Journal of Theoretical and Applied Information Technology
31st December 2017. Vol.95. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HYBRID DATA DEDUPLICATION TECHNIQUE IN CLOUD COMPUTING FOR CLOUD STORAGE

HESHAM ABUSAIMEH [1], OMAR ISAID [2]

[1]Associate Professor of Computer Science, Middle East University, Amman, 11831 Jordan.
habusaimeh@meu.edu.jo, hesham@abusaimeh.com
[2] Developer at Damacsoft, Amman, Jordan.

E-mail: [1]habusaimeh@meu.edu.jo, hesham@abusaimeh.com, [2]omar.isaid@damacsoft.com

ABSTRACT

This paper proposed a hybrid data deduplication technique in cloud computing, that combines different types of data deduplications for satisfying different demands and requirements. The hybrid data deduplication consists of two different hybrid subsystems, each hybrid subsystem contains a file level deduplication and chunk level deduplication. The file level deduplication shows better execution time for deduplication redundant files. And, chunk level deduplication for detecting duplicated chunks among files at the data. The subsystems are: hybrid file variable size chunk level deduplication (FVCD), and hybrid file fix size chunk level deduplication (FFCD). The FVCD satisfies the requirements of users and applications that required better effectivity in reducing the size of data. While, the FFCD provides lower execution time. And, it can be tuned by changing its chunk's size. Where, increasing the chunk's size reduces the execution time. But, it decreases the effectivity of reducing the size of data.

Keywords: *Data Deduplication, Cloud Computing, Cloud Storage*

1. INTRODUCTION

The data deduplication is intelligent compression technique that aims to remove the redundant copies of data and replace it with references to original data. The data deduplication is considered intelligent compression method. Since, it removes the redundant content among the files at the data, not only the repeated content at the same file. And so, the data deduplication reduces the expenses of storage and management of large volume and diverse types of data [1,2]. On the other hand, the cloud computing provides processing and storage resources for massive processing and storage, the resources of cloud computing are pool of virtualized resources that contains different types of resources and divided into more than a cloud type and different kinds of critical entities [3,4,5]. The resource virtualization model (RVM) provides an essential contribution in enhancing the availability of the cloud, that it models the critical entities and the resources inside each entity. The RVM model has algorithms eases the convergence and maintainability of the cloud that they can categorize the resources of the cloud and maintaining variation of resources at real time [5].

The services of cloud computing can be categorized into three categories that are: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). The IaaS provides virtualized resources that enable the clients for building their own physical environment for running and deploying their applications and for storage of their data. For example, amazon web services (AWS) leases different kinds of virtual servers for the clients [3,4,6]. The PaaS provides a platform that provides a set of services for the users to run and deploy their applications. For example, Google provides the platform Google Apps that enables the corporations to set up communication and interaction services for their employees. Finally, the SaaS provides applications over the cloud for the users that provides certain functions for the users through their web browsers. For example, the QuickBooks provides accounting services for the corporations [4,7,8].

The data deduplication and cloud computing can work together for reducing the expenses of: storage, processing, and transmission of data. Because, the data deduplication removes the redundant copies of data, among the data of clients at the cloud. And, among the data of the applications that are deployed at the cloud. Consequently, the data deduplication

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

in cloud computing minimizes the cost of cloud services on the clients, and maximizes the revenues of cloud services provider. And so, nowadays data deduplication in cloud computing became hot topic and has received increasing attention from the researchers and industries [1,9].

The data deduplication has different categorization, that each categorization represents certain criteria at the data deduplication. First, the data deduplication can be categorized into post process and in-line data deduplication. At the post process data deduplication, the data is stored at the storage devices before the data deduplication. While, at the in-line data deduplication the data is stored in lining space for deduplication. And, after finishing the deduplication the data can be stored at the storage devices. Second, the source and target data deduplication, at the source data deduplication the redundant data are removed at the file system of the client. While, at the target type the data are generated at the client side and the data deduplication occurs at the servers of the server side. Finally, the most common categorization, is dividing the data into: file level, fix size chunk level, and variable size chunk level deduplication [1,10].

The file level data deduplication removes the redundant files at the data, and the chunk level deduplication divides the files into chunks and remove the redundant chunks. At the file level deduplication, a hash value is calculated for each file. The hash value is calculated using hash algorithm like: secure hash algorithm (SHA) and message digest five (MD5). And, the hash value is used for uniquely identification of the file at the system. And so, the data deduplication can seek for redundant files and replace it with references for original files. Although, the file level deduplication utilizes less memory space and consumes less execution time than the chunk level deduplication, the chunk level deduplication has better effectivity in reducing the size of data. Moreover, the file level deduplication cannot detect the incremental changes at the same file, where if two files differ only in few bytes the file level deduplication considers these files are completely different files [1,2].

The data deduplication constructs an index that contains the necessary information for seeking the duplicated file or chunk, at file level deduplication the index consists mainly of hash values of the files that already stored at the system. But, at chunk level deduplication the hash value of a file must reference the hash values of the chunks that belongs to this file [1,2].

The chunk level deduplication raised for provide better effectivity in reducing the size of data than file level deduplication. The chunk level deduplication aims for detecting the redundant chunks among different files at the data. The chunk level deduplication divides recent file into chunks, and calculates the hash value of each chunk. After that it compares the hash value of each chunk with hash values of chunks of the files that already stored at the system [1,10].

At the beginning fix size chunk level deduplication is used. And, later on it is replaced with variable size chunk level deduplication. Since, fix size chunk level deduplication suffers from the shifting problem, which reduces its effectiveness in reducing the size of data. The variable size chunk level deduplication uses more complex algorithms in dividing the files into chunks for avoiding the shifting problem. Although, variable size chunk level deduplication has better effectivity in reducing the size of data, it utilizes more memory and higher execution time than file level and fix size chunk level deduplication [1,10].

The file level and chunk level deduplication show differences in: effectivity of reducing the size of data, execution time of data deduplication, and memory utilization. The file level data deduplication is effective in removing redundant copies of files. And, the chunk level deduplication shows better effectivity in reducing the size of data, when the files at the data are related to each other. But, it consumes higher size of memory, requires longer execution time, and requires higher processing power. The variable size chunk level deduplication has better effectivity in reducing the data size from fix size chunk level deduplication. Nevertheless, fix size chunk level deduplication consumes lower resources and execution time for data deduplication [1,10,11].

This paper proposed a hybrid data deduplication in cloud computing, that is powerful data deduplication method contains different types of data deduplication to satisfy the requirements of storage large volume and diverse types of data that belongs to different types of clients and applications. The proposed hybrid data deduplication contains two different hybrid subsystems, the hybrid file variable size chunk level data deduplication (FVCD), and hybrid file fix size chunk level data deduplication (FFCD). Each hybrid subsystem uses file level deduplication and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

chunk level deduplication. Where, file level deduplication increases the performance of the system by avoiding the chunk level deduplication for recent files that are redundant files.

The rest of this paper is organized as follows: section 2 for related work that plays a major role in raising of proposed system. Section 3 includes the details about the proposed system. Where, section 3.1 is architecture of the proposed system that describes the layers of proposed system. Section 3.2 is data flow about hybrid deduplication inside the proposed system. And, section 3.3 discusses the Rabin-Karp variable size deduplication that is rolling hash algorithm used the Rabin-Karp algorithm for defining the boundaries of variable size chunks. Finally, the section 4 discusses conducted simulations for testing and obtaining results from the simulator of the proposed system. The conclusion of the work is in Section 5.

2. RELATED WORK

A deduplications systems were proposed for reducing size of data on the storage device, and reducing the number of bytes that can be transferred over the networks. The previously proposed data deduplication systems aimed to provide new architectures and schemas for solving certain problems, and satisfying certain business demands. For example, a deduplication system can be proposed for enhancing the performance of data deduplication, or improving the effectivity of private clouds that owned by certain enterprise.

2.1 Reducing Size of Data on Storage Devices

Although, the tape libraries with data deduplication is widely used nowadays for the storage of backup data. The disk-based backup appliances with deduplication compete the tape libraries with data deduplication. Because, they provide better execution time and more efficient utilization. Moreover, restoring data back at disk-based backup appliances is less time consuming, and more quick and efficient [9,11].

The data domain deduplication file system (DDFS) to avoid disk bottleneck is discussed at the reference [11], the published paper aims to avoid the bottleneck of back up jobs to increase the execution time of data deduplication at disk-based backup devices with deduplication. The published paper provided a system consists of five layers, and uses the variable size chunk level deduplication [9,11].

The Figure 1 shows the architecture of data domain file system. Where, each layer at the architecture responsible of certain function. The first layer is the interface layer that has multiple access protocols for receiving different types for data, the protocols at this layer include: network file system (NFS), common internet file system (CIFS), and virtual tape library (VTL). The second layer is the file service layer that is responsible on the file services like metadata management. The third layer is the content layer that divides the data stream into segments. The segment store layer is responsible for chunk level deduplication, it detects the duplicate chunks and keep track of references, and constructing the file recipe to restore the data stream. Which, includes mapping the bytes to the chunks and making a reference between each data block and its chunks. Moreover, the proposed system compresses the data using the Ziv-Lempel algorithm [9,11].

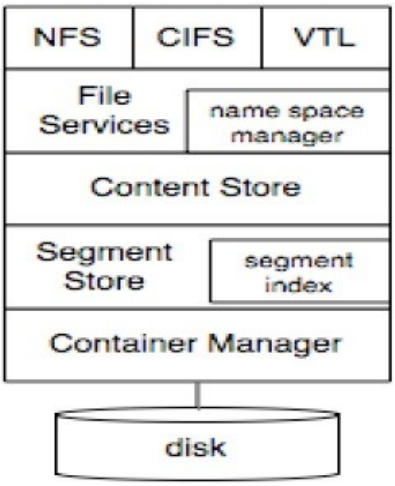

Fig. 1: Architecture Of Data Domain File System

After filling a container with the unique chunks the container is appended to the storage layer, and the indices list is updated to map each chunk to the contained container. The last layer is the storage layer that contains the containers of the data and the container manager, the containers contain the data that are ready for the storage. And, the container manager is responsible of allocation, de-allocation, reading and writing of data, and providing reliable storing for the containers [10,12].

The schema redundancy elimination at block level (REBL) is published at [13], the REBL schema is a combination of variable size chunk level deduplication and delta-encoding via resemblance detection (DERD). The variable size chunk level deduplication removes the duplicated chunks. After that, the DERD responsible of removing unused

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

data that are scattered through the chunks. Although, this schema provides better effectivity in reducing the data size, it consumes relatively higher execution time and resources [13,14].

The extreme binning data deduplication system is discussed at published paper [15], which is a scalable data deduplication system proposed for nontraditional backup system. The non-traditional backup system consists of different types of devices that have dynamic size, and the data are distributed on these devices. The aim of this research is providing new design of data deduplication that avoids the bottlenecks on the non-traditional backup system and provides better performance and throughput.

The index of extreme binning consists of two parts, to avoid loading all the records of the index at the memory without a real demand. The index of extreme binning consists of two tiers the primary index and the mini secondary index. The primary index contains the representative chunk index of the file, the hash value of the file, and the pointer to mini secondary index or the (bin). The pointer at the primary index refers the records at the mini secondary index that belongs to the chunks of the file and is the recipe for the file reconstruction [15]. The enhanced dynamic whole file deduplication (DWFD) is discussed at [16], that is a contribution for improving the efficiency of extreme binning and make it more appropriate for data backup at private cloud. Where, the private cloud has fewer resources, and it needs more efficient methods for utilization of resources. The DWFD suggests dividing the primary index into parts, each part is dedicated for data of a specific user. When, new stream of data is received for backup, the system does not need to check all the records at the index for finding a file's duplication. But, it needs to check the parts of index that belongs to the user of the newly incoming data stream.

A method of object-based deduplication is discussed at [11], that models the files of the data into a set of objects. And, applies the hash algorithm on each objects to obtain the addressable and unique identification value for each object that called (Object_ID). The provided method aims to remove the redundant data between the data flow that may not have a large space of repetition. Indeed, it was emphasized that the proposed method showed better efficiency for deduplication the unstructured data, unrelated data, and compound files more than the file level and chunk level deduplication. Because, these types of data not have a lot of commonalities.

The method of object-based deduplication consists of three main modules that are: file parser, object extractor, and duplicate object resolver. The file parser parses the data into primitive and compound objects, the primitive object is a representation of basic data structure such as image. And, the compound object is combination of more than a primitive object. The file parser defines the boundaries of the primitive objects at the compound object to be ready for extraction by object extractor. The object extractor extracts the primitive object from compound object using a recursive process, and creates the (Object_ID) for each primitive object using SHA1 algorithm. After that the object extractor classifies the object at the data depending on its type and size. Moreover, the method of object-based deduplication introduced the object size threshold to facilitate removing the redundant bytes that scattered among different objects.

The published paper introduced the duplicate object resolver that is responsible of metadata management and reference an object at the storage. The object resolver organizes the metadata of objects at B+ tree, that each node contains metadata about certain object and reference of this object at the storage. The metadata of an object includes the (ObjectID) and information about the contents of the object, which includes the size and type of data. Moreover, the (ObjectID) contains necessary information about the size of the object, and for locating the object at the storage.

2.2 Redundancy Elimination from Network Traffic

The World Wide Web (WWW) is a distributed system that has a lot of data shared by a huge number of users, the number of users that are using the WWW is increasing rapidly. Because, the WWW an efficient method for accessing large size and wide range of resources. And so, the congestion and the overloading are common problems in WWW.The data deduplication methods can be employed to improve the web performance and reduces the download time for the user [18, 19].

An architecture for analyzing the effectivity of web proxy cache in reducing the redundancy at web traffic is presented at [20]. And, it provided the protocol-independent technique for eliminating redundant network traffic, and covering the shortage of web proxy cache. The provided protocol is independent from the application protocol of the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

data flow, the provided protocol can support FTP data flow, streaming media, emails, and others.

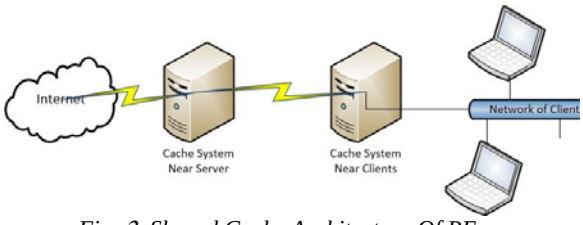

Fig. 2: Shared Cache Architecture Of RE

The Figure 2 represents the shared cache architecture of the provided system, that consists of two parts one is near to the server and the other is near to client side. The provided protocol is implemented on the cache systems for deduplication the redundant network traffic. The protocol produces a representative fingerprint for each packet. Where, the representative fingerprint represents the content of a packet and can define the similarities between two packets. And, these representative fingerprints form the index at the cache. Moreover, the data at the packets are identified using fingerprints that are integers generated using one-way hash function. The repeated regions at data stream are marked using anchors that identifying the start and end of repeated regions in data stream. The repeated regions are sent to other cache side in the form of encoding token, and the other side decode the data to be available for the users. Although, the independent protocol for data deduplication is independent and can support any type of application protocol, it is not a replacement for the proxy cache technique. Because, it consumes more computation power and memory [20].

The provided chunk and object level deduplication is provided at the reference [17], that uses the proxy cache and protocol-independent technique for eliminating redundancy (RE) at the same system. The published paper uses both systems at the same system for many reasons. First, the proxy cache cannot support all the objects of http protocol, where some objects and uncatchable according to RFC 2616 that belongs to http protocol. Second, the RE provides better efficiency and can detect redundant bytes that scattered among more than a chunk. Third, reducing the consuming of memory, where the intensive data processing of the RE is not necessary for some data flow. Forth, the RE can be deployed to work with the proxy cache, without significant modification at the WAN optimization system.

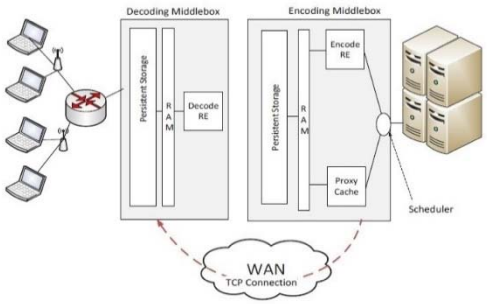

Fig. 3: Architecture Of Hybrid Redundancy Elimination For WAN Optimization

The Figure 3 shows the architecture of the system that consists from the encoding and decoding components. The architecture of the system consists of: encoding middle box, decoding middle box, and scheduler. The proxy cache is implemented using the Squid, and it has additional features for increasing the system performance. Moreover, the RE at the system is modified version of legacy RE. Where, new methods are implemented for improving RE performance. At the following there are more details about encoding components:

A-Scheduler

The scheduler works on multiple layers to define if the received object is cacheable or not, it checks the TCP buffers for the connections with port 80. The cacheable objects are sent to proxy cache and not cacheable objects are sent to RE module, and the remaining TCP/UDP traffic is sent either for the RE module that shows better saving in this context or leaves them without deduplication [16].

B-Proxy Cache Module

The proxy cache is responsible of caching the commonly requested objects, that it can prevent downloading the same content more than one time. The http proxy cache saves the web object at persistent storage, and saves the hash value of object and additional metadata in RAM. This module has additional feature called partial downloading, it saves the fragments of the object that is commonly requested instead of saving the whole object. Also, the http cache module has additional optimization feature related to the http 206 partial content response. The proxy cache can cache the fragments of the object and parse the range offset to find the duplicated objects [16].

C-The RE Cache Module

The protocol-independent technique for eliminating redundancy or RE is already discussed at this section. At the legacy RE the whole object is loaded in the memory and the hashing starts byte-by-byte, in order of increasing the performance the RE at the system has circular queue on application layer for

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

each TCP connection. Where, the content of data is stored at persistent storage and the hash values that represent the bytes at the data stream are stored at the RAM.

3. PROPOSED SYSTEM

3.1 Architecture of Proposed System

The proposed system consists of three layers the interface of the data, data deduplication layer for deduplication and restore data back, and the last layer for saving the data and metadata after deduplication or reading the data and metadata for viewing or updating the data. The Figure 4 shows the layers of the proposed system.

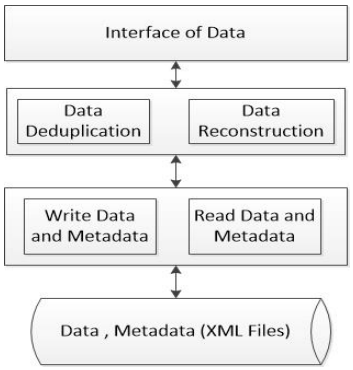

Fig. 4: Architecture Of Proposed System

The interface of data receives stream of data, and stores the bytes of stream at storage devices of cloud in form of files. The deduplication module at the second layer analyzes the data and performs the data deduplication that transforms the form of data and generates the metadata and recipes for reconstruction the data later. The third layer is the engine that responsible of saving the data and metadata after deduplication. On the other hand, the process that needs to access the data send request to interface of data that must be able to send the request to second layer. Which, sends the required parameters for third layer. The third layer loads the related metadata for loading the requested data, that are restored back using the data reconstruction module at second layer. Finally, the interface must be capable to send the data back for the user or process.

The interface of data reads the data in the form of bytes, and stores these bytes in form of files. The data deduplication module at the second layer reads the bytes of each file for defining the boundaries of chunks at these bytes, and during that the system generates metadata that will be used later by the data reconstruction module. The last layer saves the transformed bytes and keep these bytes distinguished in form of chunks. And, it saves metadata into xml files that plays major role in distinguishing saved bytes in form of chunks and reconstruction the data back. When, the user requests the data the last layer retrieves the data and metadata into a set of bytes and the reconstruction module at the second layer restores the bytes to its original form and sends these bytes to interface of data. Which, sends the bytes of data to the processes that requested the data.

3.2 Data flow at deduplication

This section describes the flow of data during the data deduplication, and shows main components and processes at the proposed system. First, file level deduplication checks a recent file from data against duplication, if the file is redundant the deduplication sets the reference to unique file instead of the file. Otherwise, the deduplication sends the file to the scheduler, that uses the user's preferences for determining the suitable type of chunk level deduplication of the file. The chunk level deduplication checks the chunks of the file against duplication, and moves to the next file at the data. This data flow is repeated for all files at the data.

The Fig. 5 shows detailed flow chart of the deduplication system. At the beginning data deduplication reads a file from the recent data, and creates the FileId of the file using the SHA1 algorithm. The file level deduplication organizes the FileId of the unique files that already stored at the system to form the index, and look up for file deduplication using the newly created FileId of the file. If the file was stored before at the system, the file is considered a unique file and is saved at the system. Otherwise, the file is considered a unique file and sent to scheduler that sends the file to the appropriate chunk level deduplication.

The fix size chunk level deduplication divides each file into fix size chunks, and the size of a chunk can be resized to accommodate with user demands. Where, there is an inverse relation between the size of a chunk from side and effectivity of reducing data size and reducing utilization of memory from another side. After tuning the chunk's size of fix size chunk level deduplication, it can provide better execution time than variable size chunk level deduplication especially it uses simpler chunking algorithm. On the other hand, the variable size chunk level deduplication divides each file into variable size chunks using rolling hash algorithm that uses complex multiplication similar to the concept of Rabin fingerprint that is used at Rabin Karp string searching algorithm.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The chunk level deduplication divides each file into chunks, and calculates the uniqueId for each chunk. At the fix size chunk level deduplication, the uniqueId of each chunk consists of (ChunkOrder.FileId.ChunkId). The (ChunkOrder) represents the order of the chunk at its file. Where, at fix size chunk level deduplication the unique and duplicate chunks can be at the same file. Since, the size of most chunks are the same. But, the uniqueId of a chunk at variable size chunk level deduplication is (FileId.ChunkId). The chunk level deduplication uses the uniqueId of each chunk at the file to look up at the index for seeking duplication. The index of each chunk level deduplication consists of the hash values (FileId) of all files at the storage, each FileId at the index references the information about the chunks of this file.

The chunk level deduplication looks up at the index to check if a chunk at the file is duplicated with another chunk belongs to another file that already stored at the system. The duplicated chunk is replaced with reference to the unique chunk, and unique chunk is prepared for storage after the deduplication. After that, the suitable metadata are prepared depending on the type of chunk. And, after checking all the chunks of newly incoming file, the FileId and information of this file is added to the index. And, the data deduplication moves to next file.

Finishing the deduplication of all files at the newly incoming data, the bytes of data are sent for lower layer to be ready for storage. Which, stores the data in the new form, and the metadata that are necessary for reconstruction the data later on.

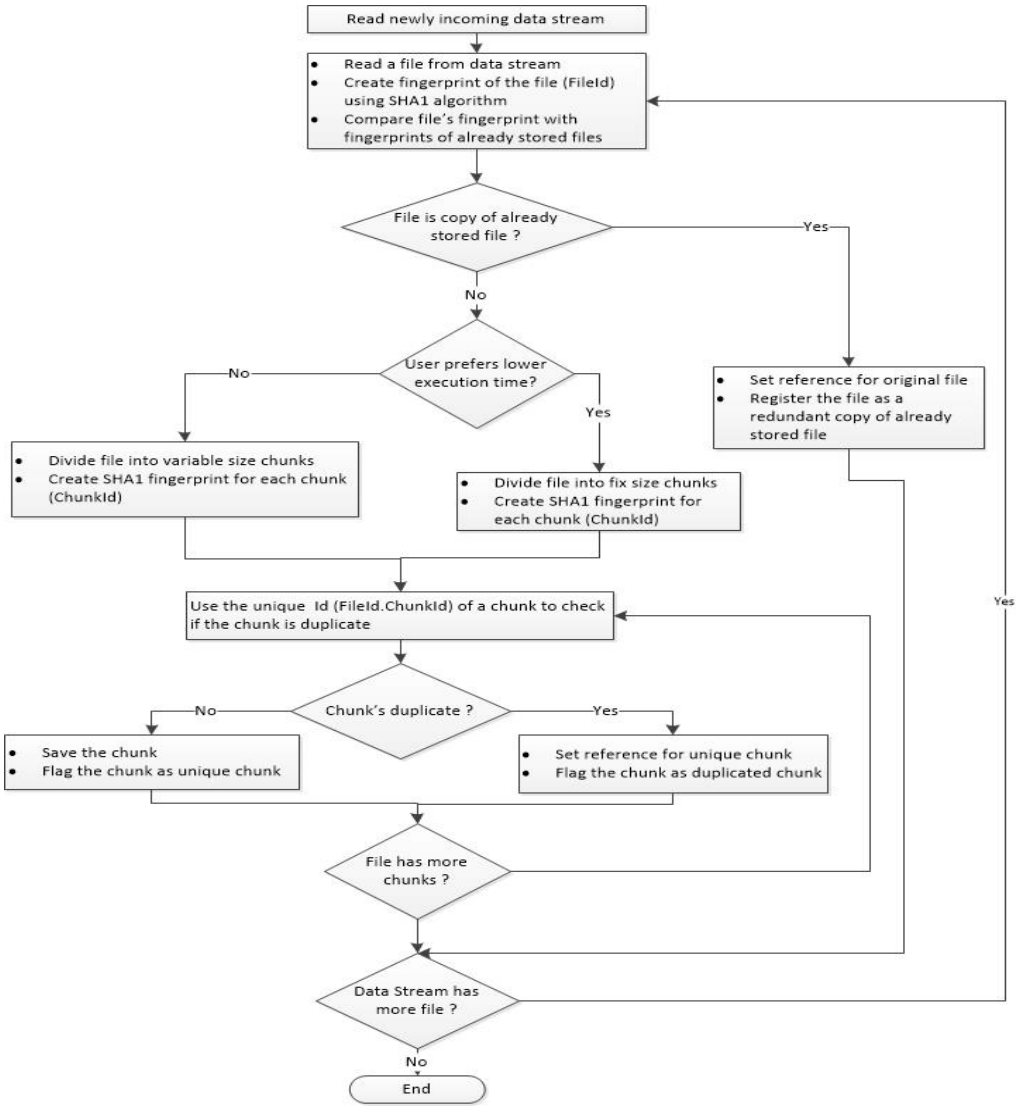

Fig. 5: Flow Chart At Hybrid Data Deduplication

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.3 Rabin-Karp variable size chunking [21][22]

The Rabin-Karp algorithm is string searching algorithm that is developed at 1980 by M. O. Rabin and R.M. Karp. The algorithm runs in time proportional to M+N, and the worst case is O(MN). There are algorithms have better worst case than the Rabin-Karp algorithm. The Rabin-Karp algorithm is used for solving the string matching problem. The string matching is the problem of finding the occurrences of certain substring in document(s), and it is a common problem in text editors. Also, the string matching problem can be employed in other fields like the searching using internet search engines. Where, the search engine is looking for certain substrings at relevant webpages in response to user query.

The Rabin-Karp searches for certain substring (pattern) occurrences at string, and calculates the hash value of this pattern. After that, the algorithm keeps moving on all substrings at string until it finds a substring has hash value similar to the hash value of the pattern. Indeed, the Rabin-Karp is fingerprint algorithm that uses the Rabin fingerprint for calculating hash value of each substring. Where, the hash value (integer) represents M-character substring. And, it uses the hash values of substrings in the searching process. Where, substrings that have similar hash value are equal to each other. Unless, there is a collision.

The Rabin-Karp algorithm is modified to become a solution for variable size chunking, the Rabin-Karp variable size chunking (RKVC) slides over the data stream for finding a set of bytes that has accepted hash value to become a variable size chunk. Where, if thirteen LSB of binary format of hash value is zero, the hash value is accepted and a new variable size chunk is found.

First, the RKVC defined an initialized window from the data stream that can be from byte at index 0 to byte at index 1023 (1024 bytes). After that RKVC calculates the hash value of the initialized window, the hash value of the initialized window is calculated depending on the following:

$$h(w) = h(t_0) + h(t_1) + h(t_2) + \cdots + h(t_s) \text{ .....eq. 1}$$

$$mult = mult(0) + mult(0) * q + mult(1) * q + \cdots + mult(s-1) * q \text{ ..... eq. 2}$$

$$h(t_0) = t_0 \text{ .... eq. 3}$$

$$h(t_{i+1}) = (h(t_i) * q) + t_{i+1} \text{ .... eq. 4}$$

$$mult(0) = 1 \text{ ..... eq. 5}$$

$$w : intialized\ window$$

$$t_0 : byte\ at\ index\ 0$$

$$s: size\ of\ intialied\ window$$

$$q: long\ prime\ number$$

If the hash value of the initialized window is accepted to form new variable size chunk. If the hash value of the initialized window is not accepted, the algorithm adds new byte for the window and recalculate the hash value depending on the following:

$$h(z) = [h(w) - (mult * h(t_{s+1})) * q] \text{ .....eq.6}$$

h(z): hash value of initialized window after adding new byte

The algorithm keeps adding bytes to the window one by one, and recalculating the hash value after adding each byte, until finding a set of bytes that has accepted hash value. At this point the window is considered a new variable size chunk. After that an empty window is created, and a new byte is added to this empty window. And, the hash value of this window is calculated using previous hash value, and the algorithm keeps adding bytes to the window one by one until find accepted variable size chunk. This algorithm keeps working in this way until reaching to the last byte at data stream of the file.

4. SIMULATION

A simulator was designed and developed for testing the proposed system that is developed using: Java programming language, IDE NetBeans 8.1 RC. The simulation was conducted on Dell laptop: core i7,2.50 GHz,8 GB RAM,1 TB HDD, and has Windows 7 professional 64-bit operating system.

Two test cases were prepared for acquiring results from the simulator, each test case has a set of directories. And, each directory has a set of files that either related or not related to each other, the data types of files at the directories can be: pictures (png, jpeg), documents, and audio and video files (3gp,MP4). These files are retrieved using advanced Google search, and the relations between the pictures are created using the Paint applications and between the audio and video files using the WonderShare video editor. Each test case is used

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

for conducting three scenarios. First scenario compares between FVCD and FFCD when chunk's size is 500 bytes, at second scenario the chunk's size of FFCD was increased to 1 KB, and at third scenario the chunk's size was increased to 1 MB. The results of each scenario are organized into a table that are represented using a curve chart to show the differences between the subsystems clearly.

A set of procedures were specified for obtaining the results from simulator using the test cases, and deriving the evaluations from these results. The data deduplication is carried out for the directories at a test case one by one, and after finishing the deduplication of a directory the data at the system and the execution time of deduplication were recorded. After that the testing process moves to the next directory at the test case. Finishing the deduplication of all directories at a test case, the difference between the original size of data and size of data after deduplication is used for calculating the effectivity of the deduplication. The following are the procedures for testing the system:

1. Choose files at a directory of test case for deduplication
2. Finishing the deduplication of the directory, the execution time of the deduplication process and the size of data at the deduplication system are recorded.
3. The efficiency ratio of the deduplication is derived using the following equations:
   $(A - B) = Z$ ……eq. 7
   $Efficiecny\ Ratio = \left(\frac{Z}{A}\right) * 100\ \%$ …. eq.8
   $A: size\ of\ files\ before\ deduplication$
   $B: size\ of\ files\ after\ deduplication$
   $Z: size\ of\ redundant\ data$
4. The previous points are repeated until finishing the testing of FVCD and FFCD using the test cases.
5. The results of testing the FVCD and FFCD were organized into a table of five columns. The first column represents the order of the directories at testing case, the 2nd and 3rd columns represent the results of testing FVCD. And, the 4th and 5th columns are the results of FFCD. Notice that record of headers is not considered a record.
6. The tables that contain the results of testing the simulator were represented into curve chart, that have two curves a curve represents the result of testing FVCD and another curve for the results of testing FFCD. The X axis at the curve chart represents the size of data at the system after deduplication, and the Y axis represents the time of execution for deduplication of files at a directory. Each curve at curve chart consists of a set of separated points, each point represents the results of deduplication files at a directory in a test case.

Generally, first scenario did not show promising results, where the results of testing FFCD was not practical. Since, at first scenario of the test case A the FFCD showed lower effectivity in reducing the size of data than FVCD. And, at the first scenario of test case B the FFCD showed much higher execution time from FVCD. At the second scenario after increasing the chunk's size of FFCD into 1 KB, the execution time of FFCD becomes better. But, nearly no improvement on the effectivity of reducing the size of data. Finally, at the last scenario after increasing the chunk's size of FFCD was promising for hybrid data deduplication system. Where, FFCD showed better execution time more than the FFVD. And, FVCD showed better effectivity in reducing the size of data more than FFCD.

4.1 Scenarios for testing using test case A

The test case A consists of eleven folders, and the execution time and size of data is recorded after the deduplication of each folder.

Table 1. Size And Accum. Size For Directories At Test Case A

#	Original size (MB)	Accum Original size (MB)
1	4.03	4.03
2	6.4	10.43
3	14.4	24.83
4	20.8	45.63
5	64.1	109.73
6	2.09	111.82
7	0.552	112.372
8	34.3	146.672
9	36.4	183.072
10	41	224.072
11	49	273.072

The table 1 represents the size of each directory, and accumulation (accum.) size of directories at the test case A.

Journal of Theoretical and Applied Information Technology
31st December 2017. Vol.95. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

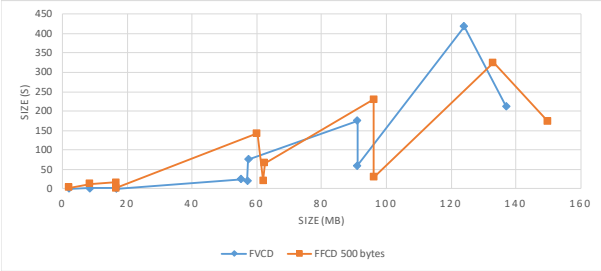

Fig. 6 Curve Chart Of Results At First Scenario In Test Case A

The curve chart at Figure 6 shows that FVCD has better efficiency ratio and slightly better execution time. Where, it is efficiency ratio is better with 5% and its execution time is better with nearly 3 s. The efficiency ratio of FVCD is 50% and average execution time for deduplication the directories at the test case is 90 s. While, the efficiency ratio of FFCD is 45% and average execution time nearly 93 s.

Table 2. Results Of FFCD With 500 Bytes And FFCD With 1 KB

#	FFCD 500 bytes	Time	FFCD 1 KB	Time
1	2.01	2.7	2.01	1.3
2	8.41	12.11	8.41	5.1
3	16.5	15.62	16.5	6.13
4	16.5	1.2	16.5	0.6
5	60.1	143	60.2	59.1
6	62	21.3	62.2	10
7	62.5	65.8	62.7	33.24
8	96.2	230.5	96.5	85.55
9	96.2	29.7	96.5	16.5
10	133	324.4	133	131.35
11	150	173	151	75.51
AVG		92.7		

The table 2 represents comparison between FFCD when chunk's size is 500.And, FFCD after increasing the chunk's size to 1 KB.

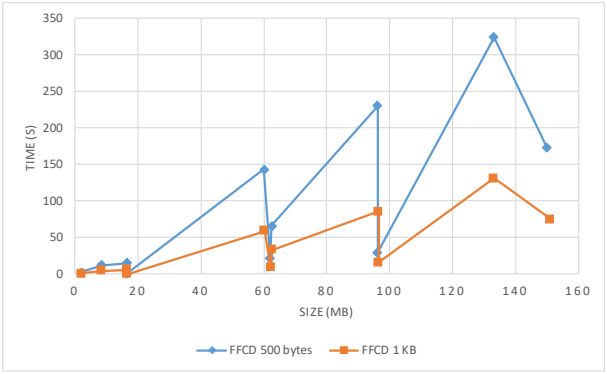

Fig. 7 Curve Chart Represents Comparison Between FFCD With 500 Bytes And FFCD With 1 KB

The Figure 7 represents the comparison between FFCD with chunk's size 500 bytes and FFCD after increasing the chunk size to 1 KB. The curve chart shows obviously that increasing the chunk's size reduce the execution time of the data deduplication.

Table 3. Comparison FFCD 1 MB And FVCD

#	Accum size (MB)	Time (s)	Accum size (MB)	Time (s)
1	2.01	0.53	2.01	0.18
2	8.41	1.36	8.41	0.37
3	16.5	1.93	16.5	0.81
4	16.5	0.81	16.5	0.64
5	55	24.17	80.7	2.2
6	57	20.02	82.8	0.74
7	57.4	75.52	83.3	3.92
8	91	175	117	6.47
9	91	60.21	177	3.66
10	124	418.68	157	11.29
11	137	213	187	6.85

The table 3 represents the results of comparison FVCD, and FFCD after increasing the chunk's size to 1 MB.

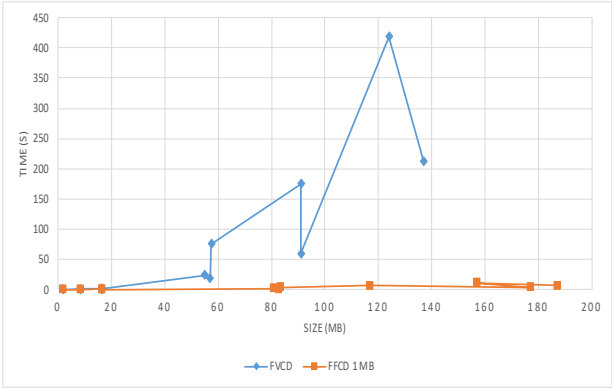

Fig. 8 Curve Chart Represents Comparison Between FVCD And FFCD With 1 MB

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The Figure 8 shows obviously that FFCD with 1 MB chunk's size has lower execution time than FVCD. While, FVCD shows better effectivity in reducing the size of data.

The efficiency of FFCD is less nearly with 18% and its execution time faster with 30 times. The efficiency ratio of FVCD is 50% and average execution time is 90, while it is 32% for FVCD and average execution time nearly 3.3 s.

4.2 Scenarios for testing using test case B

The testing of hybrid subsystems using test case A showed promising results for hybrid system. The hybrid FVCD showed better capability of reducing data size, and hybrid FFCD showed better execution time at third scenario when chunk's size becomes 1 MB. And, the effectivity of deduplication increased when there are more files related to each other and from same data type.

Table 4. Size And Accum. Size Of Directories At Test Case B

#	Original size (MB)	Accum size (MB)
1	15.5	15.5
2	26	41.5
3	52.7	94.2
4	38	132.2
5	200	332.2
6	138	470.2
7	20.2	490.4
8	11.6	502
9	41.1	543.1
10	210	753.1

The table 4 shows the size and accumulated size of directories at test case B, the test case B consists of 10 directories that its size higher that the size of the directories of test case A and their file more related to each other.

Table 5. Results Of First Scenario Test Case B

#	Accum size (MB)	Time (s)	Accum size (MB)	Time (s)
1	10.5	1.68	10.5	24.75
2	10.5	0.44	10.5	0.56
3	10.5	0.72	10.5	0.7
4	33.3	3.83	32.8	58.62
5	33.3	4	33	3.3
6	149	27.84	94.8	385.57
7	164	3.75	115	151.82
8	173	27.69	125	142.57
9	214	52.8	165	401.8
10	214	19.89	165	43.6

The table 5 shows the results of testing FVCD and FFCD, when the chunk's size is 500 bytes

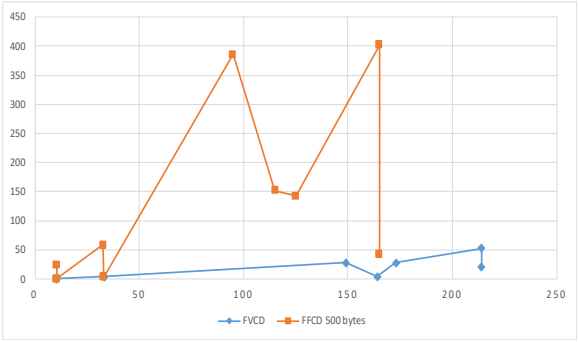

Fig. 9 Curve Chart Is Results Of First Scenario In Test Case B

The Figure 9 represents the results of comparison FFCD and FVCD, that it shows the FFCD has better capability in reducing size of data than FVCD. But, this is not a justification to use this type at hybrid system. Because, its execution time much higher than FVCD.

At this scenario of test case B, the FFCD shows better efficiency and lower average execution time, it is efficiency ratio was better with 6% and FVCD is faster nearly with 8 times. The efficiency ratio of FVCD is 72% and average execution time is 14.3 s. While, efficiency ratio of FFCD is 78%. And, the average execution time is 121 s.

Table 6. Results Of Third Scenario At Test Case B

#	Accum size (MB)	Time(s)	Accum size (MB)	Time(s)
1	10.5	1.68	11.7	0.64
2	10.5	0.44	11.7	0.51
3	10.5	0.72	11.7	0.93
4	33.3	3.83	37.3	1.47
5	33.3	4	37.3	3.74
6	149	27.84	160	4.62
7	164	3.75	181	1.02
8	173	27.69	192	2.61
9	214	52.8	233	5.12
10	214	19.89	233	7.37

The table 6 shows the results of the comparison FVCD and FFCD with chunk's size 1 MB. The FFCD has better execution time than FVCD. And, the FVCD has better capability in reducing the size of data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

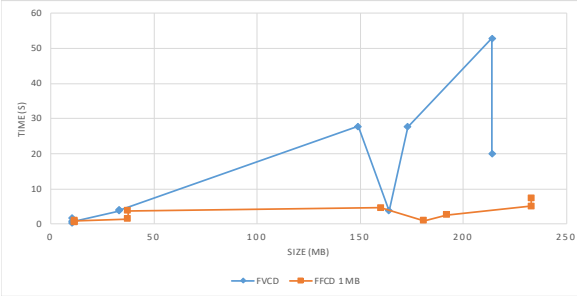

Fig. 10 Curve Chart Of Results Of Third Scenario At Case B

This Figure 10 shows the FVCD has better efficiency and higher execution time. And, the FFCD has better performance and lower efficiency. At this scenario FFCD has lower efficiency ratio and execution time. The efficiency ratio of FVCD is better with 3% and FFCD faster nearly with five times. The FVCD has efficiency ratio 72% and average execution time nearly 14.3 s. And, the FFCD has efficiency ratio 69% and average execution time nearly 2.8 seconds.

5. CONCLUSION

The hybrid data deduplication for cloud computing is designed for satisfying the diverse demands of applications and users at cloud of cloud service providers. The design of hybrid data deduplication got benefits from different type of data deduplication types to produce a powerful data deduplication system. Using the file level with chunk level deduplication can improve the execution time of data deduplication. When, the data has a lot of duplicated files. Moreover, the composition of file level and variable size chunk level deduplication is better for the users and applications that can tolerate higher execution time for having better effectivity in reducing the size of data. The FFCD shows better results than expected. Where, the FFCD showed effectivity nearly similar to FVCD when the chunk's size is 500 bytes. But, at test case B it showed much higher execution time than FVCD. The promising results appeared when the chunk's size of FFCD increased to 1 MB. The FVCD uses more complicated algorithm for chunk level deduplication for avoiding the shifting problem at FFCD. Which, means more processing overhead. Moreover, at FFCD smaller the chunk's size means more processing and higher execution time. So, the chunk's size of FFCD and parameters at FVCD must be tuned, until the hybrid deduplication system can have two different subsystems with different characteristics. Which can satisfy diverse requirements of users and applications. Therefore, the research objective has been achieved.

REFERENCES

[1] Vikraman,R.,Abirami,S.,*A Study on Various Data Deduplication Systems*, International Journal of Computer Applications ,International Journal of Computer Applications (IJCA),Volume 94 , Number 4,2014,Pages 35-40.

[2] Guo,F.,Efstathopoulos,P.,*Building a high-performance deduplication system*, USENIX annual technical conference, (Portland in U.S.A,2011), Publisher: USENIX Association Berkeley, Publication Date:2011-06-15,Pages:1-14

[3] NIST organization, The NIST Definition of Cloud Computing (September 2011), retrieved at 2015 from:http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[4] Kulkarni, G., Waghmare, R., Palwe, R., Waykule, V., Bankar, H., Koli,K., Cloud Storage Architecture,7th International Conference on Telecommunication Systems, Services, and Applications (TSSA),(Bali in Indonesia,2012),Pages : 76-81

[5] Liang,Q., W.,Yuan-Zhuo, ,Z.,Yong-Hui,Resource Virtualization Model Using Hybrid-graph Representation and Converging Algorithm for Cloud Computing,International Journal of Automation and Computing, December 2013, Volume 10, Issue 6, pp 597–606

[6] Amazon Web Services, the web site link is retrieved at 2015 from: https://aws.amazon.com/

[7] Singh,S.,Anand, S., *Implementing Storage as a Service in Cloud using Network Attached Storage*, International Journal of Computer Applications, Volume 108 ,Issue 13,December 2014,Pages 6-9

[8] Google Apps Official Web site, retrieved at 2015 from:https://www.google.com/work/apps/business/

[9] National Computer Broad, Guidelines on server consolidation and virtualization, retrieved at 2015 from:http://www.ncb.mu/English/Documents/Downloads/Reports%20and%20Guidelines/Guideline%20on%20Server%20Consolidation%20and%20Virtualisation.pdf

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[10] Lokeshwari,V.,Prabavathy,B.,Babu,C., *Optimized Cloud Storage with High Throughput deduplication Approach* , International Conference on Emerging Technology Trends (ICETT) , published by International Journal of Computer Applications (IJCA) (India,2011),Pages 32-37

[11] Yan, F. ,Tan,Y.,*A Method of Object-based Deduplication* , Journal of Networks ,Journal of Networks, Volume 6, Number 12, December 2011,Pages : 1705-1712

[12] Zhu,B.,K., Li,H., Patterson, *Avoiding the disk bottleneck in the data domain deduplication file system*, USENIX Conference on file and Storage Technologies(2008, San Jose in California)

[13]Kulkarni,P.,Douglis,F.,LaVoie,J.,Tracey,J.,*Redundancy Elimination Within Large Collections of Files*, Annual Technical Conference,(2004,Boston in Massachusetts)

[14] Shieh F.,Arani, M.,Shamsi, M., *Deduplication Approaches in Cloud Computing Environment: A Survey*, International Journal of Computer Applications ,Volume 120 ,Issue 13,2015,Pages :6-10

[15] Bhagwat, D.,Eshghi K., Long, D., Lillibridge,M., *Extreme Binning: Scalable, parallel deduplication for chunk-based file backup*, IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (London,2009)

[16] M., Devi,V.,Khanna,A.,Bhalaji, *Enhanced Dynamic Whole file Deduplication (DWFD) for Space Optimization in Private Cloud Storage Backup*, International Journal of Machine Learning and Computing, Volume 4, Issue 4, 2014, Pages 376-38

[17] Callaway, R., Devetsikiotis,M.,*Chunk and Object level deduplication for Web Optimization: A hybrid Approach*, IEEE International Conference on Communications (ICC), (Ottawa, June 2012), Pages 1393-1398

[18] Wang,J., A survey of web caching schemes for the Internet, ACM SIGCOMM Computer Communication Review Homepage archive, Volume 29 Issue 5, October 1999 ,Pages 36-46

[19] Maxcdn One, Proxy Caching, retrieved at 2015 from :https://www.maxcdn.com/one/visual-glossary/proxy-caching/

[20] Spring,N.,Wetherall,D., A protocol-independent technique for eliminating redundant network traffic, conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, (Stockholm in Sweden, 2000),Pages 87-95

[21] Cormen, H.,Leiserson,C.,Rivest, R.,Stein,C., Introduction to Algorithms, 3rd edition, IT Press,U.S, 2009, ISBN-13: 978-0262033848

[22] Sedgewick, R., Wayne, K., Algorithms 4th edition, Addison-Wesley Professional, U.S,2011, ISBN-13: 860-1400041420