INTERNATIONAL JOURNAL FOR EMERGING RESEARCH & DEVELOPMENT

(Volume1, Issue2)

Available online at: www.ijernd.com

Removing duplicated files on cloud storage using map reducing algorithm

Fairlin Jenefa. A
fairlinpenin@gmail.com
Ponjesly College of Engineering, Nagercoil, Tamil Nadu

Shajini. N
shajini88@gmail.com
Ponjesly College of Engineering, Nagercoil, Tamil Nadu

Geo Jenefer. G
geo.jenefer@gmail.com
Ponjesly College of Engineering, Nagercoil, Tamil Nadu

ABSTRACT

Cloud computing is a type of computing that relies on shared computing resources rather than having local servers or personal devices to handle applications. This technique avoids the duplication of data in the cloud and stores each file only once. Initially, each file are divided into more number of chunks and using this chunks a hash code is generated to avoid data duplication. Map Reduce Algorithm is used for the compression of chunks and using this chunks a hash code is generated to avoid data duplication. The Two Thresholds Two Divisors (TTTD) algorithm was proposed to improve the duplication of data by controlling the chunk-size variations. The first and the most important step is the granular division and subdivision of data. For proper granularity, an effective and efficient chunking algorithm is a must. The chunking process may increase the performance of the system. By using this method each file can be uploaded to the cloud only once, so it reduces the redundancy of files in the cloud.

Keywords: *Chunks, Map reduce, Hashing, Data duplication etc.*

1. INTRODUCTION

The big sensing data is large in volume and it comes from the different sensing systems. This data is very large and it requires processing before storing it on the cloud. Therefore, this purpose that data should be compressed first and then it should be stored in the cloud. There are different sources of the big sensing data such as camera, video, satellite meteorology, traffic monitoring, complex physics simulations. Hence big data processing is a big fundamental challenge for the modern society.

MapReduce can take advantage of locality of data, processing data on or near the storage assets to reduce data transmission. The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem and passes the input back to its master node.

Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

1.1. Related Works

[2] A Running Time Improvement for Two Thresholds Two Divisors Algorithm by Bing Chun Chang (2016) has dealt that the Chunking algorithms play an important role in data de-duplication systems. The Basic Sliding Window (BSW) algorithm is the first prototype of the content-based chunking algorithm which can handle most types of data. The Two Thresholds Two Divisors (TTTD) algorithm was proposed to improve the BSW algorithm in terms of controlling the variations of the chunk-size. In this project, the BSW algorithm and TTTD algorithm from different factors by a series of systematic experiments are compared. According to this analyses and the results of experiments, a running time improvement for the TTTD algorithm. This new solution reduces about 7 % of the total running time and also reduces about 50 % of the large-sized chunks while comparing with the original TTTD algorithm and make average chunk-size closer to the expected chunk-size. These significant results are the second important value of this project.

[10] An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment by Fang Liu et all (2011) has dealt that the market for cloud backup services in the personal computing environment is growing due to large volumes of valuable personal and corporate data being stored on desktops, laptops, and smartphones. Source deduplication has become a mainstay of cloud backup that saves network bandwidth and reduces storage space. However, there are two challenges facing deduplication for cloud backup service clients: (1) low deduplication efficiency due to a combination of the resource-intensive nature of deduplication and the limited system resources on the PC-based client site; and (2) low data transfer efficiency since post-

deduplication data transfers from source to backup servers are typically very small but must often cross a WAN. In this paper, an application-aware source deduplication scheme, to significantly reduce the computational overhead, increase the deduplication throughput and improve the data transfer efficiency.

[4] Block Level Data Deduplication for Faster Cloud Backup by Deshmukh et al (2015) has dealt that the easy access of information and data on the internet enable the users to download and store lots of data in their system. Most of the storage space is occupied by the duplicate data that have been downloaded or gathered from other resources and stored in the system thus increasing the requirement of the storage space. While taking the backup to the cloud same redundant data is uploaded to the cloud storage. With the increase in the amount of such redundant data proper utilization of the storage resources and bandwidth is not possible.

[5] Effective Big Data Storage Framework for Cloud Memory Management by Jaai Arankalle et al (2017) has dealt that the Big sensing data is extensively used in both industry and scientific research applications where the data is generated with huge volume. Cloud computing provides the best platform for big sensing data processing and storage. It moves around four important areas of analytics and Big Data, namely (i) data management and supporting architectures; (ii) model development and scoring;(iii) visualization and user interaction; and (iv) business models. However, the storage pressure on cloud storage system caused by the explosive growth of data is growing by the day, especially a vast amount of redundant data waste a lot of storage space. Data deduplication can effectively reduce the size of data by eliminating redundant data in storage systems.

[9] Enhance Data De-Duplication Performance with Multi-Thread Chunking Algorithm by Jia Zhao et al (2014) has dealt that a multi-threaded Frequency-Based Chunking method (FBC), which exploits the multicore architecture of the modern microprocessors. The legacy single threaded Frequency-Based Chunking method makes much improvement in reducing the total chunk size and the metadata overhead, compared to the popular Content-Defined Chunking algorithms (BSW, TTTD, etc.). Yet it also leaves much to be explored in terms of faster and more efficient file de-duplication and backup. This work is aimed at a significant increase in the chunking time performance, by applying concurrent frequency-based data chunking algorithm.

[3] File Deduplication with Cloud Storage File System by Chan-I Ku et al (2013) has dealt that the Hadoop Distributed File System (HDFS) is used to solve the storage problem of huge data, but does not provide a handling mechanism of duplicate files. In this study, the middle layer file system in the HBASE virtual architecture is used to do File Deduplicate in HDFS, with two architectures proposed according to different requires of the applied requirement reliability, therein one is RFD - HDFS (Reliable File Deduplicated HDFS) which is not permitted to have any errors and the other is FD-HDFS (File Deduplicated HDFS) which can tolerate very few errors. In addition to the advantage of the space complexity, the marginal benefits from it are explored. Assuming a popular video is uploaded to HDFS by one million users, through the Hadoop replication, they are divided into three million files to store, that is a practice wasting disk space very much and only by the cloud to remove repeats for effectively loading.

[1] Learning Compression Approach Based On Scalable Data Chunk Similarity for Big Sensing Data Processing On Cloud by Akanksha More, Mr. S. B. Javheri (2017) has dealt that the Big sensing data is the data with high volume and high velocity. The big sensing data is generated in the many systems at different applications such as research and big organizations. Cloud is the platform where the data can be stored, serviced and computed at the cloud. Since the data is big volume there is need of data processing such as data compression. The data compression techniques requires large scalability and efficiency for the processing of the large volume of data in the high rate. In this paper, the technique for the compression of the data. This technique is based on the on cloud requirement of data compression and similarity calculation between data chunks. In this method firstly the big data is divided into the different chunks and then these chunks are compressed according to the chunks. Here another algorithm known as Jaccard is used to find the similarity between the data. Then this algorithm is compared with the existing systems similarity algorithm.

[6] Secure Static Data De-Duplication by Rohit Pawar et al (2010) has dealt that the Data de-duplication is a technique used to improve storage efficiency. In static data de-duplication system, Hashing is carried out at client side. Firstly hashing is done at file level. The de-duplicator identifies duplication by comparing existing hash values in metadata server. If match is found, then logical pointers are created for storing redundant data. If match doesn't exist, then same process is carried out at chunk level. Duplicated data chunks are identified and only one replica of the data is stored in storage. Logical pointers are created for other copies, instead of storing redundant data. If it is a new hash value, it will be recorded in metadata server and the file or corresponding chunk will be stored in file server and its logical path in terms of logical pointers is also stored in metadata server. Basically static deduplication is implemented with three components: interface, de-duplicator and storage. Interface carries out hashing of uploaded file and interfaces client with deduplication. After receiving hash value, deduplication carries out its function as mentioned above. The last component storage consists of file server and metadata server. Thus, de-duplication can reduce both storage space and network bandwidth.

[7] Study of Chunking Algorithm in Data Deduplication by A. Venish and K. Siva Sankar (2013) has dealt that the Data deduplication is an emerging technology that introduces reduction of storage utilization and an efficient way of handling data replication in the backup environment. In cloud data storage, the deduplication technology plays a major role in the virtual machine framework, data sharing network, and structured and unstructured data handling by social media and, also, disaster recovery. In the deduplication technology, data are broken down into multiple pieces called “chunks” and every chunk is identified with a unique hash identifier. These identifiers are used to compare the chunks with previously stored chunks and verified for duplication. Since the chunking algorithm is the first step involved in getting efficient data deduplication ratio and throughput, it is very important in the deduplication scenario. In this paper, different chunking models and algorithms with a comparison of their performances.

[8] Using Similarity in Content and Access Patterns to improve Space Efficiency and Performance in Storage Systems by Xing Lin (2015) has dealt that the Compression works by identifying repeated strings and replacing them with more compact encodings while deduplication partitions data into fixed-size or variable-size chunks and removes duplicate blocks. While this approach have seen

great improvements in space efficiency from these two approaches, there are still some limitations. First, traditional compressors are limited in their ability to detect redundancy across a large range since they search for redundant data in a fine-grain level (string level). For deduplication, metadata embedded in an input file changes more frequently, and this introduces more unnecessary unique chunks, leading to poor deduplication. Cloud storage systems suffer from unpredictable and inefficient performance because of interference among different types of workloads.

1.2. Problem Identification

A fundamental problem that confronts peer-to-peer applications in the location of the node that stores a desired data item. In existing technique each file is uploaded to all the four clouds in peer-to-peer manner by means of chunking. While chunking each individual file is divided into four chunks. All the four chunks are equal in size. And each chunk is uploaded to four different cloud in four different locations. This process may be done one after the other. So if a new file is uploaded it needs to check with all the chunks to find the duplication of the file. If the chunks of existing files does not matches with the chunks of newly uploaded file means the file is uploaded in the cloud. If the any of the chunk matches with the existing file means it shows that the file may be duplicated. And the file is not uploaded in the cloud. But in this existing process matching is happened in all the four clouds. This may results in wastage of processing time. And the file chunks are stored in various clouds may also results a problem. Because while downloading a file the server needs to wait until the previous process would complete.

Disadvantages

- It does not eliminate the very large or very small chunk to control the variation in chunk size.
- The system only wastes network resources to transmit these very large or very small chunks when one-byte modifications happen.

1.3. Proposed Modeling

MapReduce is a framework for processing parallelizable and scalable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Computational processing can occur on data stored either in a filesystem (unstructured) or in a database (structured).

MapReduce can take advantage of locality of data, processing data on or near the storage assets to reduce data transmission. The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem and passes the input back to its master node. With the above analysis, the following detailed algorithms are used for data chunks generation. The algorithms are as follows,

- Content Defined Chunking
- Static Chunking
- Whole File Compression

The Content Defined Chunking is a variable size chunking method and this chunking method is suitable for text format and word files. The Static Chunking is a fixed size chunking method and this type of chunking is suitable for Executable Files. The Whole File Compression method is a compression method and it is suitable for image files and PDF files.

1.3.1 The architecture of Proposed Modeling

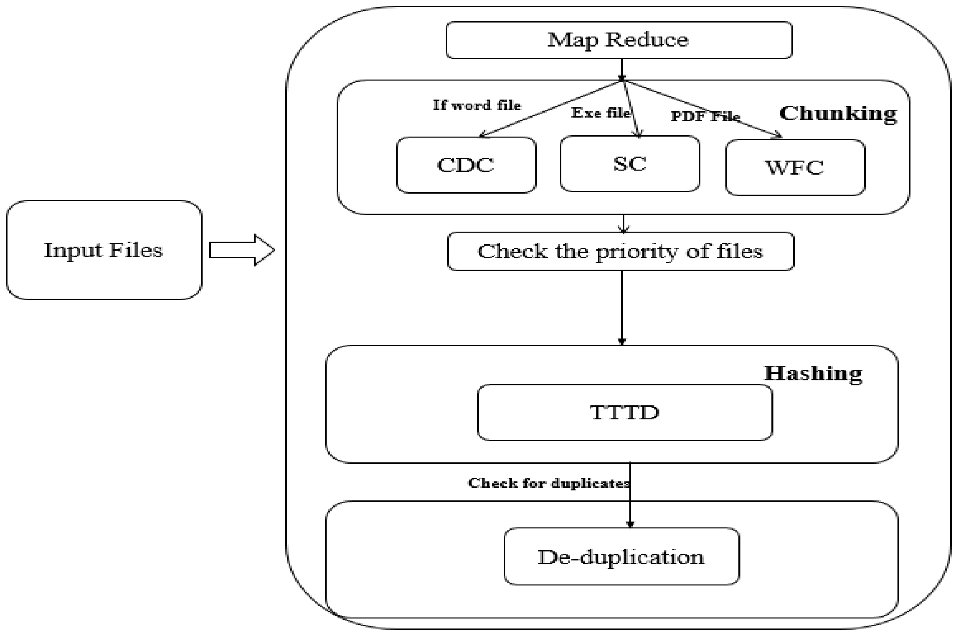

Figure 1.3.1 Data Chunking Architecture

The Figure 1.3.1 explains the Data Chunking architecture of the proposed system. The working of the system is as follows,

When a file is uploaded to the cloud, initially the Map-Reduce function will shuffle the input file and analyze the file for chunking. After that, the file is chunked based upon the behavior of the file. The chunking is performed in various ways as Content Defined Chunking, Static Chunking and Whole File Compression methods.

The Content Defined Chunking is a variable size chunking method and this chunking method is suitable for text format and word files. The Static Chunking is a fixed size chunking method and this type of chunking is suitable for executable files. The Whole File Compression method is a compression method and it is suitable for image files and PDF files.

Initially, a standard data chunk is created. This chunk is bigger in size. After chunking process, the priority of the file is checked. Then the hashing is performed.

The hashing is performed by means of Two Threshold Two Divisor Algorithm (TTTD). The TTTD Algorithm splits the standard data chunk into two parts as the main divisor and second divisor. Then for those two parts, the ASCII values are converted into the corresponding binary numbers and the values are stored in the database. When a new file is uploaded in the cloud the TTTD algorithm continues the same process and checks the hash code value in the database if the same value is found in the database the file may be duplicated. Otherwise, the file is uploaded to the cloud.

For downloading a file from the server by a client, go to file download and selects the file which the user needs to download. Then click the ok button. Thus the file was downloaded by the client.

1.4. Implementation

Implementation is the stage of the project when the theoretical design id turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constrains on implementation, designing methods to achieve changeover.

1.4.1 MapReduce

MapReduce is a framework for processing parallelizable and scalable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware).

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The Map-Reduce algorithm description is shown below in the form of a diagram below,

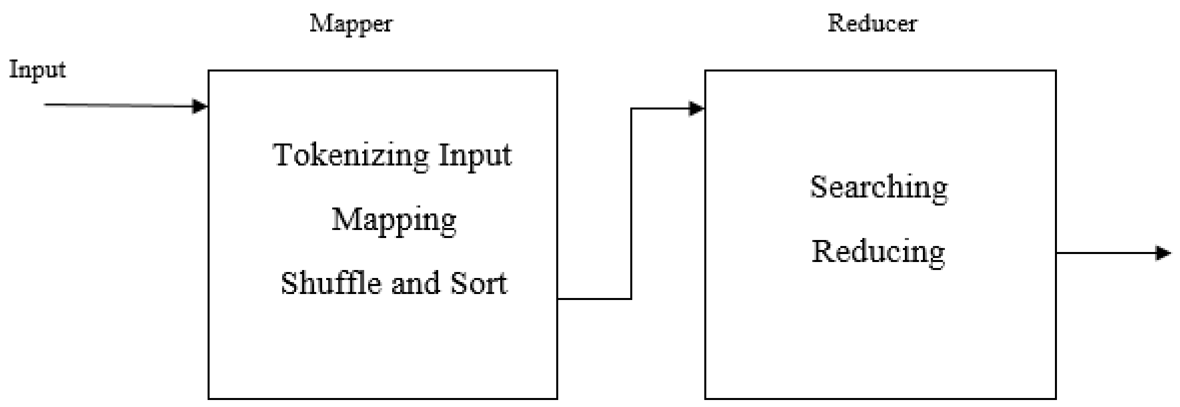

Figure 4.2 Map Reducer Algorithm

1.4.2 Data Chunk Generation

In data chunk generation, the recursive process will increase the number of data chunks step by step. The formation process of data chunks will be offered by training initial data stream. At first, a standard data chunk is created. The standard data chunk is larger than other chunks. The generated data chunks after initial selection have a different size to each other.

The data chunk generation is performed by means of the following algorithms,

- Content Defined Chunking
- Static Chunking

- Whole File Compression

Content Defined Chunking (CDC)

The Content Defined Chunking is a variable size chunking method. This chunking method will split the file into more chunks such that each chunk is variable sized. This chunking method is suitable for text format and word files.

Static Chunking (SC)

The Static Chunking is a fixed size chunking method. This chunking may split the file into the same size. So in this type of chunking, all the chunks are of similar size. This type of chunking is suitable for Executable Files.

Whole File Compression (WFC)

The Whole File Compression method is a compression method and it is suitable for image files and PDF files. The WFC will compress the files and split the files into various chunks.

1.4.3 Hashing

Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

Two Threshold Two Divisor (TTTD)

The Two Threshold Two Divisor (TTTD) method ascertains the chunks smaller than a particular size. And it considers only the chunk with the bigger size. To achieve this, the method ignores the chunk boundaries after a minimum size is reached. So the larger chunk is further divided. TTTD applies two techniques where two divisors (D, the regular divisor, and D0, the backup divisor) are used. By applying these divisors, TTTD guarantees a minimum and maximum chunk size. A minimum and a maximum size limit are used for splitting a file into chunks for searching for duplicates.

TTTD Algorithm

```
int currP = 0, lastP = 0, backupBreak = 0 ;
for ( ; ! endOfFile( input ) ; currP++ )
{
unsigned char c = getNextByte( input ) ;
unsigned int hash = updateHash( c ) ;

if ( currP – lastP < minT )
{
continue ;
}
if ( currP – lastP > switchP )
{
switchDivisor( ) ;
}
if (( hash % secondD ) = = secondD – 1 )
{
backupBreak = currP ;
}
if (( hash % mainD ) = = mainD – 1 )
{
addBreakpoint( currP ) ;
backupBreak = 0 ;
lastP = currP ;
resetDivisor( ) ;
continue ;
}
if ( currP – lastP < maxT )
{
continue ;
}
if ( backupBreak != 0 )
{
addBreakpoint( backupBreak ) ;
lastP = backupBreak ;
backupBreak = 0 ;
resetDivisor( ) ;
}
else
{
addBreakpoint( currP ) ;
lastP = currP ;
backupBreak = 0
resetDivisor( ) ;
}
}
```

1.5. Result and Discussion

1.5.1 Performance Analysis

Performance Analysis is a specialist discipline involving systematic observations to enhance performance and improve decision making, primarily delivered through the provision of objective statistical (Data Analysis) and visual feedback. The overall upload time of our proposed system is demonstrated.

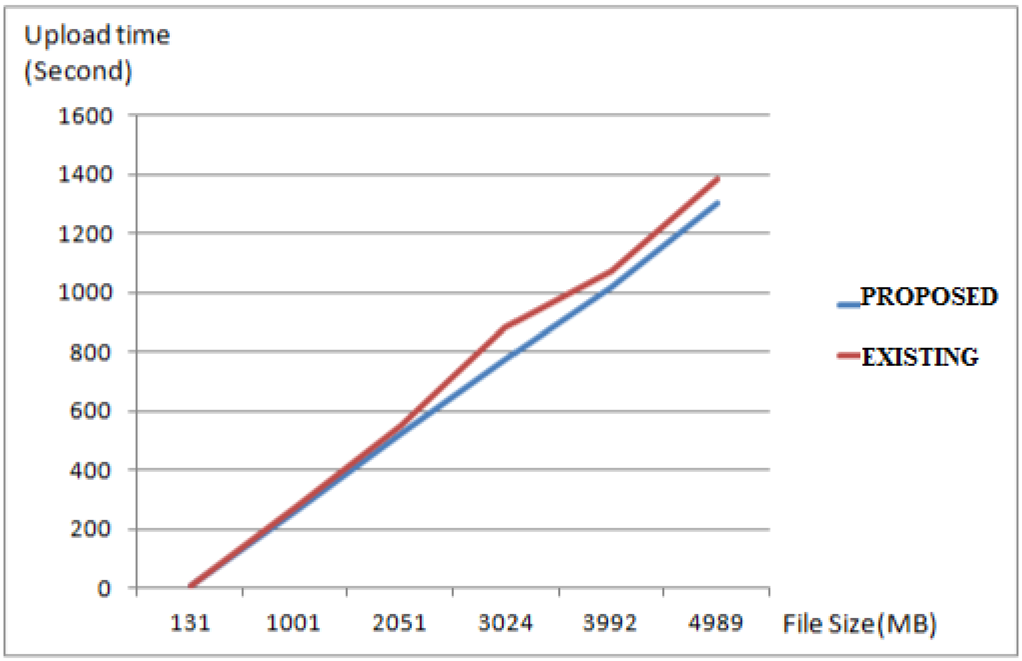

Figure 5.1 Uploading Time

In Figure 5.1, the X-axis shows the uploading time of the files. The Y-axis shows the file size. In this graph, the uploading time of the existing and proposed system is compared. From this graph, the user can notice the uploading time of the proposed system is reduced.

2. CONCLUSION

The first and the most important step is the granular division and subdivision of data. For proper granularity, an effective and efficient chunking algorithm is a must. So we have used various chunking algorithms for performing the task. The chunking process may increase the performance of the system because each file is divided into various smaller chunks so that larger files are easy to upload. By using this method each file can be uploaded to the cloud only once, so it reduces the redundancy of files in the cloud and increases the storage space in the cloud.

3. REFERENCES

[1] Akanksha More, Mr. S. B. Javheri**, “**Learning Compression Approach “Based On Scalable Data Chunk Similarity For Big Sensing Data Processing On Cloud”, International Journal of Engineering Technology Science and Research”, Volume 4, July 2017.
[2] BingChun Chang, “A Running Time Improvement for Two Thresholds Two Divisors Algorithm”, ACM 978-1-4503-0064, 2010.
[3] Chan-I Ku, Guo-Heng Luo, Che-Pin Chang & Shyan-Ming Yuan, “File Deduplication with Cloud Storage File System”, IEEE 16th International Conference on Computational Science and Engineering, 2013.
[4] Deshmukh, Vikram V. Badge, Block Level Data Deduplication for Faster Cloud Backup”, International Journal of Scientific & Engineering Research, Volume 6, Issue 7, July-2015.
[5] Jaai Arankalle, Shubhangi Bagade, Saee Joshi, Rutuja Limaye, “Effective Big Data Storage Framework for Cloud Memory Management” International Journal of Innovative Research in Computer and Communication Engineering, Vol. 5, Special Issue 6, July 2017.
[6] Rohit Pawar, Payal Zanwar, Shruti Bora, Shweta Kulkarni, “Secure Static Data de-Duplication”, Journal of Computer and System Sciences, 2010.

[7] Venish and K. Siva Sankar, “Study of Chunking Algorithm in Data Deduplication”, International Conference on Soft Computing Systems, Advances in Intelligent Systems and Computing 398, DOI 10.1007/978-81-322-2674-1_2, 2016.
[8] Xing Lin, “Using Similarity in Content and Access Patterns to improve Space Efficiency and Performance in Storage Systems”, Journal of Computer and System Sciences, 2015.
[9] Xinran Jiang, Jia Zhao, Jie Zheng, “Enhance Data De-Duplication Performance with Multi-Thread Chunking Algorithm”, COEN 283, December 9, 2014.
[10] Yinjin F, Hong Jian, Nong Xiao, Lei Tian, Fang Liu, “An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment”, IEEE International Conference on Cluster Computing, 2011.