Chunk ID : 8 bytes - the ID of the essential bit of report, the accompanying bump will have id as startChunkID + 1 and so forth; num Chunk: 8 bytes - the amount of pieces of the record; archive Size : 8 bytes - size of record in bytes; status: enum 1 bytes - the status of record, it has one of each four regards named: Uploading File - when irregularity are exchanging to server, E-Completed File - when all piece are exchanged to server anyway it isn't check as unsurprising, Corrupted File - when all protuberance are exchanged to server yet it isn't relentless in the wake of checking, E Good Completed - when all piece are exchanged to server and dependable checking completed with awesome result. Along these lines the degree of File Info challenge - the meta-data of a record will be about the same for all archive in the system, paying little regard to how immense or little the report measure is (the primary differentiation meta-data of records is the length of record Name). By using this course of action, we made a lightweight meta-data arrange for while manufacturing a noteworthy record accumulating structure. Fig 5 portrays meta-data store game plan of BFC.

3.6. Data distribution and replication

Since BFC is developed in light of ZDB - a spread key-regard storing structure. Plainly the meta-data of BFC is secured passed on and can be reproduced for adjustment to inward disappointment and load-altering. Store Services, for instance, File Info Service, Chunk Store Service scatter data using enduring hashing which is proposed in16. Stay replication 28is used to reproduce key-regard data. Each kind of store advantage has its own particular flowed ZDB events. Each ZDB event has a range [hlowerbound, hupperbound) which is used to choose the extent of key to store. In case hash (key) is in the range, it is secured in that event. In BFC, record id and bump id are auto increment entire number keys. We can use direct hash work hash (key) = key for unsurprising hashing. It is definitely not hard proportional out structure for this circumstance. Fig 6 exhibits how data is scattered and duplicated in the BFC.

3.7. Uploading and deduplication algorithm

Depicts a computation for exchanging colossal archive to BFC. Data deduplication is reinforced in BFC. There are many sorts and procedures for data deduplication 26 which can work both on client side or server-side. In BFC, we executed it on server-side. We use a fundamental system with key-regard store and SHA2 hash ability to recognize duplicate records in the whole structure in the flood of exchanging. A relationship among BFC and other circulated stockpiling structures in deduplication is showed up in Table 1 in Section 4 The exchange stream on BFC appropriated capacity system has to some degree special between adaptable client and web interface. On convenient client, after an archive to exchange is picked, we call it A, the client enlists the SHA hash estimation of substance of this record. Starting now and into the foreseeable future, the client makes key information of report including record name, record evaluate, SHA regard. This principal information will be sent to server. At server-side, if data de-duplication mode is engaged, SHA regard will be used to inquiry related report ID, if there is a record ID in the system with the SHA-regard we call it B, this suggests archive An and record B are accurately the same. So we simply suggest record A to report B by dispensing the id of record B to re f File ID property of record An a property to depict that an archive is referenced to another record. The principal information will be sent back to client and the exchange stream complete, there isn't any more wasteful exchange. For the circumstance there is no report ID related with SHA-estimation of record An or data de-duplication is weakened, the structure will make some of new properties for the record information including the id of archive, the id of first piece using ID Generator and number of protuberance processed by record size and piece measure. The client will use this information to exchange record substance to the server. By then, all pieces will be exchanged to the server.

This methodology can be executed in parallel to intensify speed. Each bump will be secured in the limit system as a key-regard consolidate, with the key is the id of piece, and the regard is data substance of the irregularity. Right when all knot are exchanged to the structure, there is a procedure to affirm exchanged data, for instance, checking the state of SHA-regard figured by client

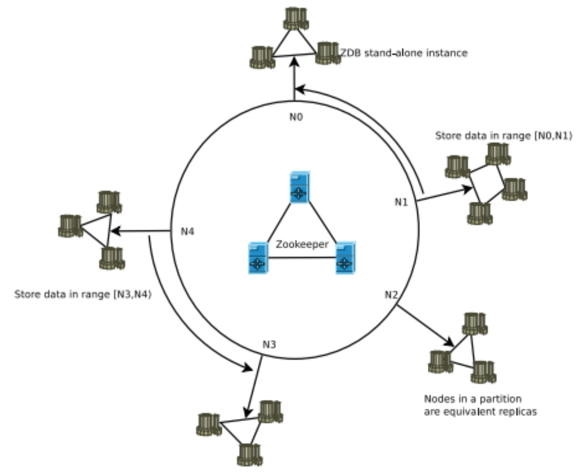

Fig. 3. Data partitioning and replication from

in addition, SHA-estimation of report made by moved bump in server. On account of everything is extraordinary, the status field of File Info is set to E Good Completed. In web-interface client exchange process, the client reliably exchanges the record to server and extras it in a concise list. By then the server figures SHA hash estimation of the exchanged record. If there is any record in the system which has the same SHA regard with it. Server will insinuate the exchanged record with this archive and empty the report at transient list. Something different, a worker advantage called File Adder will exchange record to the system using equivalent computation of the convenient application client.

IV. EVALUATION

In this piece, we quickly assess key-respect stores with proposed system of BFC to look at which is the most fitting key-respect store for BFC. By at that point, we benchmark BFC with some open source answers for securing Big-File, for example, HDFS, Cassandra, and My SQL's Blob in VNG's server develop. At last we familiarize two or three conditions with assess BFC with other individual scattered storing structures. It contains hypothetically looking, reviewing deduplication restrain. In a paper of Idilio Drago et al 5, different individual cloud stores were benchmarked in a disclosure assessment strategy. The examinations in 5 utilized files with assess: 10kB, 100kB, 1MB to explore Drop box, Sky Drive, Cloud Drive, Google Drive and Wuala. In this examination, we passed on an instance of BFC structure in Amazon EC2 to distinction and Drop box which utilizes Amazon EC2 and Amazon S3. Customers of both BFC and Drop box keep running from VietNam. As indicated by paper 6 around a few perspectives inside Drop box, we separated BFC's metadata and Drop box. By at that point, we did tests for looking cutoff of BFC and other cloud holds, for example, Google Drive, Drop box, One Drive.

4.1. Assess Key-Value store for BFC

We setup a benchmark to assess execution of key-respect stores that work in back fulfillments of BFC. This evaluation is valuable for picking the most fitting key-respect store for the course of action of BFC. We sent BFC with separate back terminations in a practically identical rigging and structure condition portrayed underneath:

1 Gigabit Network

- 04 Servers, each server has 32GB Ram, 4TBs Raid-5 HDD for storing chunks
- 02 Servers, each server has 64GB Ram, 500GB HDD for storing metadata

BFC strategy contains particular occasions of ZDB Service 19. They can be planned to utilize unmistakable key-respect store motor: Level DB, Kyoto Cabinet, ZDB, and so on. In each key-respect game-plan, we assess throughput when trade, download chronicle with various read/make stack degree. The throughput data is checked when the educational social event making when. The outcome is appeared in Fig 9. It displays that BFC utilizing ZDB has the best throughput. The throughput of BFC - ZDB is all the all the more persisting when the instructive rundown makes. ZDB is proposed to refresh securing and recovering key-respect facilitate with auto increase whole number keys. In like manner, it fits with the particular of the keys in BFC's aggregating back terminations.

4.2. Locally performance comparison

We setup a benchmark to differentiate BFC and some other open source game plan, for instance, HDFS 25, Blob of MySQL 18 and Cassandra 17. Each one of these game plans in the benchmark were sent in VNG server cultivate with a comparable gear setups, by then we gaged typical time to complete the process of examining and creating reports with different size. To store enormous reports, we expected to re-outline MySQL to change default most extraordinary field size of tables (usually each field is orchestrated to restrict size of 16MBs). Cassandra much of the time out when we make immense regard (gigantic records data) into its fragments and we ousted it from the examination.

4.3. Metadata comparison

Dropbox6 is a cloud-based limit system that empowers customers to store records, photos, accounts and distinctive reports. Drop box is available on Web interface, and many sorts of client programming's on desktop and convenient working systems. The client reinforces synchronization and sharing between contraptions with singular storing. Drop keep were generally formed Python. The nearby client uses pariahs libraries, for instance, wxWidgets, Cocoa, librsync. The fundamental inquiry in the Drop box system is a piece of 4MB data. If a record is greater than this masterminded measure, it will be part in a couple of key articles. Each basic dissent is a free segment, which is perceived by a SHA256 regard and set away in Amazon Simple Storage Service (S3). Metadata of each archive in Drop box contains an once-over of SHA256 of its pieces 7,6. In this way, its size is immediate to the measure of archive. For colossal record, it has a noteworthy metadata caused by tremendous quantities of pieces and a not inconsequential summary of SHA256 regards from them. In our examination BFC has a settled size metadata of each report, so it is less requesting to store and scale accumulating structure for enormous record.

V.CONCLUSION

BFC created a clear meta-data to make a first class Cloud Storage in light of ZDB key regard store. Each archive in the structure has a same size of meta-data, paying little personality to record evaluate. Each colossal archive set away in BFC is part into various settled size pieces (may beside the last chunk of the record). The chunks of a record have a contiguous ID run, thusly it is definitely not hard to pass on data and scale-out limit system, especially while using ZDB. This investigation similarly brings the upsides of key-regard store into gigantic record data store which isn't maintained colossal impetus as per normal procedure. ZDB19 is used for supporting progressive form, little memory-record overhead. The data deduplication method for BFC uses the SHA-2 hash work and a key-regard store to fast perceive data duplication on server-side. It is useful to save storage space and framework exchange speed when various customers exchange a comparable static data. Later on, we will continue to research and upgrade our contemplations for securing enormous data structure in a greater space of employments, especially in the "Internet of things" incline.

REFERENCES

1. D. Borthakur. Hdfs architecture guide. HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf, 2008.
2. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.
3. L. Chappell and G. Combs. Wireshark network analysis: the official Wireshark certified network analyst study guide. Protocol Analysis Institute, Chappell University, 2010.
4. V. Corporation. Zing me. http://me.zing.vn. Accessed October 28, 2014.
5. I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras. Benchmarking personal cloud storage. In Proceedings of the 2013 conference on Internet measurement conference, pages 205–212. ACM, 2013.
6. I. Drago, M. Mellia, M. M Munafo, A. Sperotto, R. Sadre, and A. Pras. Inside dropbox: understanding personal cloud storage services. In Proceedings of the 2012 ACM

conference on Internet measurement conference, pages 481–494. ACM, 2012.
7. Dropbox. Dropbox tech blog. https://tech. dropbox.com/. Accessed October 28, 2014.
8. P. FIPS. 197: the official aes standard. Figure2: Working scheme with four LFSRs and their IV generation LFSR1 LFSR, 2, 2001.
9. S. Ghemawat and J. Dean. Leveldb is a fast keyvalue storage library written at google that provides an ordered mapping from string keys to string values. https://github.com/google/leveldb. Accessed November 2, 2014.
10. S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. SIGOPS Oper. Syst. Rev., 37(5):29–43, Oct. 2003.
11. S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.
12. Y. Gu and R. L. Grossman. Udt: Udp-based data transfer for high-speed wide area networks. Computer Networks, 51(7):1777–1799, 2007.

AUTHOR'S PROFILE:

ERUGURALLA SATHISH BABAU, Presently working as Assistant Professor in Dept of Computer Science and Engineering, Vivekananda Institute Of Technology & Sciences, Karimnagar He is completed B.Tech(CSIT) from BVRIT, M.Tech(VLSI & ES) from KITS-Warangal. He had 10years of teaching experience and Gate Qualified in 2006. His interested areas are Big Data, Data mining, Networking.