International Journal of Engineering In Advanced Research Science and Technology
ISSN: 2278-256

April 2016
VOLUME -1
ISSUE-4
Page:7899-06

BFC: HIGH-PERFORMANCE DISTRIBUTED BIG-FILE CLOUD STORAGE BASED ON KEY-VALUE STORE

[1]SK.Akbar[2]V.Rajashekar[3]BanavathuBalakrishna
[1] HOD, Department of CSE Mandava Institute of Engineering Technology Vidya Nagar, Jaggayyapet. Krishna Dist, Andhra Pradesh
[2]Assistant Professor, Department of CSE Mandava Institute of Engineering Technology Vidya Nagar, Jaggayyapet. Krishna Dist, Andhra Pradesh
[3](M-tech) Department of CSE Mandava Institute of Engineering Technology Vidya Nagar, Jaggayyapet. Krishna Dist, Andhra Pradesh

Abstract: Nowadays, cloud based storage are growing and has become an emerging trendin bigdata storagefield.Manyproblemsarise while designing an efficient and low complicated storage engine for cloud-based systems with some issues like big filesprocessing, meta data, latency, parallel Input/Output, deduplication, distributednature, highscalability. Key value stores has a vital role and showed many advantages when solving those problems. This paper presents about Big File Cloud Storage(BFCS) with its modules and architecture to handle most of problems in a big file cloudstorage which his base don key value store. Here we are proposing less-complicated, fixed meta data design, which allows fast as wellas highly-concurrent, distributed fileInput/Output, and simple file and datade-duplication methodforstaticdata.Thismethodcanbeusedtobuildadistributedstoragesystemthatcanaccommodatedata whose size is uptoterabytes.

Keywords: Cloud Storage System, Key value, Big File, Distributed StorageSystem

1.Introduction

Now-a-days cloud storage system are being used forstoringthe data in gigabytes and terabytes. Cloud storage is usedforthe daily use, for backing-up data, sharing file totheircolleagues,onthesocialnetworkingsites.Theuser ofthecloudbasedsystemcanuploadthedataonthesystem andcanshareitwithothersandmakeitavailablefortheman dlater can download it. The load over the system isveryheavy. Hence, to ensure a good quality of service cloudusers, the system has to look over various requirementanddifficult problems: serving services to the user withhighquality without any bottleneck; efficiently storing,retrievingand managing the big data files; resumable andparalleldownload and upload of data; the deduplication to betakencare of for managing the storage capacity of thesystem.Traditional file-systems had to face many challengesforservicebuilderwhenmanagingahugenum berofbig-file:How to scale system; How to do distribution of data onalarge number of nodes; How to do replication data forload-balancing and fault-tolerance. The solution fortheseproblems is Distributed File Systems and CloudStoragesusing commonly is splitting big file to multiplesmallerchunks, storing them on disks or distributed nodes andthenmanaging them using a meta-data system. Storing ofthechunks and meta-data related to it efficiently and designingalightweightmeta-datarelatedtoitaresignificantproblemsthat cloud storage providers have toface.

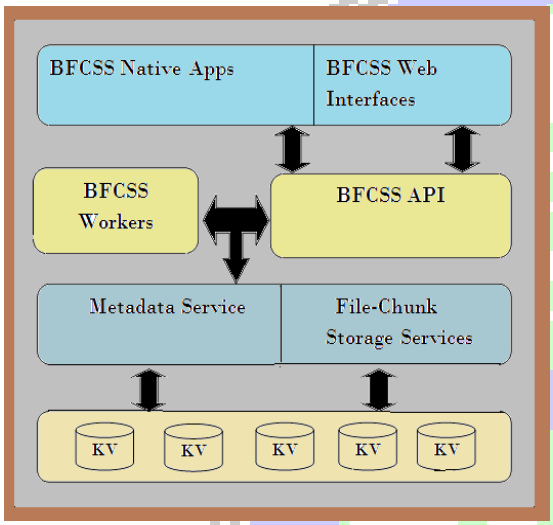
Key value stores have various advantages for storing dataindata-intensive operation. In recent years, key valuestoreshave a very unpre-cedented growth in every field. Theyhavelow latency with less response time and high scalabilitywithsmall and medium key value pair size. Current keyvaluestores are not designed for directly storing big-values, orbigfile in our case. We executed several experiments inwhichwe put whole file-data to key value store, the system didnothave good performance as usual for many reasons: firstly,thelatency of put/get operation for big-values is high, thusitaffects other parallel operations of key value storeserviceandmultipleconcurrentaccessestodifferent value.And,when the value is big, then there is no space to cacheobjectsin memory for fast access. Finally, it is difficult toscale-outsystem when number of users and data increase. Thisresearch is

implemented to solve those problemswhenstoringbig-valuesorbig-fileusingkeyvaluestores.Ithasand gets many advantages of key value store indatamanagement to research called cloud-storage systemcalledBig File Cloud Storage(BFCS).

2. Big File Cloud Storage (BFCS)Architecture

A. Overview of theArchitecture

BFCS System includes four layers: ApplicationLayer,Logical Layer, File-Chunk Store Layer and Key



valuestoreLayer. Each layer of the architecture contains severalco-ordinated components. Application Layer consistsofapplication software on desktop computers, mobiledevicesand web-interface, that allows the user to upload,downloadtheir files. This layer uses API contained in LogicalLayerand uses several algorithms for downloading anduploadingprocess which are described in subsections II-F andII-G.Logical Layer consisted of many services andworkerservices, ID-Generator services and all logical API forCloudStorageSystem.Thislayergivesthebusinesslogicpartin BFCSS.Thevitalcomponentsofthislayerareupload anddownload.Logical Layer stores and retrieves data from File-ChunkStore Layer. File-Chunk Store Layer is the mostimportantlayerwhichhasresponsibilityforstoringandcachingchunks.This layer manages information of all chunks in thesystemincluding user details and file metadata. In this,meta-datadescribesafileandhowitisorganizedinchunks.File-ChunkStore Layer also contains many distributed back-endservices. Two important services of File-Chunk StoreLayerare FileInformationService and Chunk StorageService.

Figure1:ShowstheoverviewofBFCSSArchitecture

FileInformationServicestoresinformationoffiles.Itisa key value store mapping data from fileID toFileInformstructure. Chunk Storage Service stores data chunkswhichare created by splitting the original files that useruploaded.Splitting and storing a large file as number of chunksindistributed key value store bring a lot of benefits. Firstly, itiseasier to store, distribute chunks in key value stores.Filechunkscanbestoredefficientlyinakeyvaluestore.Itisdifficult to do this with a large file directly in localfilesystem.

B. FileDescription

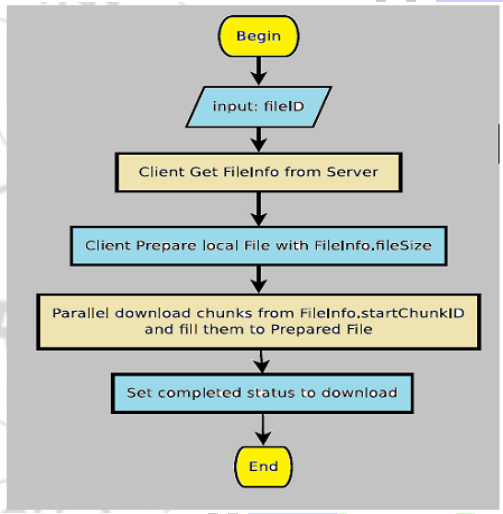
Fileconsistsofoneormorechunkswithfixed-size.Eachchunk has a unique integer Identity, and all ofchunkgeneratedfromafilehaveacontiguousrangeofchunk-id.This is a different point to many other Cloud Service suchasDropBox[12] which uses SHA-2[16] of chunk asID.

C. Storage of theChunks

The basic element in the defined cloud storage systemischunk. A chunk is generated from a file. When theuseruploadsafile,itwillbesplitintoanumberofchunks.Allchunks which are generated from a file except the lastchunkhave the same size (the last chunk of a file may have anequalor smaller size). After that, the ID generator will generateidfor the first chunk with auto-increment mechanism.Nextchunk that follows in the chunks set is to be assigned withanID and then gradually increase till the final chunk.AFileInform object is created with information such asfile-id,sizeoffile,idoffirstchunk,numberofchunksandwillbestoredtothedatabaseandthechunkswillbestoredinkeyvaluestoreasarecordwithkeyasidofchunkandvalueisdata of chunk. Chunk storage is one of the mostsignificanceof defines cloud storage. By using chunks to represent afile,wecaneasilybuildadistributedfilestoragesystemservicewith replication, load balancing, fault-tolerant andsupportingrecovery.

D. Metadata

Typically, in the cloud storage system such as Dropbox[12],the size of meta-data will respectively increase with thesizeoforiginalfile,itcontainsalistofelements,eachelementcontainsinformationsuchaschunksize,hashvalueofchunk.Lengthofthelistisequaltothenumberofchunkfromfile.Soitbecomescomplicatedwhenthefilesizeisbig.BFCSproposed a solution in which the size of meta-dataisindependent of number of chunks with any size of file, bothaverysmallfileorahugefile.Thesolutionjuststoresth



eidoffirst chunk, and the number of chunks which is generatedbyoriginal file. Because the id of chunk is increasinglyassignedfrom the first chunk, we can easily calculate the ith chunkidby theformula:

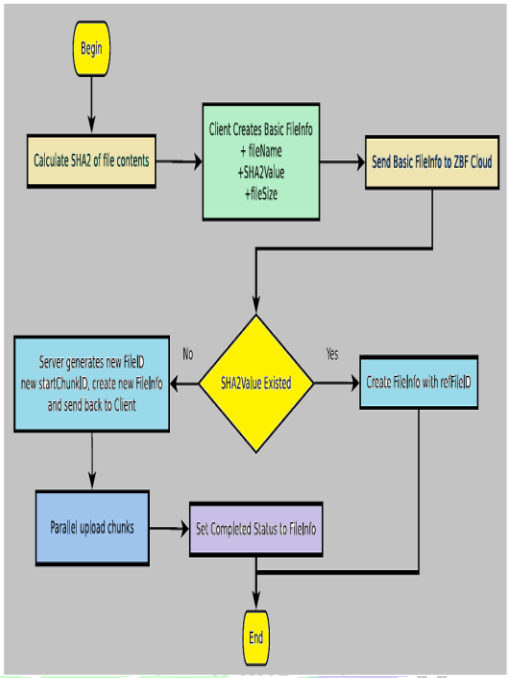
Chunk_id[i]=fileInform.startChunk_id+i

Meta-data is mainly described in FileInform structureconsistof followingfields:

- File_Name - the name offile;
- file_id:- uniqueidentificationoffileinthewholesystem;
- sha:- hashvaluebyusingSHAalgorithmoffiledata;
- reference_file:- id of file that have previous existedinSystem and have the same sha256 - we treat these filesasone, reference_file is valid if it is greater thanzero;
- start_Chunkid:- theidentificationofthefirstchunkoffile,thenextchunkwillhaveidasstart_Chunkid+1andsoon;
- num_Chunk:-thenumberofchunksofthefile;
- file_Size :- size of file inbytes;
- file_status:- the status of file, it has one in fourvaluesnamely

UploadingFile - when chunk are uploading toserver;CompletedFile - when all chunk are uploaded to server butitis not check asconsistent;

CorruptedFile - when all chunk are uploaded to server butitis not consistent afterchecking;



GoodCompleted - when all chunk are uploaded to serverandconsistent checking completed with good result. Byusingthis solution, we can create a lightweight meta-datadesignwhen building the defined cloudstorage.

E. Uploading and DeduplicationMechanism

Figure 2 describes an algorithm for uploading big filetoBFCS.Datadeduplication can be defined in the cloudstorageBFCS. There are many types and methods ofdatadeduplication [3] which can work both on client-sideorserver-side. We use a simple method with SHA2hashfunction to detect duplicate files in the system duringtheuploading offile.The upload service on BFCS cloud storage system has alittledifferent between mobile client and web interface.

TheclientcomputestheSHAhashvalueofdatacontentof thisfileP.After that, the client creates a metadata of file includingfilename, file size, SHA value. This information will be senttoserver. At server-side, if data deduplication is enabled,SHAvalue will be usedt

see associated file_id, if there is a file_id in the system with

the SHA-value we call it Q, this means that file P and file Q

are the same. So we simply refer file P to file Q by assigning the id of file B to reference_file property of file P - a

property that describes that a file is referenced to another file,thus the upload flow complete, there is no more wasteful upload of file. In the case there is no fileID associated with SHA-value of file P or data deduplication is disabled, the system will create some of new properties for the file information including the id of file, the id of first chunk using id_Generator and number of chunk calculated by file size and chunk size. This process can be done in parallel to maximize speed of operation. Every chunk will be stored in the BFCS storage system as a key value pair.

F. Downloading Mechanism

Figure 3 describes an algorithm for uploading big file to BFCS. Firstly, the client selects the id of file that will be downloaded to the server. If FileInform of the file_id exists, this information will be sent back to the client. The client uses the FileInform information to schedule the download process. Every downloaded chunk will be save directly to its position in this file. When all chunks are fully downloaded successful, the download process is completed

Figure 3: Download Mechanism:

3. Conclusion

BFCS, a simple meta-data to create a high performance Cloud Storage based on MYSQL key value store. Every file in the system has a same size of meta-data regardless of file-size. Every big-file stored in BFCSS is split into multiple fixed-size chunks (may except the last chunk of file). The chunks of a file have a contiguous ID range, thus it is easy to distribute data and scale-out storage system, especially when using MYSQL. This research also brings the advantages of key value store into big-file data store which is not default

supported for big-value. The data deduplication method of BFCSS uses SHA-2 hash function and a key value store to fast detect data-duplication on server-side. It is useful to save storage space and network bandwidth when many users upload the same static data.

References

[1] ThanhTrung Nguyen, Tin Khac Vu, Minh Hieu Nguyen, Ha Noi, Viet Nam, "BFCSS: High-Performance

Distributed Big-File Cloud Storage Based On Key value Store", June 1-3 2015, IEEE SNPD 2015, 978-1-4799-8676-7/15(Base Paper).

[2] T.T.Nguyen and M.H.Nguyen , "Design Sequential Chunk identity with Light weight Metadata for Big File

Cloud Storage", IJCSNS International Journal of Computer Science and Network Security, VOL.15 No.9,

September 2015.

[3] Jin Li, Xiaofeng Chen,Xinyi Huang, Shaohua Tang and Yang Xiang, Mohammad Mehedi Hassan, AbdulhameedAlelaiwi, "Secure Distributed Deduplication Systems with Improved Reliability", 2015,

10.1109/TC.2015.2401017, IEEE Transactions on Computers.

[4] ThanhTrung Nguyen · Minh Hieu Nguyen, "Zing Database: High-Performance Key value Store For

Large-Scale Storage Service", 17 August 2014, Springer - Vietnam J ComputSci (2015), DOI 10.1007/s40595-014-0027-4.

[5] Joshi Vinay Kumar, V Ravi Shankar, "De-duplication and Encryption in Cloud Storage", May 2015,

International Journal of Innovative Research in Science, Engineering and Technology, Vol. 4, Special Issue 6.

[6] Leeba Varghese , Suranya G, "Test Pattern Generation Using LFSR With Reseeding Scheme For BIST

Designs", December 2014, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 5.

[7] ChristianForfang, "Evaluation of High Performance Key value Stores", June 2014, Norwegian University of

Science and Technology.

[8] NesrineKaaniche, Maryline Laurent, "A Secure Client Side Deduplication Scheme in Cloud Storage

Environments", 2013, Institut Mines-Telecom, Telecom SudParis, UMR CNRS 5157.

[9] IdilioDrago, Enrico Bocchi, Marco Mellia, Herman Slatman, AikoPras, "Benchmarking Personal Cloud

Storage", October 23–25, 2013, ACM, 978-1-4503-1953-9/13/10.

[10] MihirBellare, SriramKeelveedhi, Thomas Ristenpart, "DupLESS: Server-Aided Encryption for Deduplicated Storage", 2013, USENIX Security Symposium.

[11] Iuon-Chang Lin and Po-ChingChien, "Data Deduplication Scheme for Cloud Storage", 2012, International Journal of Computer, Consumer andControl (IJ3C), Vol. 1, No.2.

[12] IdilioDrago, Marco Mellia, Maurizio M. Munafò, Anna Sperotto, RaminSadre, AikoPras, "Inside Dropbox: Understanding Personal Cloud Storage Services", November 14–16, 2012, ACM, 978-1-4503-XXXX-X/12/11.

[13] Russell Sears, Raghu Ramakrishnan, "bLSM: A General Purpose Log 0Structured Merge Tree", May 20–24, 2012, ACM, 978-1-4503-1247-9/12/05.

[14] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, "Bigtable: A Distributed Storage System for Structured Data", Google, Inc.

[15] DavidKarger, Eric Lehma, Tom Leighton, Matthew Levine, Daniel Lewin, RinaPanigrahy "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web".

[16] "Secure Hash Standard",Computer Systems Laboratory National Institute of Standards and Technology Gaithersburg, MD 2089, Issued April 17, 1995.

[17] MartinPlacek, RajkumarBuyya, "A Taxonomy of Distributed Storage Systems".

[18] DhrubaBorthakur, "HDFS Architecture Guide", Copyright © 2008 The Apache Software Foundation.

[19] SanjayGhemawat, Howard Gobioff, and Shun-TakLeung, "The Google File System", October 19–22, 2003, ACM, 1-58113-757-5/03/0010.