

Xu Xiaowei

Data Deduplication in Windows Server 2012 and Ubuntu Linux

Bachelor's Thesis
Information Technology


May 2013



MIKKELIN AMMATTIKORKEAKOULU

Mikkeli University of Applied Sciences

DESCRIPTION

 <p>MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences</p>		Date of the bachelor's thesis 2013/5/8
Author(s) Xu Xiaowei	Degree programme and option Information Technology	
Name of the bachelor's thesis Data Deduplication in Windows Server 2012 and Ubuntu Linux		
Abstract <p>The technology of data deduplication might still sound unfamiliar to public, but actually this technology has already been a necessary and important technique among the datacentres over the whole world. This paper systematically introduces the data deduplication technology including what is data deduplication, where it is from, and the mainstream classifications. Also the appliance of data deduplication in Windows Server 2012 and Ubuntu will be simply covered so that it is easier to understand the principle and effect better.</p>		
Subject headings, (keywords) Data deduplication, Windows Server 2012, Ubuntu, ZFS		
Pages 60	Language English	URN
Remarks, notes on appendices		
Tutor Juutilainen Matti	Employer of the bachelor's thesis Department of electrical engineering and information technology	

CONTENTS

1 INTRODUCTION	1
2 DATA DEDUPLICATION	1
2.1 What is data deduplication.....	2
2.2 History of the deduplication	2
2.3 Comparison to compression.....	3
3 CLASSIFICATION	5
3.1 Deduplication based on processing position.....	6
3.1.1 Source deduplication.....	6
3.1.2 Target deduplication (Deduplication Appliance-based)	7
3.2 Deduplication based on time of processing	8
3.2.1 Offline deduplication(post-process deduplication or asynchronous) 9	
3.2.2 In-line deduplication(synchronous)	10
3.3 Deduplication based on storage position	11
3.3.1 Secondary deduplication.....	11
3.3.2 Primary deduplication.....	11
3.4 Deduplication based on algorithm	13
3.4.1 Delta-encoding Deduplication	13
4 HASH-BASED DEDUPLICATION	15
4.1 Principle	15
4.2 General process of hash-based deduplication	16
4.3 Classification.....	17
4.3.1 Whole File Hashing (or File-level Deduplication)	18

4.3.2	Block-level Deduplication (or Sub File Hashing)	20
4.3.3	Byte-level Deduplication	23
4.4	Problem of hash-based deduplication and related strategy	24
4.4.1	Hash collision.....	24
4.4.2	Granularity	25
4.4.3	I/O optimization.....	25
5	WINDOWS SERVER 2012	27
5.1	Introduction.....	27
5.1.1	Features of deduplication in Windows Server 2012	28
5.1.2	How does it work	28
5.1.3	The components of deduplication	29
5.2	Installing	30
5.3	Data deduplication in windows server 2012.....	33
5.3.1	Add deduplication feature.....	34
5.3.2	Enable deduplication	38
5.3.3	Process deduplication.....	42
6	LINUX.....	45
6.1	Introduction of Ubuntu	46
6.2	Data Deduplication by ZFS	46
6.2.1	Introduction of ZFS.....	46
6.2.2	Deduplication in ZFS	47
6.3	Installing.....	47
6.4	Data Deduplication with ZFS	49
7	CONCLUSION.....	52

1 INTRODUCTION

Nowadays, content delivery networks, online backup storage, news broadcasting, blog sharing and social networks as an ascendant part of Internet services are data centric. Hundreds of millions of users of these services generate petabytes of new data every day. For instance, as of April 2011, Dropbox, an online file-sharing and backup services, has more than 25 million 2GB dropboxes (total of 50 petabytes).^[31] A large portion of internet service data is redundant for the following two reasons. Firstly, because of the significant decline in the storage cost per GB, people tend to store multiple copies of a single file for data safety or convenience. Secondly, while incremental (or differential) data backups or disk image files for virtual desktops tend not to have duplicated whole-file copies, there is still a large fraction of duplicated data portion from the modifications and revisions of the files.^[32]

2 DATA DEDUPLICATION

As is known to us all, this is an era of information. Data as a main carrier of information, its importance is self-evident under the premise of development in the information age of globalization nowadays. Due to the rapid growth in the amount of enterprise data and data transfer rate requirements continue to increase, the mass storage of data center space and high-bandwidth network transmission is facing serious challenges in the field of network storage. Therefore it has been quite significant that how can we storage efficiently for them can be hold for a longer time with the data increasing in a speed that we can not imagine in the past. According to the certain research, there are 80%~90% of data is redundant in backup and archival system^[1] and 80% of data is redundant in primary storage system based on virtual machine.^[2] In order to solve this tricky problem, saving more space and cost, data deduplication has naturally become a focus that more people are paying attention to.

2.1 What is deduplication

Data deduplication is a technology that refers to the elimination of redundant data through saving the unique copy of the data. ^[3] Deduplication reduces the required storage capacity since only one copy of data is stored and if duplicates are detected, it will not be stored again. Instead of storing, a data pointer is created and used to point it to the only one copy in the disk. As illustrated in FIGURE 1, we have 15 chunks in original data, but only 4 left after removing the duplicates

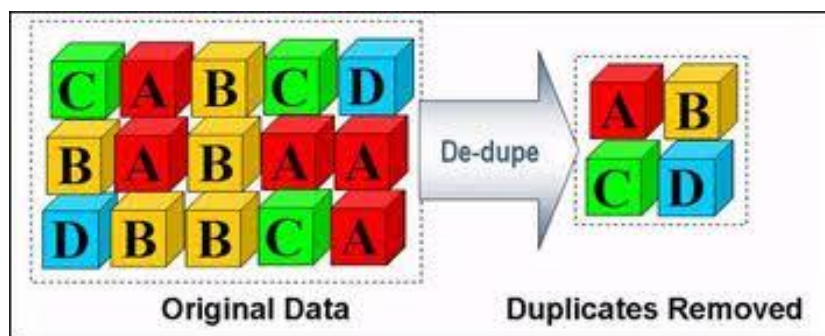


FIGURE 1. Removing redundant information with data deduplication

With this new technology, the amount of data that has to be stored has been reduced greatly, which means that we could save money which was supposed to buy store devices and more data could be stored in one disk and tape. Less data leads to less bandwidth and that could give us a higher speed in remote backup, replication and disaster recovery processes. ^[3]

2.2 History

The technology of data deduplication is brought forward since 2007, when this technology is getting to be known by more and more people and accepted worldwide. But actually we could see its initial form as early as 1997 in November, when Microsoft released Exchange Server 5.5, which maintain single-instance-storage (SIS) of messages which can be seen the original form of the data deduplication, namely, a kind of whole-file-chunking as we all know nowadays. ^[4] It is also in Microsoft

Exchange 2000 and Microsoft Exchange Server 2003 as an important part. ^[5] The basic idea is that if a message is sent to one recipient, and if the message is copied to 20 other recipients who reside in the same mailbox store, Exchange Server maintains only one copy of the message in its database. Exchange Server then creates pointers. These pointers link both the original recipient and the 20 additional recipients to the original message. If the original recipient and the 20 additional recipients are moved to another mailbox store, only one copy of the message is maintained in the new mailbox store. The new mailbox store can be on another server in the same site or in an administrative group. If the server is in another site, single-instance storage is retained only if you use the Move Mailbox Wizard in Microsoft Exchange Server 2003 Service Pack 1 (SP1) or later versions. SIS has already been a standard configuration since Windows Storage Server 2003 R2. ^[4]

But actually, people had already been thinking about how to store data effectively since the first data center came up. Most believe that data center was born in the late 90s in the dot com boom, but as a matter of fact, data centers are as old as early computer era previously known as private computer or dedicated computer room which can date back to 1960s. ^[6] But with the rapid development of information technology, the data generated in the process of information has shown an explosive growth. This increasingly grim situation has forced us to create more effective ways to store more data with the limited space which naturally bring our topic out.

2.3 Comparison to compression

Compression or data compression, bit-rate reduction, involves encoding information using fewer bits than the original representation. Traditional data compression is kind of the extension or applying of the Morse's code with the same idea that Morse has that longer sequences of bits can be represented by shorter ones. The idea of this technology is that it tries to decrease the size of some certain file by eliminating redundant data in the file without changing the content of the file. For instance, if we

give compression to three apples, then the result will be that we get three smaller apples. They are still apples although we may cut some redundant parts off but becoming smaller. So with compression we can save a lot of space by compressing them.

Data deduplication also provides us more space just like compression but in another way which is related to compression in some levels conceptually but generally total different. As we mentioned above, the idea of compression is longer sequences of bits can be represented by shorter ones, if we change the bits to file, we can get the similar idea of deduplication. So we can easily find that compared to traditional data compression technology Deduplication technology can not only eliminate the redundancy of the data within the file, also eliminates data redundancy between shared data sets within the file. We still take apples for example, if we give data deduplication to three apples, the result will be that we only get one apple provided they own the same properties. However it is not fully correct to understand that data deduplication is just to eliminate redundant copies of files, in fact, there is other forms of this technology. We can cut files into chunks or even bytes first, and then eliminate redundancy. These methods will be introduced with more details later.

Before we compare which technology is better or not, the ideas of each technology must be well distinguished. So in compression, we are trying to make the size of a certain file smaller by replacing some characters to some other smaller ones without changing the content of file. As to deduplication, we are trying to save space by storing the only one copy and eliminate the ones that have been seen and stored before, and the copies could refer to files, chunks, or bytes depending on what method you are using. Compression technology is pretty easy to implement but limited to relatively short data sets. Data deduplication is more effective in practice, but it will become difficult to implement anyway since there is definitely more data to process. We have to admit that deduplication has proven its achievements in different kinds of enterprise storage and backup system, but it does not mean it is better than

compression then, the best answer for this question could be:it depends. It depends on what kind of data do you have, if you have so many duplicate data, data deduplication would be better for you, compression would be a nice answer if you have very good compressibility in your data.

Since compression and data deduplication good at saving space in different way, it is also a good idea to combine them together just as shown in Figure 2.

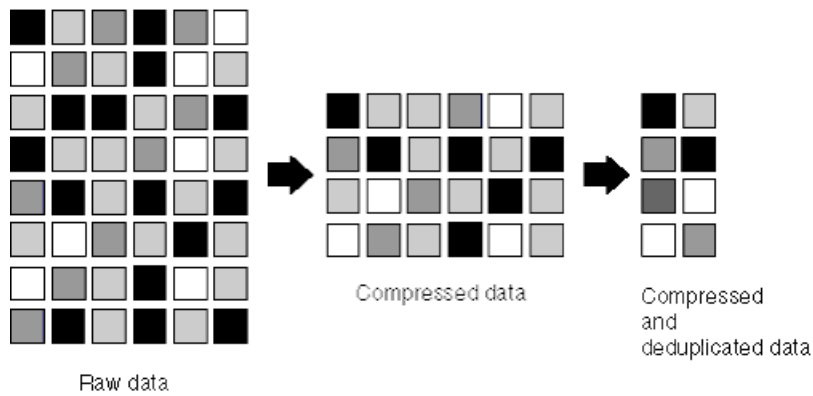


FIGURE 2. Combination of compression and deduplication

3 CLASSIFICATION

We could classify deduplication in many ways, there are generally four classifications : from point of application ,from time of application, granularity and algorithm, there basic relationship be seen in FIGURE 3.

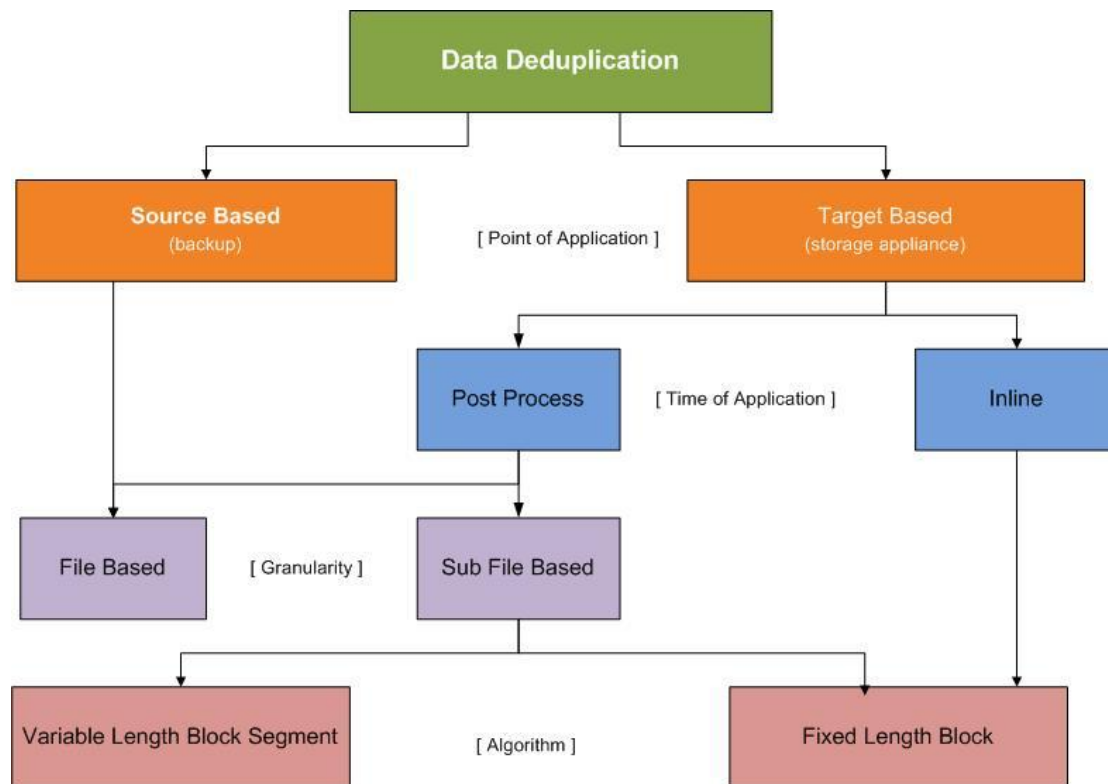


FIGURE 3 General classification

3.1 Deduplication based on processing position

Technically, there are two ways of deduplication if we classify it from the processing position, source-deduplication and target-deduplication which will be discussed in detail next.

3.1.1 Source deduplication(Client-based)

Source is the place where your data is created. If we remove all the duplicates before we transmit it to the backup, then it is called source-deduplication. With this method we could largely reduce network traffic and bandwidth as well as substantially reduce backup window since we are sending data with no redundant during each backup session. On the other hand, if we are dealing with large amounts of data, it will increase the time of process for the data will make the servers on source side much heavier. Just as shown in FIGURE 4, the servers on the left side are so-called source

where a client-located deduplication engine is typically used to check for duplicates. Through the deduplication process, only unique copy will be transmitted to the disk.

[7]

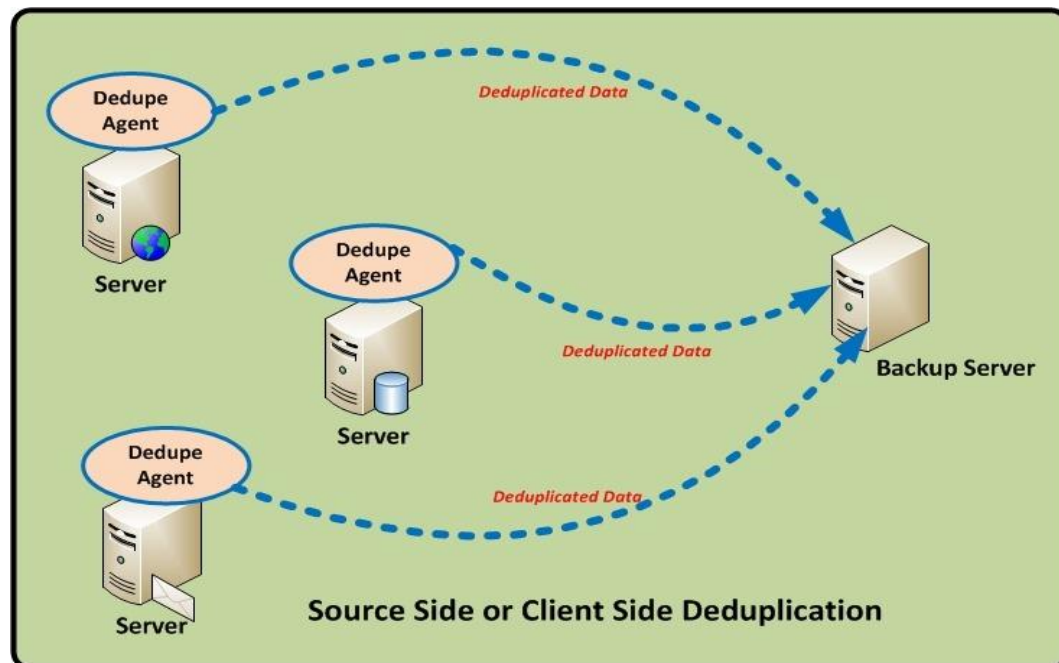


FIGURE 4. source side or client side deduplication

Usually this method is used when remote offices must back up data to the primary data center which is the situation that deduplication need to be done in client server.

[7]

LBFS(Muthitacharoen, Chen, & Mazieres 2001) has used this approach to create bandwidth-efficient file system, also cloud-based file systems wrappers for exemple,S3QL(S3QL 2012) and OpenDedup(OpenDedup 2012) has performed this method to reduce customer storage burden. [8]

3.1.2 Target deduplication (Deduplication Appliance-based)

If the deduplication is performed at the backup target after transmission, we call it target-deduplication. In this way, all backups can be aggregated into a single storage

system. Like we just mentioned above, it will take too much time when the deduplication is processed in source-deduplication, but it is a significantly excellent idea to process the data in target-deduplication since it can provide much faster performance when dealing with large amounts of data. However requiring hardware at remote sites is considered a drawback in comparison to source deduplication. Another drawback is that there is no bandwidth savings in this method because all the data need to be transmitted via network. As shown in FIGURE 5, all the backup data without deduplication is transmitted to the target device which owns the deduplication appliance to perform the deduplication process.

Now, Venti(Quinlan & Dorward 2002) and ZFS(Bonwick 2009) are typical systems that are performing deduplication in the storage appliance. ^[8]

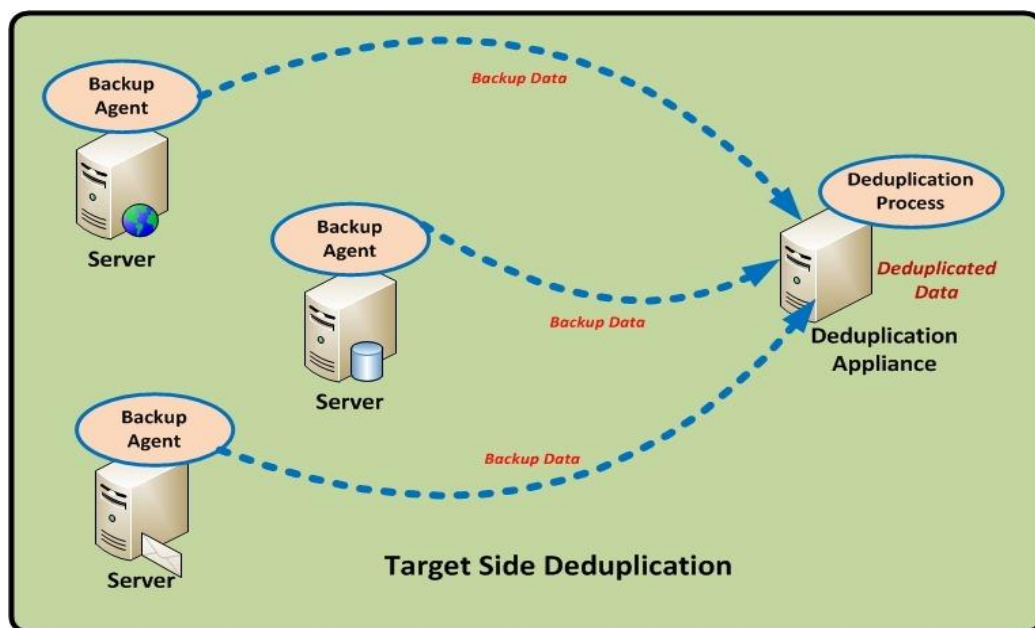


FIGURE 5 target side deduplication

3.2 Deduplication based on time of processing

There is another way to classify data deduplication. It can be divided into two ways depending on the time of processing, offline deduplication and in-line deduplication.

3.2.1 Offline deduplication (post-process deduplication or asynchronous)

Offline deduplication, also called post process deduplication, is a method that the new data we received will be first stored on a storage device, and then the second step will be that duplicate-detection process analyzes the stored data for duplicates which means the process will detect if the new data has been seen before, if it is then the process will delete the redundant blocks of data and replaces it with a pointer, but if the data has not been seen before, then no change will happen, the data will still be in the storage device, as shown in FIGURE 6. Since the data that need to be deduplicated is processed at a later time, it does not affect write latency with small CPU occupancy which is therefore good for latency-critical applications, where the ingestion speed of data is in high demanding. But requiring additional storage to store new data that has not yet been analyzed is one of the disadvantages of asynchronous deduplication. Another one is that we need to keep two protocols for reading data, one is for unprocessed another one is for deduplicated data. ^[8]

NetApp ASIS(Alvarez 2011), EMC Celerra (EMC 2010) and IBM StorageTank (IBM Corporation 2002) are using this approach in their storage systems.

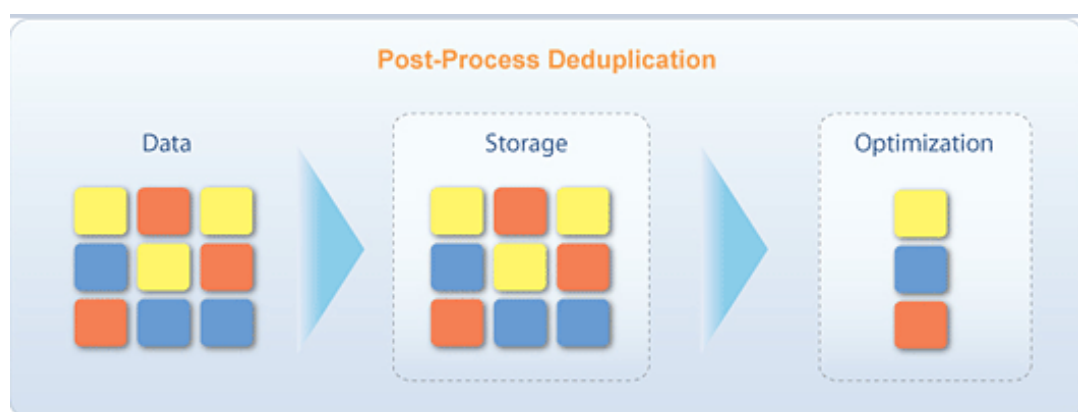


FIGURE 6 post-process deduplication

3.2.2 In-line deduplication(Synchronous)

While an in-line deduplication method, on the other hand, the process is occurred when the data is still in RAM before writing through I/O. That is saying that the job that detecting whether the data has been seen before will be done when the data is in RAM, and if it is, a pointer will be written, if it has not been seen before, then this new data will be transmitted to the storage device through I/O and at the same time a new hash will be written into hash table. The signature of the data entering the device and its lookup in the signature index are performed before writing the data. Thus, no additional space is required on in-line deduplication since the process of optimization is already done before the process of storage, as shown in FIGURE 7. Besides, only one protocol is needed for retrieving data. But it is more sensitive to write latency for the overhead added by the deduplication. Since the server is capable of telling if the given data has been seen before, avoiding to transmit duplicated data from the client to the server, it matches well client side deduplication.

Archival storage and network file system Venti (Quinlan & Dorward 2002), cloud-based file systems S3QL (S3QL 2012) and OpenDedup (OpenDedup 2012), and general purpose file system and volume manager ZFS (Bonwick 2009) are using this approach in their systems. ^[8]

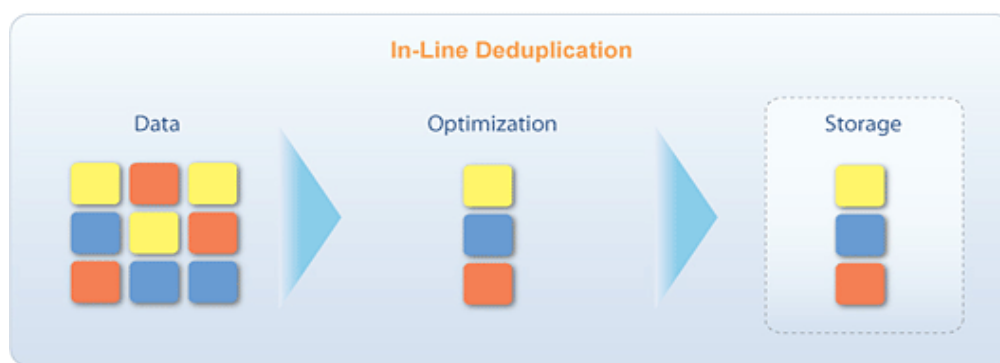


FIGURE 7 In-Line deduplication

We could also combine them into one which is called Adaptive Data Deduplication that combines the advantages of the two previous methods. Using in-line method when the requirement of performance is low and using offline method when the requirement of performance is high.

3.3 Deduplication based on storage position

Another way to classify data deduplication is according to the storage position. Mainly there are two ways—primary storage and secondary storage depending on where the deduplication is occurred.

3.3.1 Secondary deduplication

The first deduplication solution is mainly aimed at the backup data space as it came out, which is now widely known as secondary deduplication.^[9] It basically stores the data that is not so active and it is not directly accessible by the CPU. It usually has very large volume which contains a big amount of data including duplicates. In data center, backup data need to be backed up daily, weekly and monthly, we therefore have big loads of duplicate data, which is the fact that could make us understand why deduplication in secondary storage has been largely and widely used nowadays.^[10]

3.3.2 Primary deduplication

Primary storage, on the other hand, stores data that is near active or transaction data which is accessible to the CPU on the computer. Traditionally speaking, it is not necessary to perform a deduplication on a primary storage since firstly it itself has very small volume and more importantly there is not much duplicate data in primary storage in the past. But now, there are mainly three reasons that force us to pay more attention to the primary storage deduplication, firstly storage is growing too fast for IT staffs to deal with; secondly, the redundancy of virtualized server and desktop images

can be very high; thirdly, also the most important one, duplicates on primary storage can effect the performance of secondary storage deduplication. We usually use ratio to measure the deduplication, as shown in FIGURE 8. Space reduction ratios are typically depicted as “ratio:1” ^[10] From space reduction ratio we could also get the space reduction percentage, as shown in Table 1 and Figure 9. Deduplication ratios are dependent on the type data being processed. Backup data for example, is highly repetitive so that ratio can be as high as 20:1 which, however, is almost not existing in primary storage. 2:1 is much closer ratio for primary storage. ^[11]

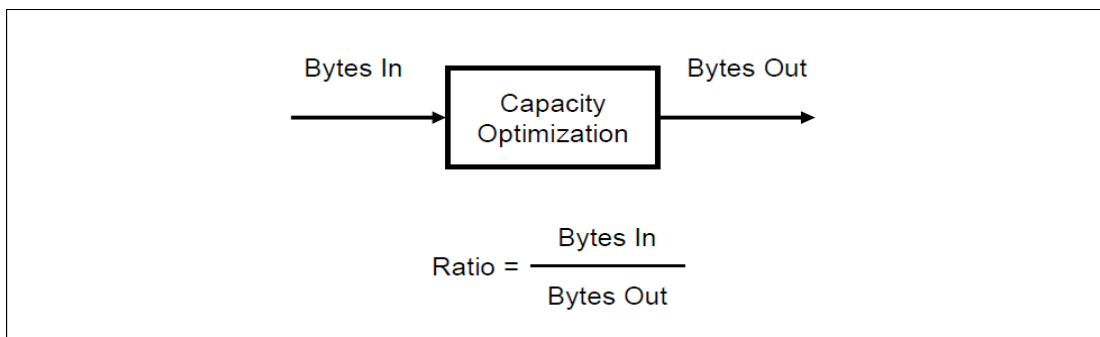


FIGURE 8 space reduction ratio ^[11]

The deduplication technology of Data Domain is based on backup deduplication and some new companies like Ocarica and Storwise put their effort in deduplication on primary storage

TABLE 1 space reduction ratios and percentages ^[11]

Space Reduction Ratio	Space Reduction Percentage = 1 - (1 / Space Reduction Ratio)
2:1	1/2 = 50%
5:1	4/5 = 80%
10:1	9/10 = 90%
20:1	19/20 = 95%
100:1	99/100 = 99%
500:1	499/500 = 99.8%

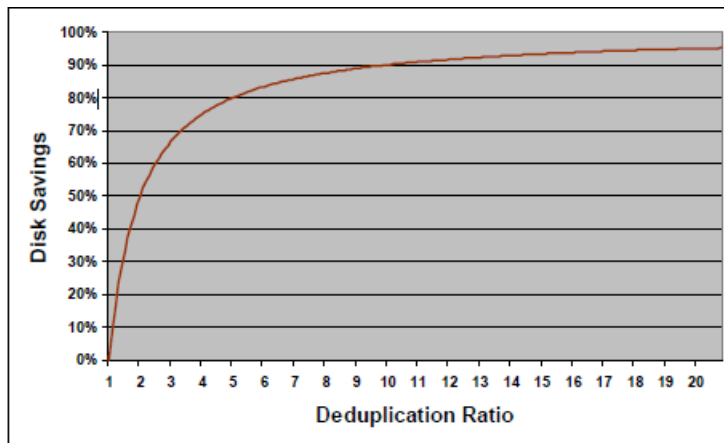


FIGURE 9 space reduction percentages ^[11]

3.4 Deduplication based on algorithm

There are two major ways of deduplication classified by algorithm, they are delta-encoding and hash-based deduplication.

3.4.1 Delta-encoding deduplication

We usually use the Greek letter delta (Δ) to denote the change in variable. FIGURE 10 shows an example of how this is done. The first value in the encoded file is the same as the first value in the original file. After this step, each sample in the encoded file is the difference between the current and last sample in the original file. ^[12]

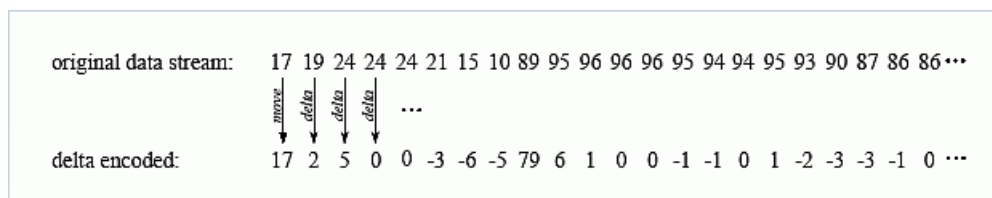


FIGURE 10 Delta-encoding

Thus, Delta-encoding deduplication is a technology that performing deduplication based on the difference between two files. Delta-encoding handles data in a very small granularity so that this algorithm does not care about the complete objects, it processes data in a form of states and differences. The concept of delta-encoding is actually first used in traditional compression field like 7-zip and bzip2, after that people are intended to use it in the deduplication area. ^[13] Under the knowledge of the basic idea of delta-encoding, we could understand the algorithm in a similar way. If now we have two files which have greatly high commonality, the first one refers to a reference file. With delta-encoding algorithm, an encoded version of the second file related to the reference file will be generated which means that instead of storing two highly similar files at the same time, the difference between the two files will be produced as a new file, replacing the two files with the reference one. One thing need to point out is that it only is meaningful when the size of the difference between the second one and the reference one is less than the reference one itself. ^[13] Specifically speaking, this algorithm works by comparing the contents of both files and identifying similar parts in both files. If the parts detected are shared by both files, then the pointer of the encoded file generated through delta-encoding algorithm will point to the reference file, and the pointer will point to the actual data when the parts identified is unique. The encoded version is technically called a diff or a patch file. ^[8]

The higher resemblance between the file being encoded and the reference file, the higher deduplication ratio or the reduction ratio we could get as we analyze the principle of delta-encoding. It has been applied in cases where files evolve over time for this advantage. It can also be used to save bandwidth on network transfers through allowing servers to send updated web pages in a diff form to optimize bandwidth utilization. ^[8]

We will discuss about hash-based deduplication, the other algorithm, which is more important and more widely used in detail in next chapter.

4 HASH-BASED DEDUPLICATION

Hash-based deduplication, just as its literal meaning, is a deduplication approach that based on hash function, which is the most important and widely used in most deduplication project nowadays.

4.1 Principle

Simply speaking, hash function is a function that creating a unique fingerprint for each unique file and its main purpose is to change large data sets of variable length to smaller data sets of a fixed length, as shown in FIGURE 11. When hash function is finished, every data chunk will have its own hash, and by comparing those hash of new chunks with hash of previously stored data, the deduplication process will find if there is a match or not. If there is a match found, then the new data will not be stored again since it has already been saved once but the data is replaced by a metadata reference that will refer to the fingerprint of the previous stored chunk. ^[8]

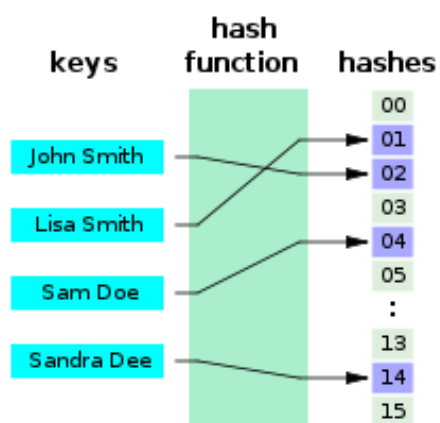


FIGURE 11 hash function

A secure cryptographic function must be used to hash chunks in order to have every chunks unique fingerprint. There are two main categories of cryptographic hash

function algorithms that are widely used nowadays, they are Message-Digest (MD) Algorithm including MD2, MD3, MD4 and MD5 designed by Professor Ronald Rivest of MIT and Secure Hash (SHA) Algorithm including SHA-0, SHA-1, SHA-2 and SHA-3 published by the National Institute of Standards and Technology (NIST). There is a possibility that two total different chunks are incorrectly identified to hold the same content so that they will get the same fingerprint at the end of process which will lead to delete the file that should be stored. ^[14] According to the birthday paradox, the probability of such occurrence is $\geq 50\%$ when we have $2^{b/2}$ items where b is the number of bits of the resulting hash function ^[15] 2128 items can be hashed before the probability of a false identification reaches 50% when SHA 256(belonging to SHA 2) is utilized where $b=256$, which proves that we do not need to worry about this problem technically. ^[14]

It can be divided into five steps if we summarize the whole process of hash-based data deduplication.

4.2 General process of hash-based deduplication

To implement hash-based deduplication, there are five general steps we need to do.

1) Data partitioning

According to the strategy of data partitioning, divides a collection of files into a number of data chunks and create a chunk index.

2) Feature selection

Based on content expectations, we select a unique feature that can mark it and add it to the chunk index.

3) Detection

Compare the feature of chunk to the existing feature in chunk index

4) Elimination of redundancy

If we have the same identification then do not store the new file

5) Data storage

If the identification differs from the existing feature then store the data on the disk and add the new feature to the chunk index.

4.3 Classification

An important part in hash-based deduplication process is the technique to divide complete data into chunks. The smaller granularity the chunks have, the higher possibility the redundancy have, therefore the higher deduplication ratio we can get. And of course the more chunks in common, the more space we can save. ^[8] There are basically three different hash-based deduplication approaches classified by granularity, they are whole file hashing (or file-level deduplication), block-level deduplication and byte-level deduplication described as follow. But whatever approach we take, the process are almost the same, as shown in FIGURE 12.

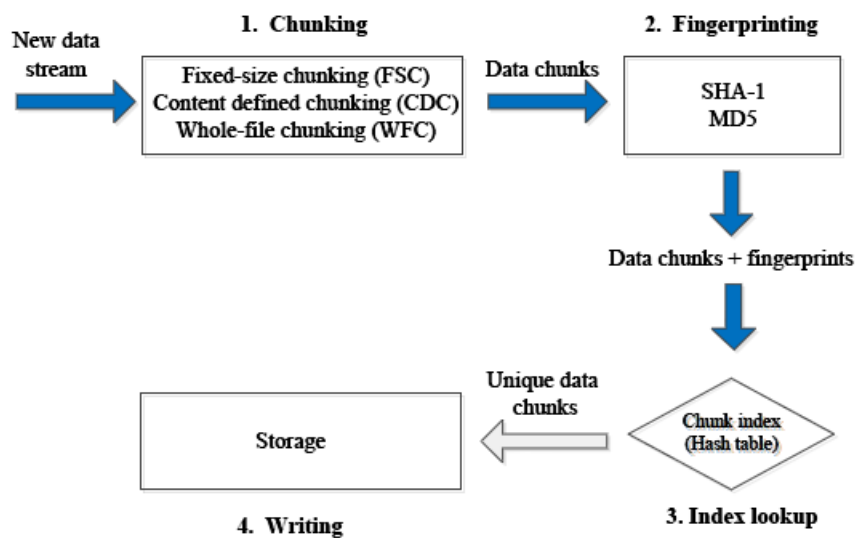


FIGURE 12 overview of data deduplication based on hash function ^[20]

4.3.1 Whole File Hashing (or File-level Deduplication)

File-level deduplication is often referred to as Single Instance Storage (SIS), considering the whole complete file as chunk to perform the deduplication process which is considered a fast and simple approach. ^[16] Although it is impossible for whole file hashing to eliminate duplicates within files, it is still very efficient method of decreasing storage consumption and the whole-file deduplication actually achieves 3/4 of the space savings of the most aggressive block-level deduplication which we will discuss later for storage of live file systems. ^[17]

Whole file hashing basically works in this way that when new data arrives at the storage system, the hash of the data will be first calculated usually through MD5 or SHA-1 hashing algorithm, then compare the hash calculated to those hashes which are already in hash table. Identifying them as identical if they have the same hash, and create a pointer that point to the stored file instead of storing new data. If the hash is new and has not been stored in hash table yet then identify the file as a new file and the system will update the hash table as well as storage the new data. FIGURE 13 shows the whole process. ^[18]

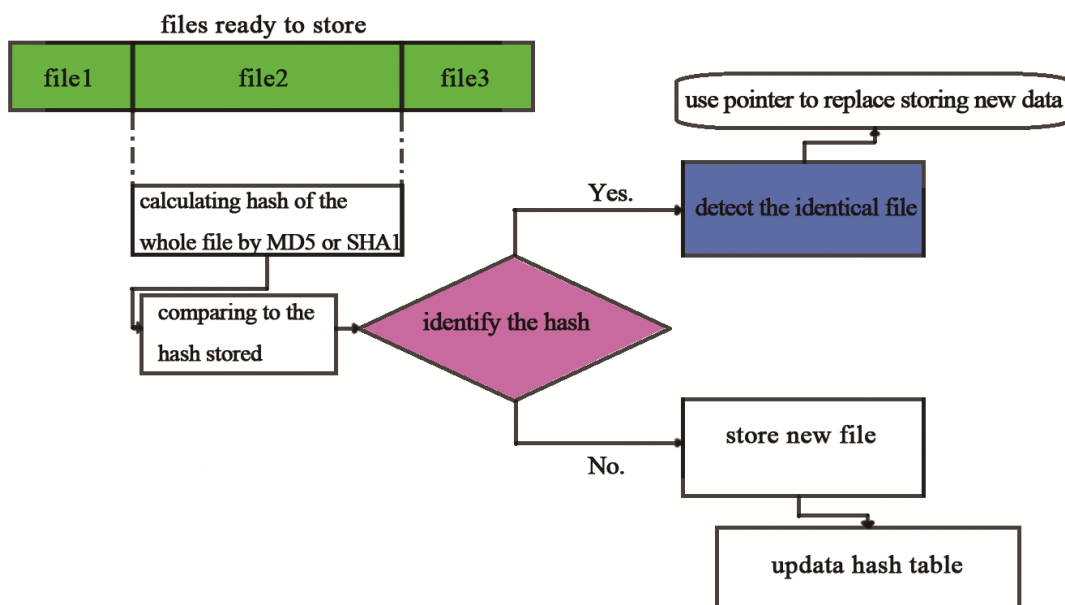


FIGURE 13 File-level deduplication

We now take SIS in Windows 2000 as an example to make better understanding of file-level deduplication. SIS is used in Windows 2000 to support the Remote Install Server which is an application that allows a server owner to configure a server with a set of machine images. There are different files in those machine images which at same time have a lot of duplicates that need SIS to deduplicate. As shown in FIGURE 14, there are mainly two components in SIS: kernel-level file system filter driver(called the SIS filter or just the filter) and user-level service (called the groveler). The role of groveler is of low-importance, its main task is to tracks changes to the file system, maintains a database of hashes of files in the file system, detect hash to see if the files are identical and report them to the filter for merging. The filter transparently deals with files that have identical contents and its main technique is to handle reads by redirecting them to common files and to handle writes using copy-on-close. Copy-on-close is a technique that the command of copy will not be implemented until all the updated related to this file are finished and only new data will be stored to the storage device. Generally speaking, the idea is SIS as a component of Windows 2000 detects files that own identical contents and merges them into special links that present the semantics of separate files while using the disk and file cache space of a single file in most situation. ^[19]

As we mentioned above, whole file hashing is a very simple and effective way to perform the deduplication , but because of the avalanche of hash function, the hash will be total different even there is a little difference between two files which will lead that a lot of duplicates will not be detected within files. Block-level deduplication is brought out to solve this problem.

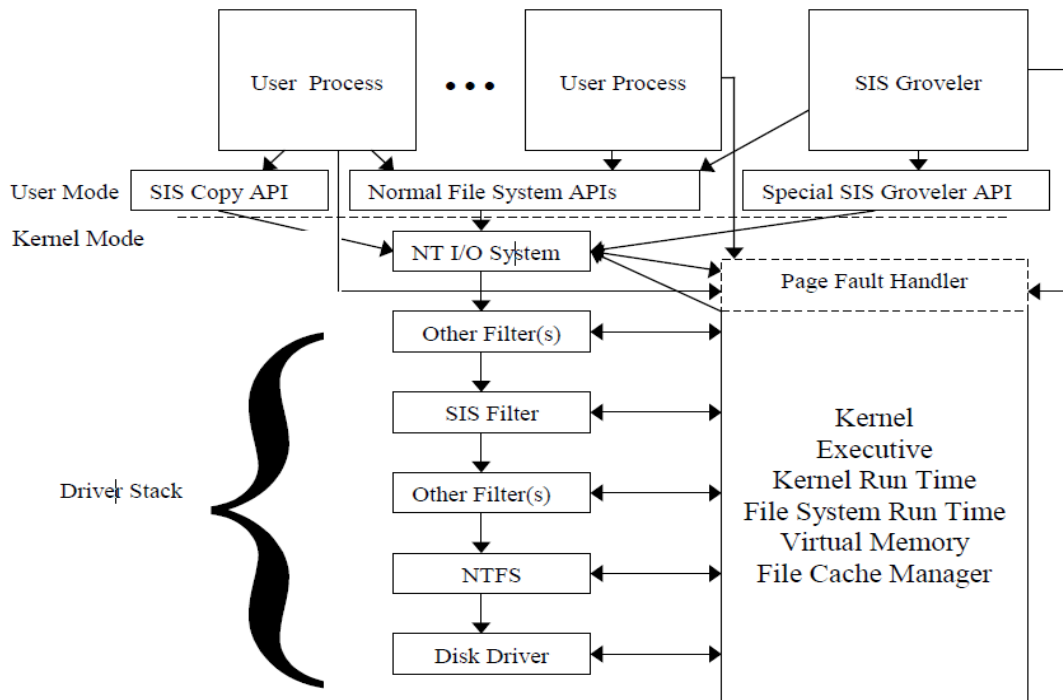


FIGURE 14 SIS Architecture ^[19]

4.3.2 Block-level deduplication (or Sub File Hashing)

Block-level deduplication is a technology that before data deduplication is performed, the data will be broken into a lot of small pieces. Compared to whole file hashing, sub file hashing has smaller granularity, which means a higher deduplication ratio. But on the other hand, block-level deduplication needs a much bigger hash table since more hashes will be generated, which means more computing time is needed when duplicates are being determined and the backup performance will be more affected by the deduplication process. ^[21]

The basic idea of block-level deduplication is similar to file-level deduplication, after the whole file has been divided into chunks, hash of each chunk will be generated and detected if the hash is already in hash table or not, if the hash is identified identical, then there will be a pointer pointing to the location of stored data, if the hash is unique, it will be stored in hash table as a new one and data will be written to disk, as shown in FIGURE 15. ^[16]

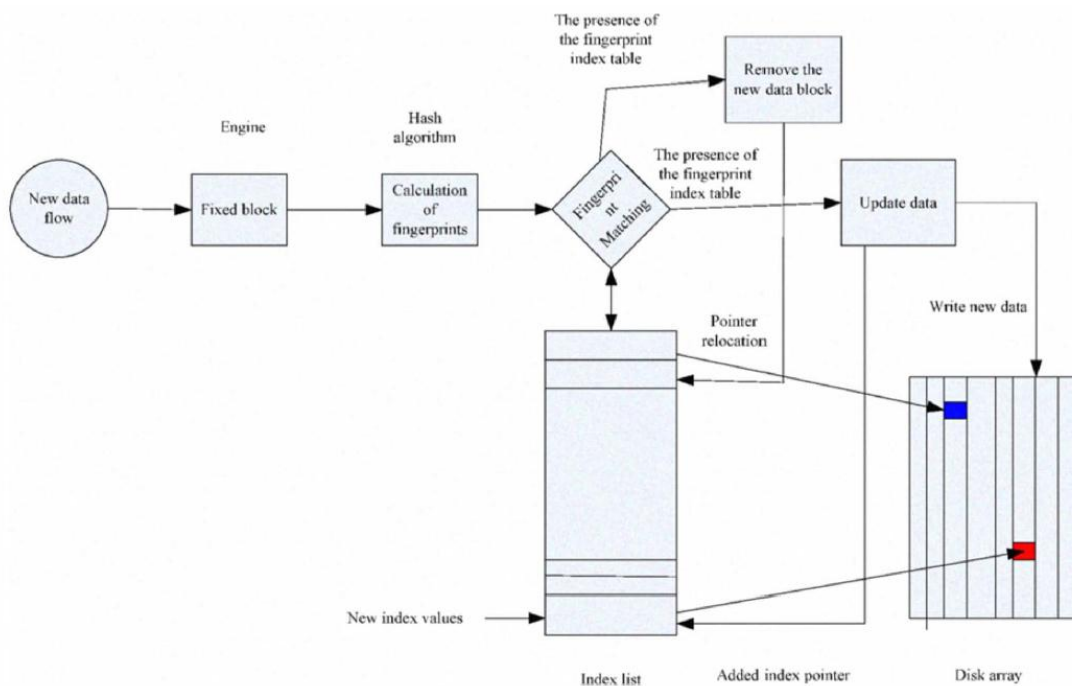


FIGURE 15 block-level deduplication ^[16]

The most two common types of block-level hashing are fixed-sized chunking (FSC) and variable-length chunking (or content-defined chunking, CDC) which are classified by the variability of data's length.

Fixed-sized chunking (FSC) Fixed-sized chunking is the approach that the file will be split into fixed-size chunks before the deduplication process. The advantage of this method consists of ease of implementation and fast chunking performance. Besides, the performance of I/O is improved since the chunk size can be aligned with the disk block size. But there are also two main disadvantages here, the first one is it has lower throughput than whole file chunking as we mentioned above, the second one is the boundary shifting problem that the hash of all chunks will change if there is update of a single file such as adding or removing which means that this approach could be more properly used in the system which do not update their data too frequently. ^[8]

This approach is often used in content addressable storage (CAS), for example, Venti (Quinlan & Dorward 2002) and Solaris`ZFS (Bonwick 2009). According to the research of Nath, a chunk size of 1 KB or 2KB can provide high savings in disk space which can up to 84% in CAS system. ^[22]

Variable-length chunking (or Content-defined chunking, CDC) Content-defined chunking, differing from fixed-sized chunking, the chunk is divided by a rolling hash function that calculates chunk boundaries based on the file contents. A sliding window technique is used to check the boundary by calculating the fingerprint of Rabin. The basic idea is that first calculate the FP(fingerprint) in the window W, making mod with the given integer D and comparing with the given remainder r. The right side of the window (like W2) will be set to the boundary otherwise the window will keep sliding to next byte(like W1) and the process will continue till to the end of file, as shown in FIGURE 16. In this way, the approach will have a very fine resistance to the chunk boundary shifting problem, properly solving the problem that fixed-sized chunking has, which means that it is a great choice for dealing with dynamic data which are updated frequently. Also CDC is efficient for deduplicating correlated data. ^[8] A new fingerprinting algorithm called Pretty Light and Intuitive (PLAIN) is introduced by Spiridonov as part of LBFS. This technology has higher algorithm than Rabin hash function by eliminating the randomization. ^[24]

The disadvantage of CDC is obvious since we have known well about the algorithm of this approach. It increase significant latency overhead to the deduplication for calculating those chunk boundaries. ^[8]

This method is mainly used in Low Bandwidth File System (LBFS) (Muthitacharoen, Chen, & Mazieres 2001) to chunk and detect similarities between files. ^[8]

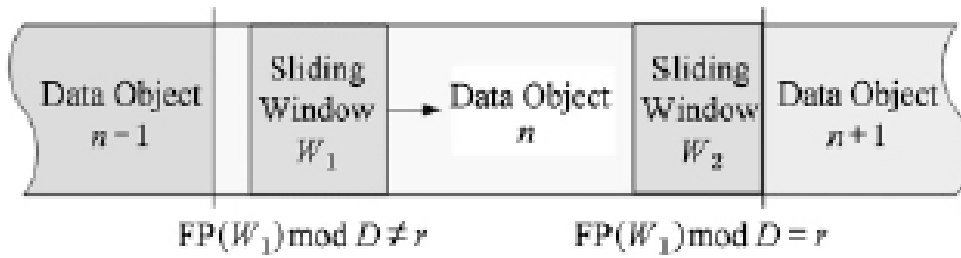


FIGURE 16 The principle of content-defined chunking ^[8]

4.3.3 Byte-level deduplication

Byte-level deduplication is more granular than block-level deduplication, but it needs more processing power therefore. Usually this approach is implemented in the form of a purpose-built appliance.

The idea of this approach is that the data is detected and compared in byte level to a previously stored data. Since it works in a post-process way, it means that all the data will first be stored onto a disk appliance and then the process begins, which leads to the fact that a large space of disk appliance is required. And the step is similar to block-level deduplication, as shown in FIGURE 17

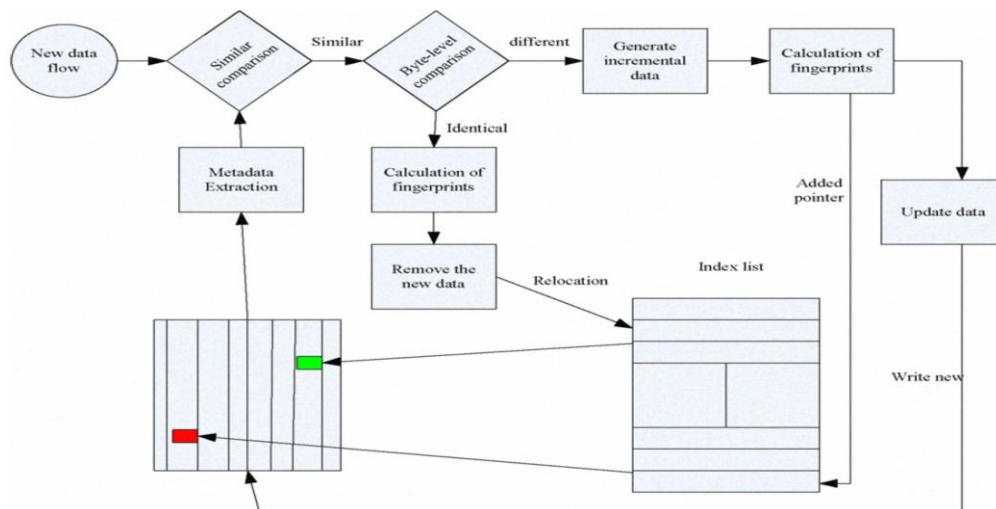


FIGURE 17 byte-level deduplication

To summarize, just as shown in FIGURE 18, these three approaches of deduplication actually use the same algorithm with different granularity.

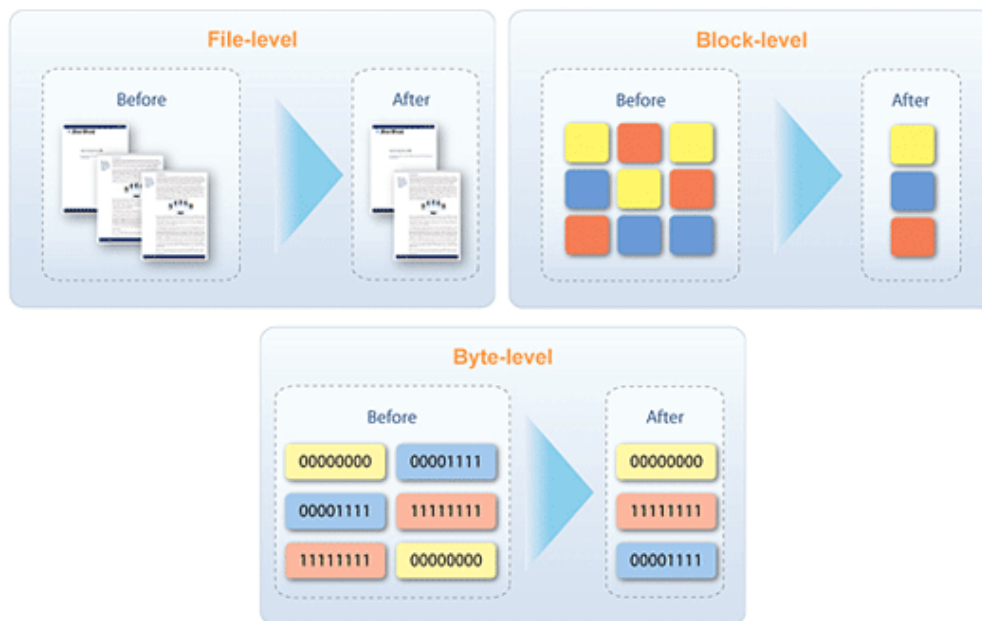


FIGURE 18 three different ways of hash-based deduplication

4.4 Problem of hash-based deduplication and related strategy

Although the technology of data deduplication has made great contribution to data storage, we still have to face the problems that this technology exists. Following are some problem and related strategy.

4.4.1 Hash collision

Hash collision, simply saying, is that two different data generate the same hash. We take a famous hash function CRC32 as an example. If we feed “plumless” and “buckeroo” in this function, it generates the same hash, as shown in FIGURE 19. ^[23] If hash collision occurs in data deduplication, it means that the two total different data will be identified two same data which will directly lead to the loss of data. But the more bits in the chunk identifier, the smaller is the chance of collisions. For example, the 128-bit hash function MD5 is processed on a dataset of 1 TB with a chunk size of

8 KB, the probability of one chunk ID collision is about 2.3×10^{-23} . And for the 160-bit SHA-1 function, the chance will be only 5.3×10^{-33} . Therefore it is not a issue if we have enough amount of data to be stored. Besides, long hashes will reduce the chance of hash collision but it requires more processing time,so the selection of the hash function is greatly application dependent. Usually, MD5 and SHA-1 are the most two poplar functions which are widely used in deduplication. ^[24]

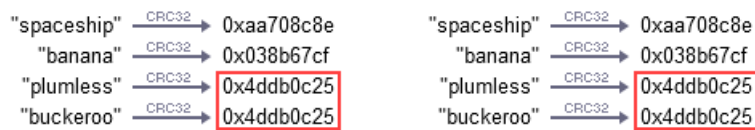


FIGURE 19 hash collision ^[23]

4.4.2 Granularity

Depending on the granularity of chunks, the effect of data deduplication will be greatly different. Those data which are divided into big size chunks will own a relatively small hash table which means fewer ID lookup is going to be processed. Also, the usage of CPU and I/O operations will be kept in a small range. However, bigger chunks means that there will be a lot of redundant data which can not be detected and those space taken by duplicates can not be saved.

Those data which has fine-grained chunks, on the other hand, are generating a bigger hash table cause of the huge amount of chunks and therefore CPU and I/O load will be very high as well as gaining more saving space.

4.4.3 I/O optimization

In data deduplication process, with more and more hash added into hash table, more space will be taken for the table, making it impossible to store all the hash in the RAM. Besides, since the position of hashed have high randomization and low hit rate

it will cause that disk query will be frequently processed which will seriously affect the performance of I/O. So the technology of I/O optimization is urgently needed to increase the throughput in data deduplication. ^[25]

So far, there are three main ways to improve this situation, Summary Vector, Stream-Informed Segment Layout (SISL) and Locality Preserved Caching (LPC). ^[25]

Summary Vector, basically, is using a Bloom filter to detect if the hash is in the hash table. By testing the existence of hashes in advance, we can reduce the number of times to do the lookups in the hash table. Here is the way that how Summary Vector works, as shown in FIGURE 20: At first all bits in the array are 0. Once there has one insertion, bits will be specified by several hashes, and h1, h2, h3 of the fingerprints of the segments are set to 1, as shown in (a). If the situation goes like (b), a 0 is existing in a new detection, that means this segment can not be in the hash table. But if the new detection result is as same as the old, that means there is a high probability that the segment is already in the hash table, but not a guarantee. ^[25]

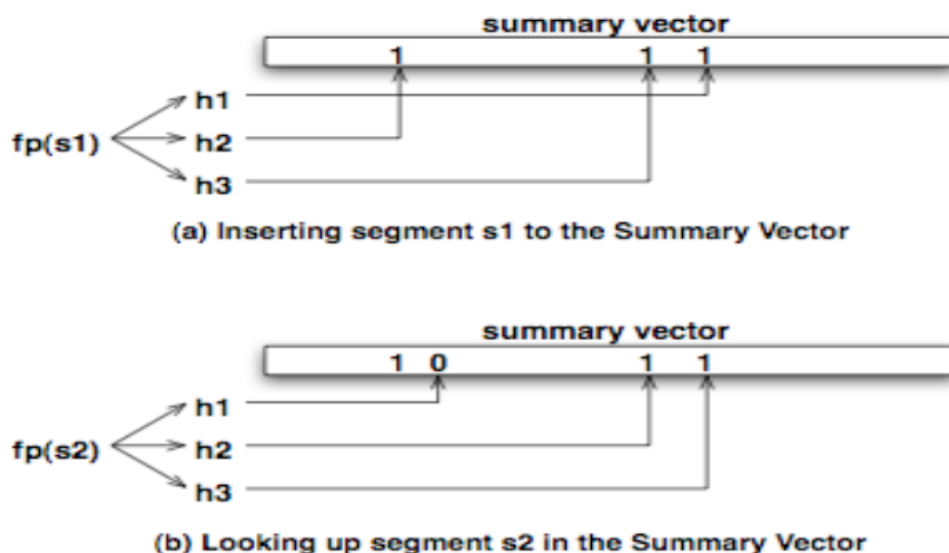


FIGURE 20 principle of Summary Vector ^[25]

Stream-Informed Segment Layout (SISL) is used to create spatial locality for both segment data and segment descriptors and to enable Locality Preserved Caching. In most backup applications, segments are likely to reappear in the same or very similar sequences with other segments. When a duplicate segment x is contained in a new data, a high chance that other segments in its locale are duplicates of the neighbors of x exists. It is called segment duplicate locality, which SISL is designed to preserve. This technique helps the system achieve high read throughput since fewer I/Os are needed to operate when segments of the same data are written to a container together. ^[25]

Locality Preserved Caching is used to accelerate the process of identifying duplicate segments. Traditionally, if we want to find a hash we need to go through the whole index, therefore the ratio of missing a cache will be very high. LPC is an application that takes advantage of segment duplicate locality so that if a segment is a duplicate, the base segment is highly likely cached already. ^[25]

5 WINDOWS SERVER 2012

Windows Server 2012 is the sixth release of Windows Server, which is the server version of Windows 8 and succeeds Windows Server 2008 R2. Data deduplication, as a new feature, has been added into Windows server 2012 with which you could store data more efficiently. ^[26]

5.1 Introduction

Data deduplication in Windows Server 2012 includes detecting and deleting duplicates. By dividing data into variable-sized chunks (32-128 KB), identifying duplicate chunks, and maintaining the only copy of each chunk. Redundant copies of the chunk will be replaced by a pointer to the single copy. ^[27]

5.1.1 Features of deduplication in Windows Server 2012

It stores more data in less physical space. It does a better job in storage efficiency than was possible by using features such as SIS or NTFS compression. In Windows Server 2012, data deduplication uses subfile variable-size chunking and compression, which deliver optimization ratios of 2:1 for general file servers and up to 20:1 for virtualization data.

Also, data deduplication has high scalability, efficiency and flexibility. It can run in a very high speed as high as 20 MB of data per second and can simultaneously process multiple volumes without affecting other workloads on the server. Deduplication can be completely stopped if the server is detected too busy. Besides, the controller can fully control the time when the deduplication process will run.

This feature has high reliability and data integrity as well. The integrity of the data is maintained. For all metadata and the most frequently referenced data, data deduplication remains redundancy to ensure that the data can be recovered in the event of data corruption.

Windows Server 2012 has its optimization functionality built into Server Manager and windows PowerShell which means that the data deduplication can be carried out through GUI or commands. ^[27]

5.1.2 How does it work

The approach that Windows Server 2012 uses belongs to CDC (content-defined chunking) ,dividing the data into pieces of chunks, identifying the identical part, deleting the redundancy, as shown in FIGURE 21.

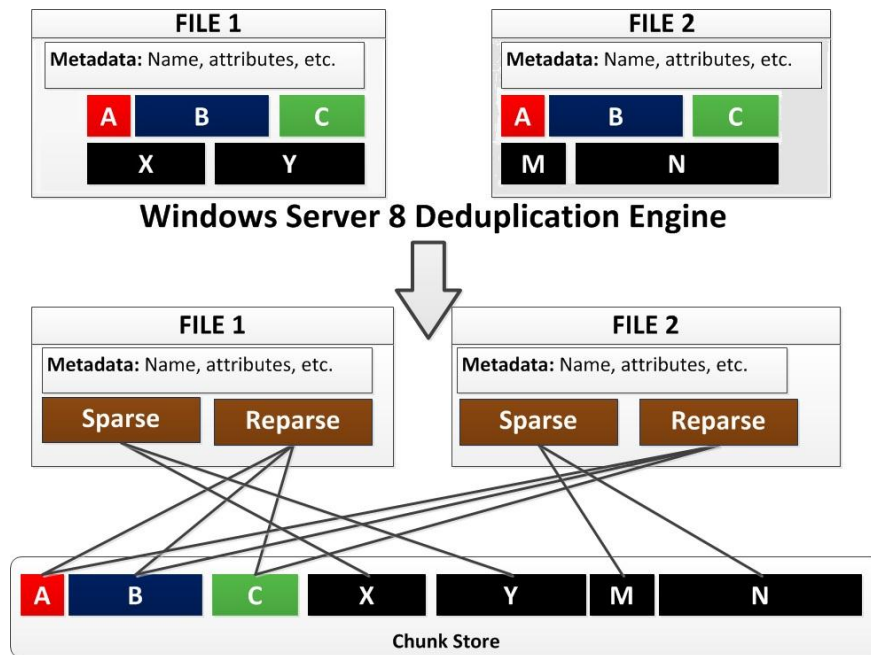


FIGURE 21 deduplication in Windows Server 2012 ^[28]

After the function of deduplication is enabled, the volume will contain four main parts: unoptimized files, optimized files, chunk store and additional free space.

5.1.3 The components of deduplication

We can say that filter driver is the most important part that with it the whole deduplication technology could work as we expect.

We could use server manager, powershell, WMI which make the management system, to manage the dedupe service which get control of dedupe jobs. And those dedupe job are actually working with dedupe filter driver. File metadata plays a role that tell the files where all the data has gone and when deduplication is conducted the metadata will point to the regular storage and chunk store as well, as shown in FIGURE 22.

Also with post-process approach being adopted, the deduplication will wait for a correct time to do their job in a background, and a correct time refers to a situation that the deduplicatio will not affect too much on the performance of system.

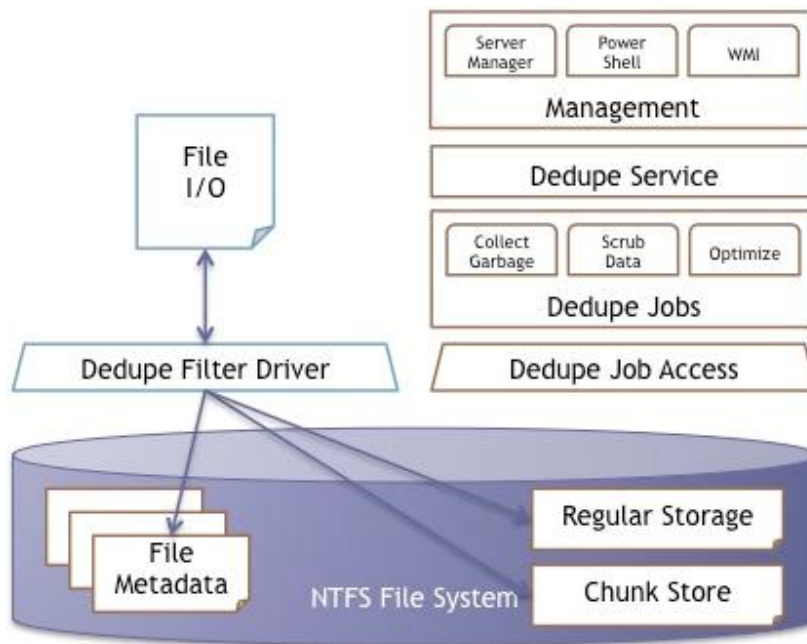


FIGURE 22 components of deduplication ^[28]

5.2 Installing

I am going to use VMware Player to install Windows Server 2012.

STEP 1 Choose Create a New Virtual Machine after opening the software, as shown in FIGURE 23.

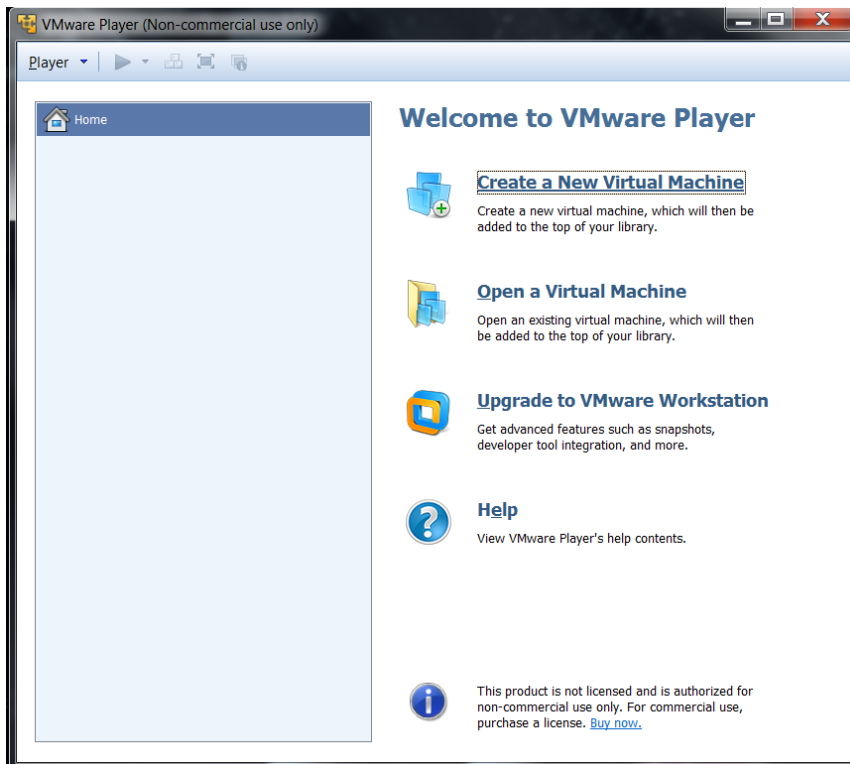


FIGURE 23

STEP 2 Choose I will install the operating system later

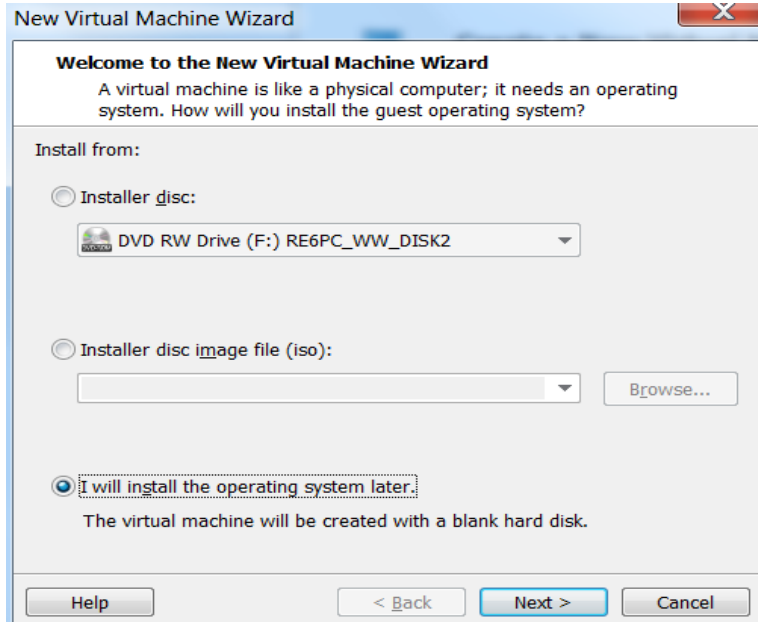


FIGURE 24

STEP 3 Choose Microsoft Windows and the version is Windows Server 2012

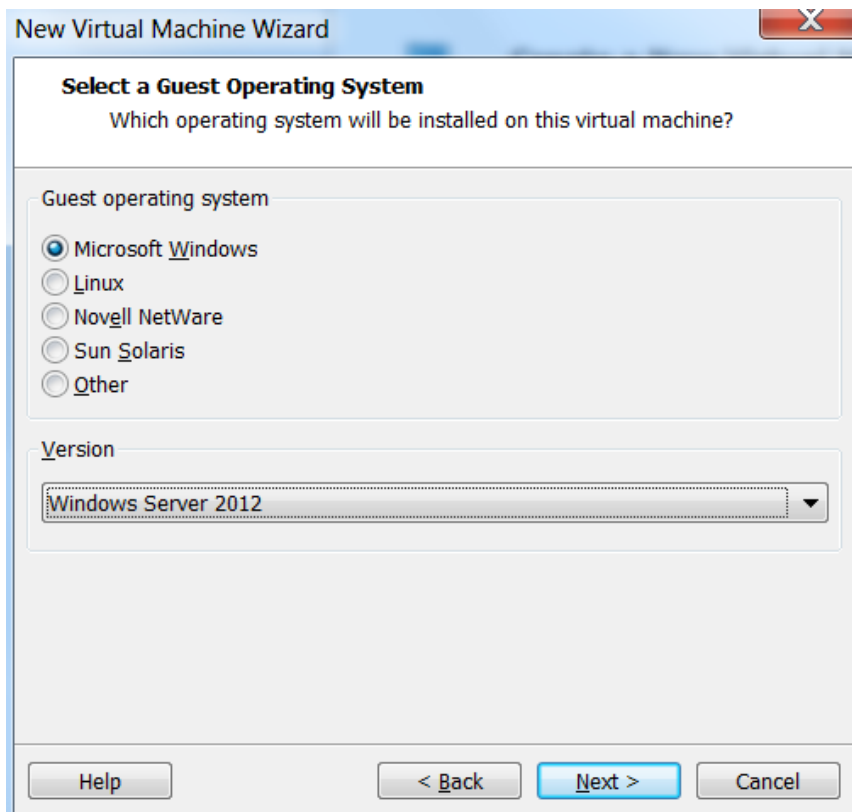


FIGURE 25

STEP 4 I choose 40 GB as the maximum disk size

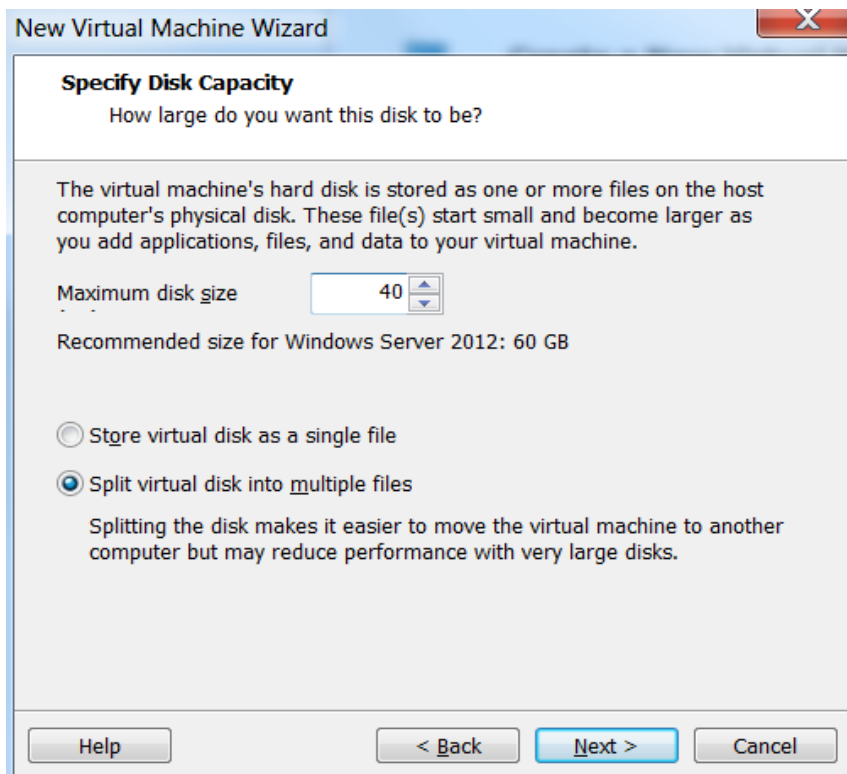


FIGURE 26

STEP 5 Configure the hardware. I choose a 4 GB memory and 2 processors. Also add the Windows Server 2012 Evaluation ISO in it.

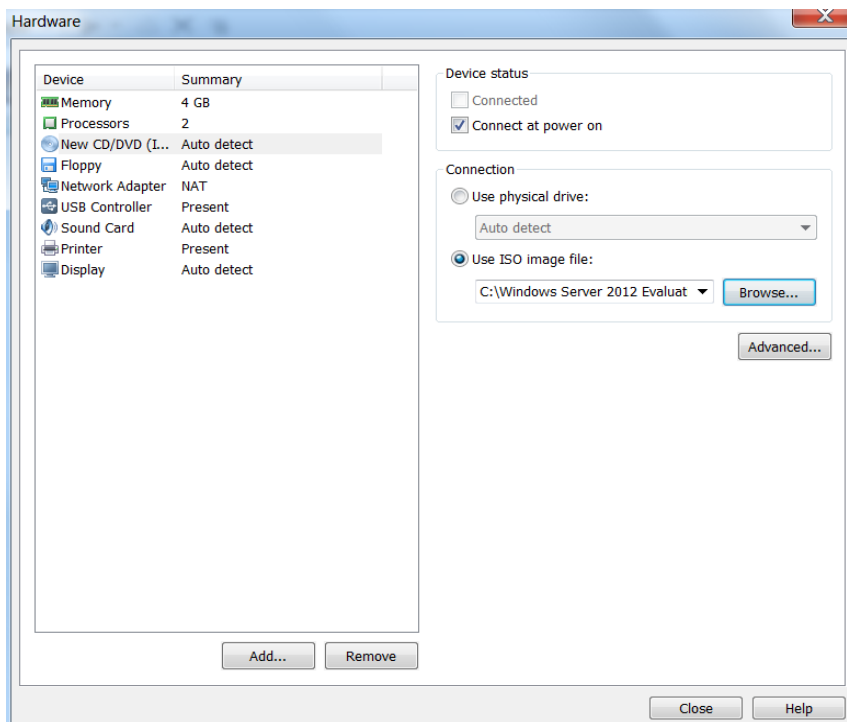


FIGURE 27

5.3 Data deduplication in windows server 2012

We can install the system after configuration. When the process is finished, we can then log in and the system interface will appear, as shown in FIGURE 28.

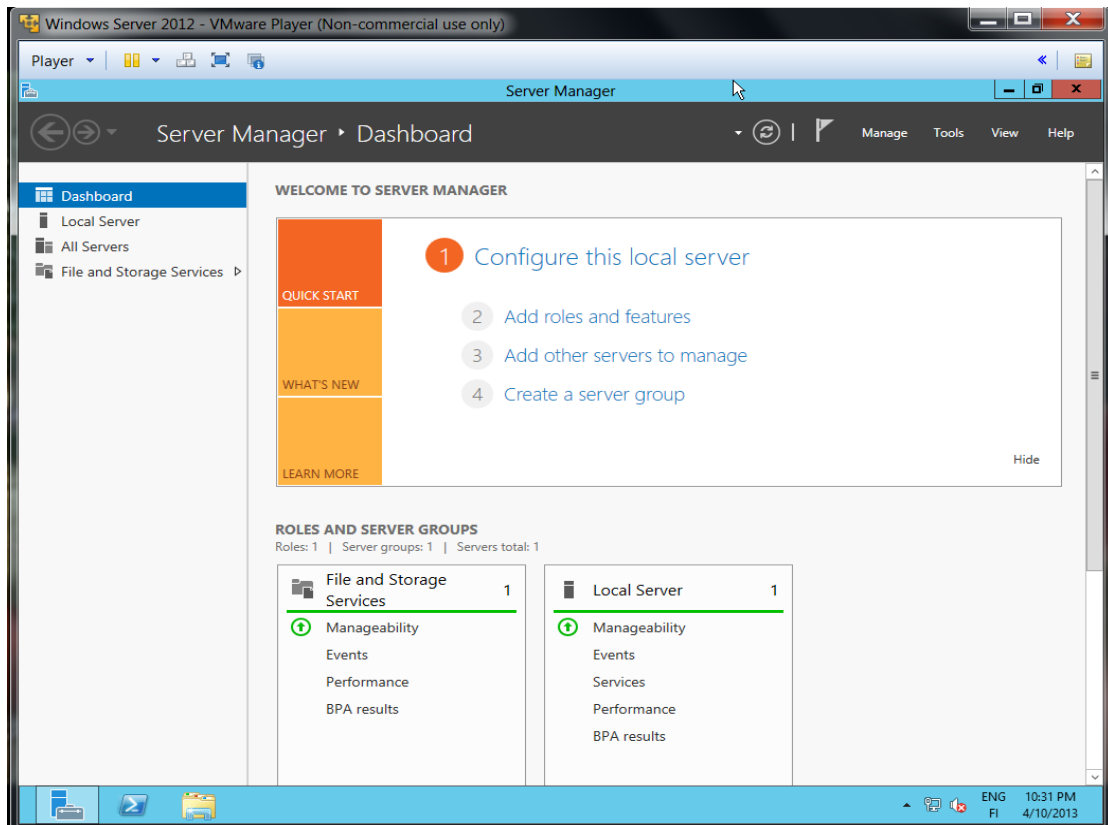


FIGURE 28

5.3.1 Add deduplication feature

STEP 1 Choose add roles and features

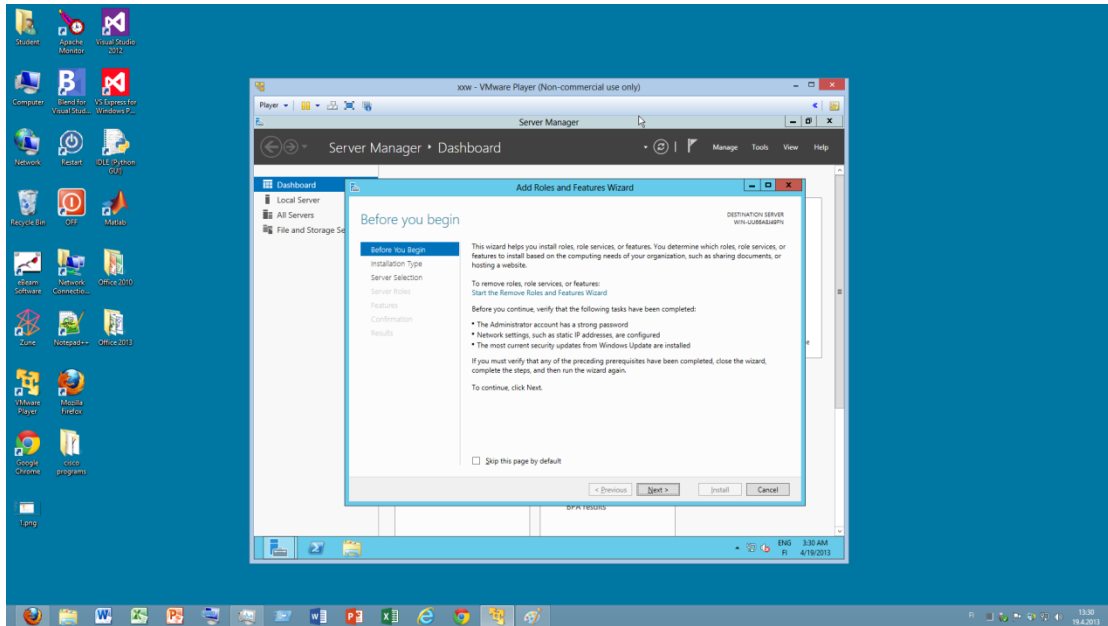


FIGURE 29

STEP 2 Select server roles. Tick the File Server and Data Deduplication which are in File and ISCSI Services, as shown in FIGURE 30

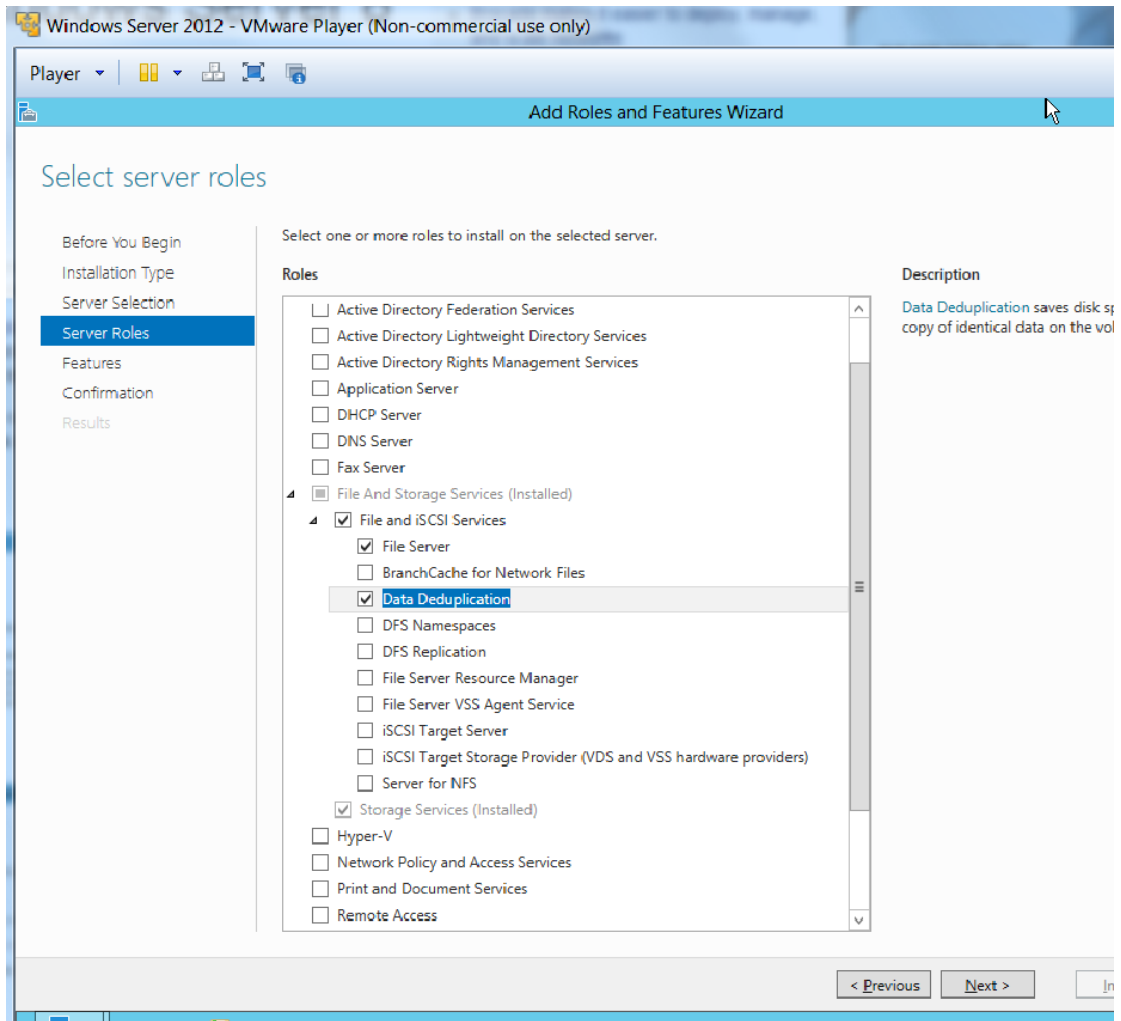


FIGURE 30

STEP 3 Confirmation. Confirm that File Server and Data Deduplication are added, as shown in FIGURE 31

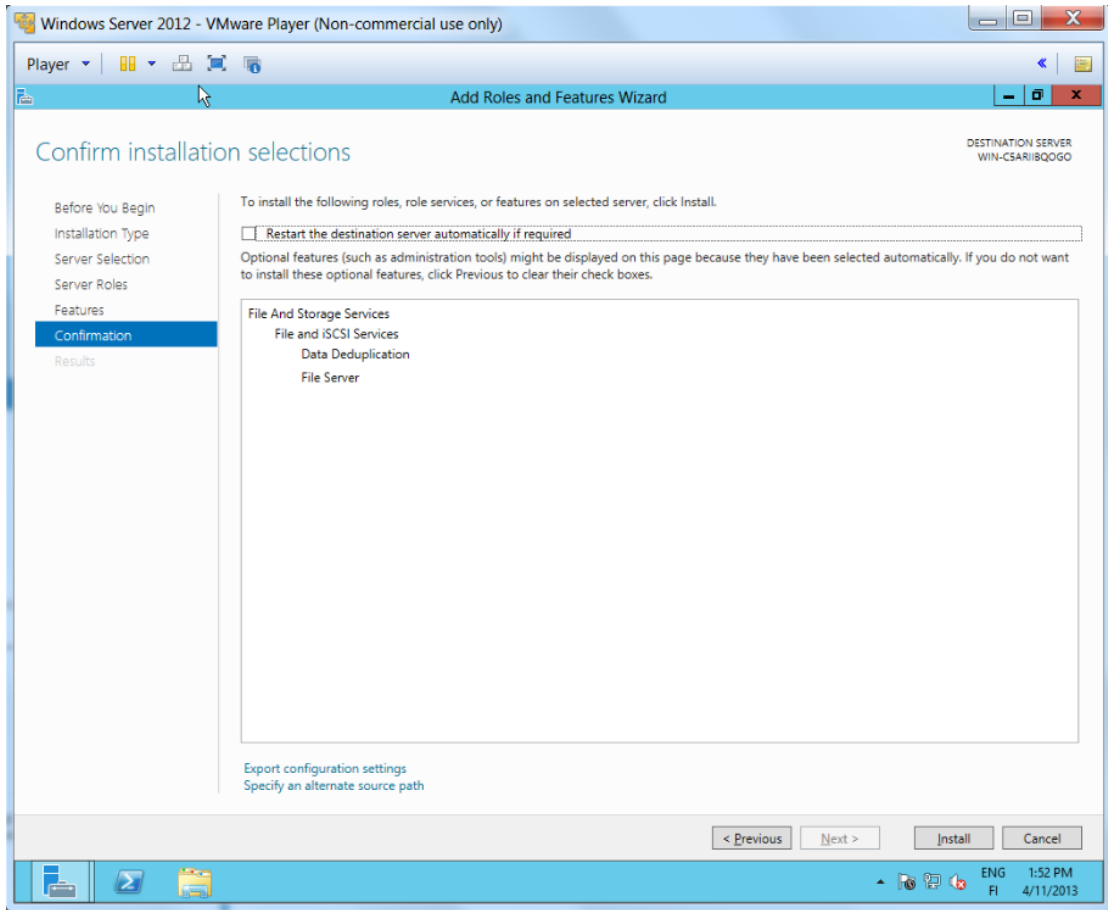


FIGURE 31

STEP 4 Installing

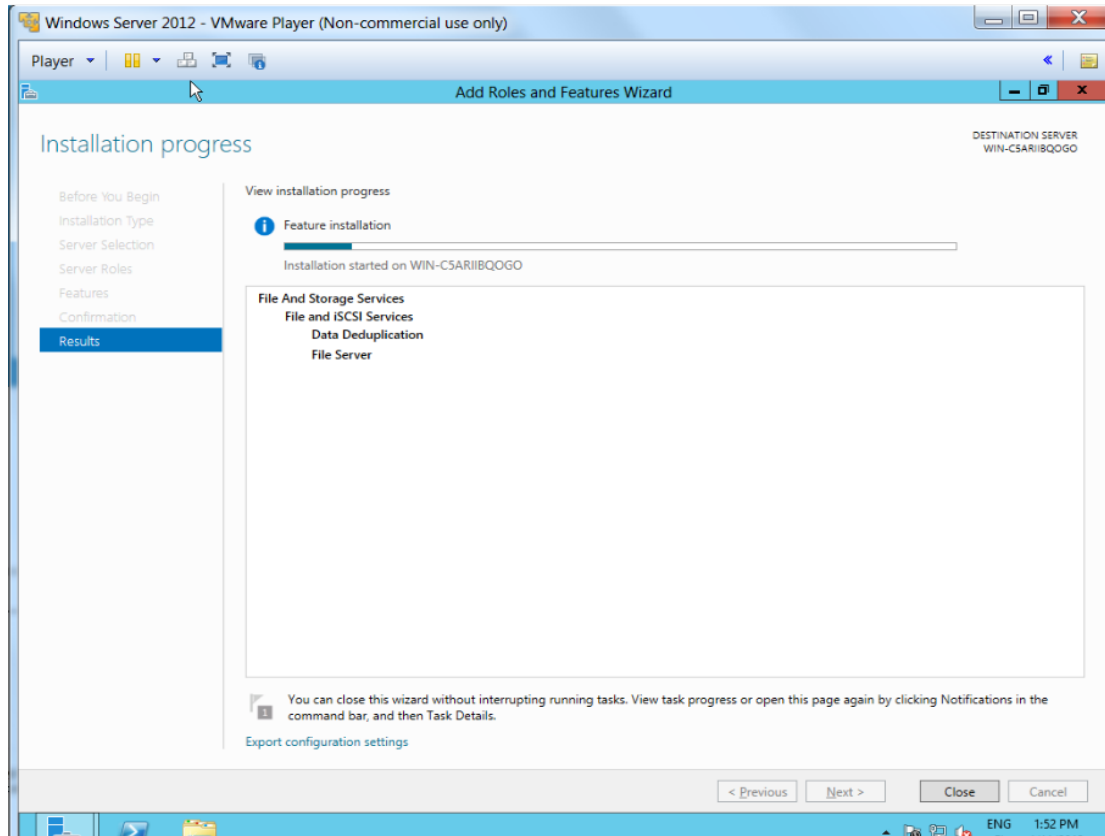


FIGURE 32

5.3.2 Enable Deduplication

STEP 1 Right-click the disk, there is a Configure Data Deduplication, as shown in FIGURE 33.

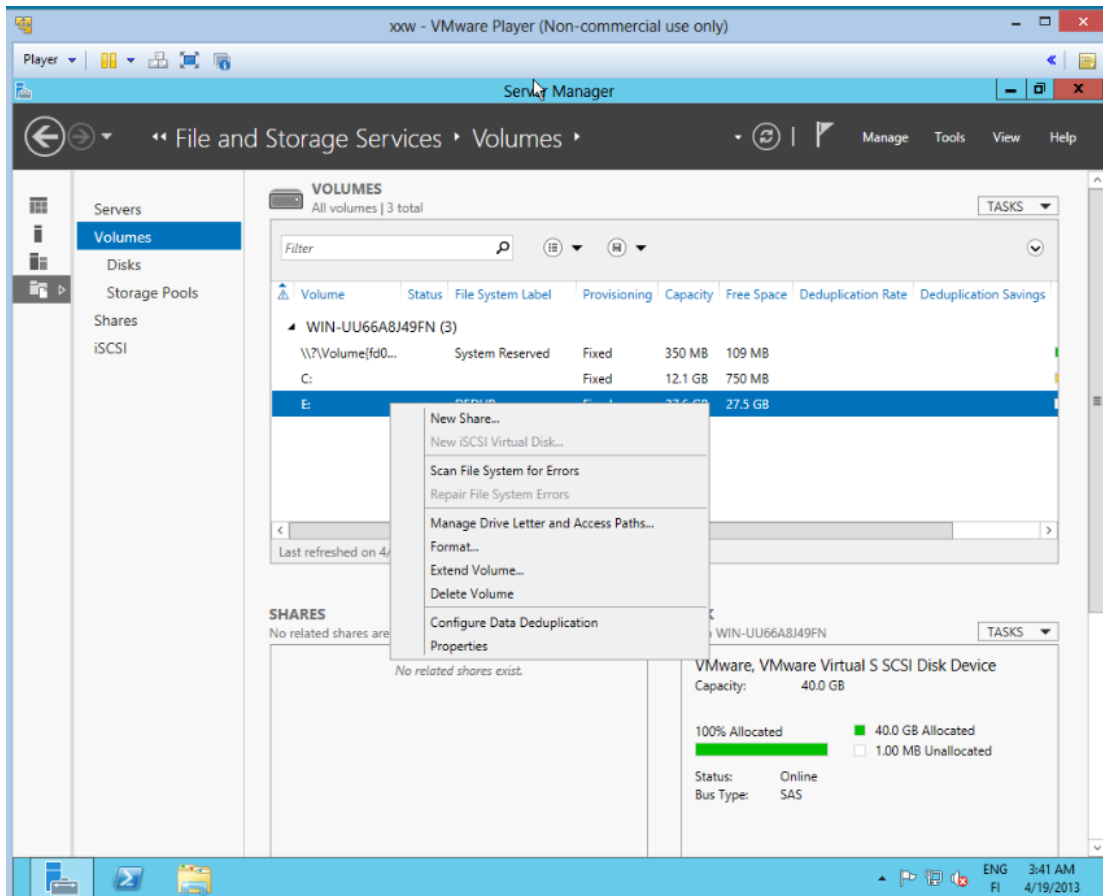


FIGURE 33

STEP 2 Enable Data Deduplication. Also, data should be configured carefully so that when duplicates are old enough they should be deduplicated. In our case, in order to show the results immediately we set 0, as shown in FIGURE 31, which means that all the data can be deduplicated even though they are still fresh.

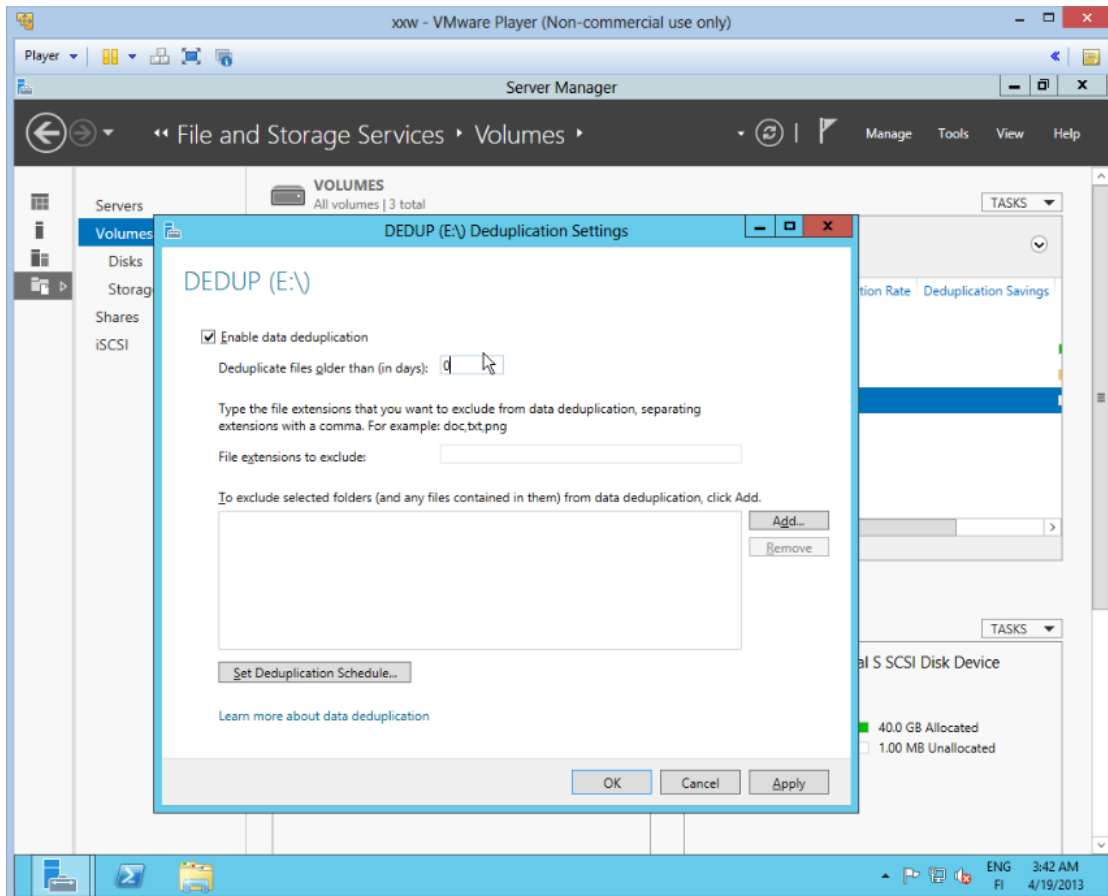


FIGURE 34

STEP 3 Set Deduplication Schedule. We can set up the schedule so that we are able to make the deduplication start at any time we want, as shown in FIGURE 35. For our case, in order to show the result immediately, I choose to begin the process manually which I will cover below.

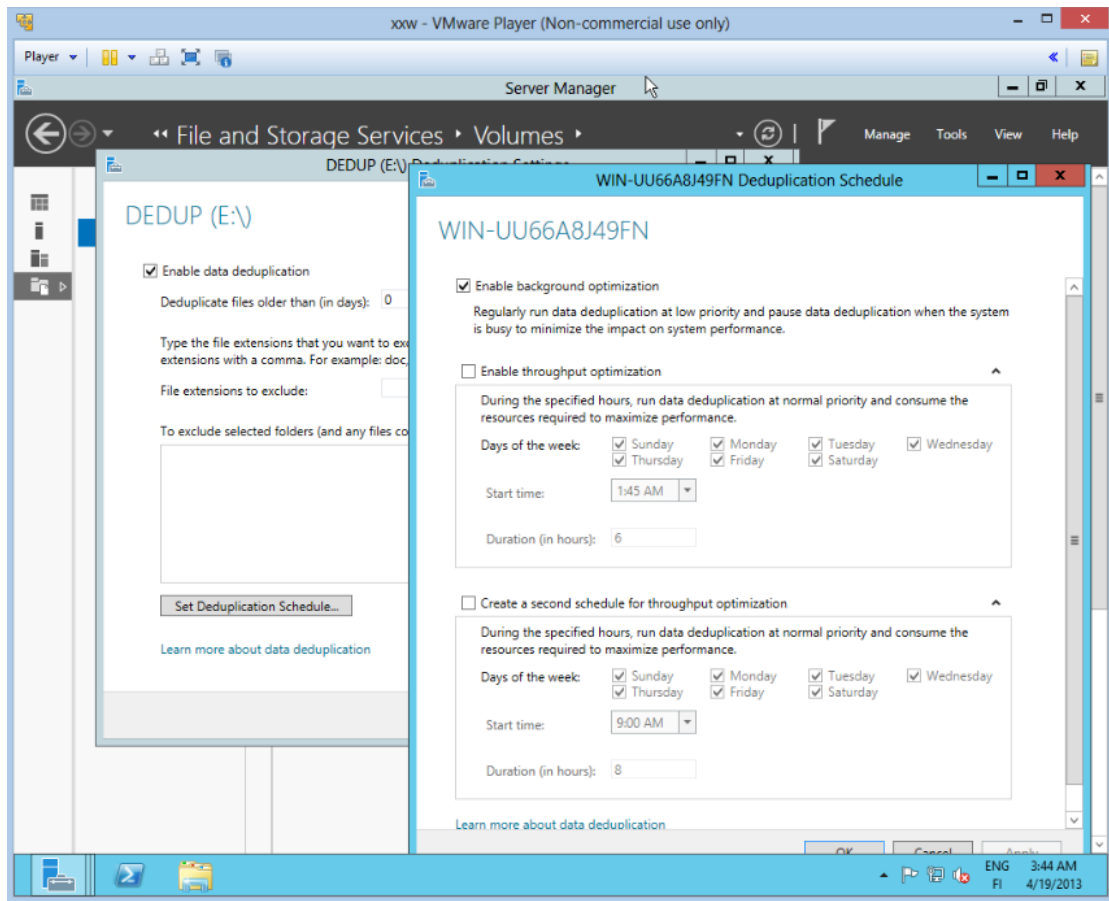


FIGURE 35

5.3.3 Process Deduplication

As FIGURE 36 shows, we are going to use FILE to process the deduplication. After making 4 copies of the original file, it takes about 22.6 GB, 4.92GB left.

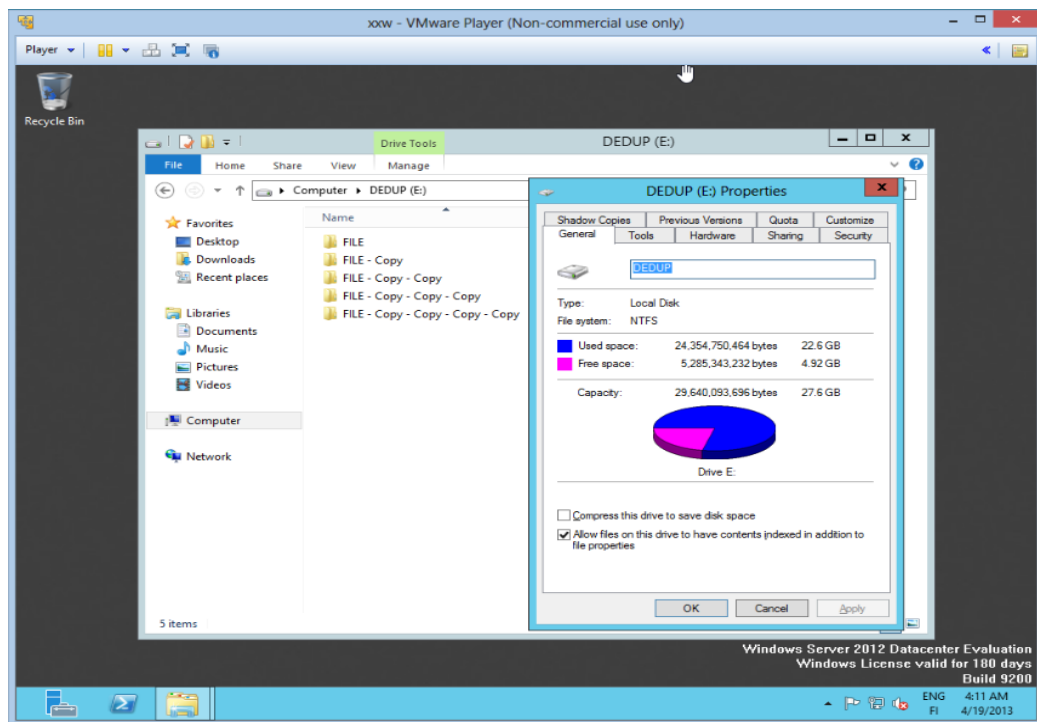


FIGURE 36

It is very convenient for users to carry out deduplication through Powershell. First, open the Windows Powershell. Since we want to deduplicate E: disk, so we need to type the command below :

start-dedupjob e: -type optimization

As we can see in FIGURE 37, the ScheduleType is Manual, which means we can actually manually start deduplication without setting schedule with Powershell. To see the status of deduplication ,type the command below:

get-dedupjob

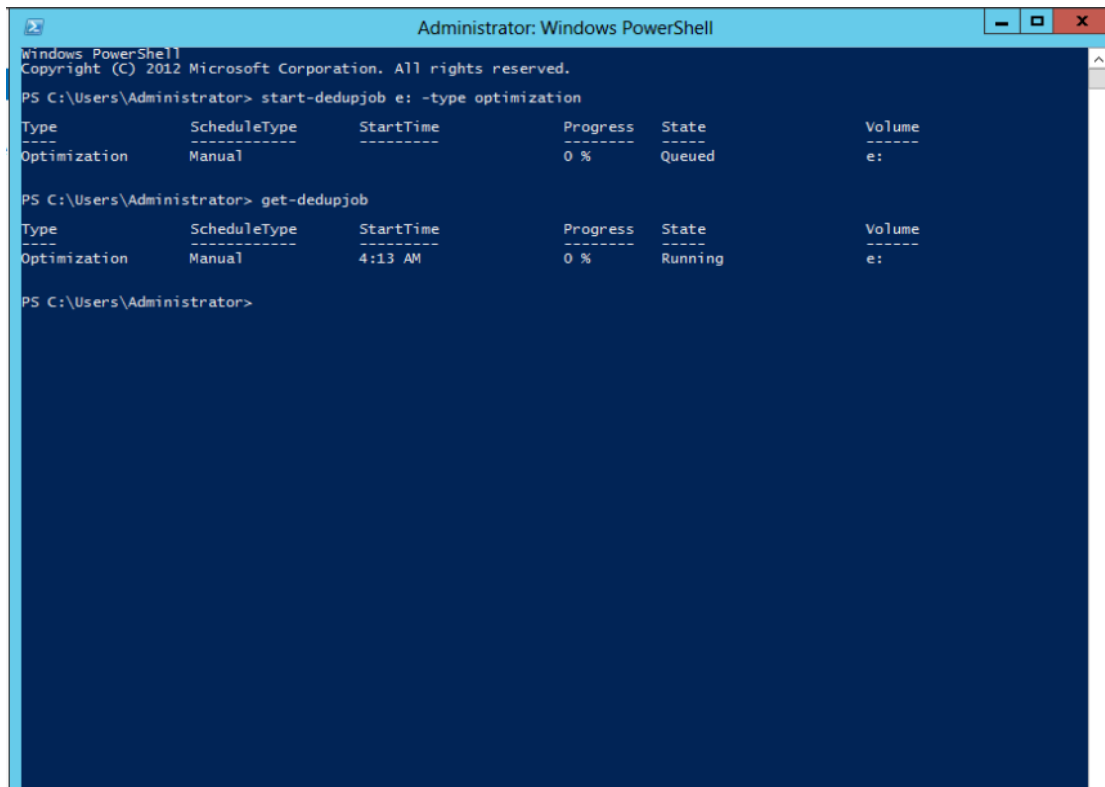


FIGURE 37

As you can see in FIGURE 38, it takes about 20 minutes to finish the job.

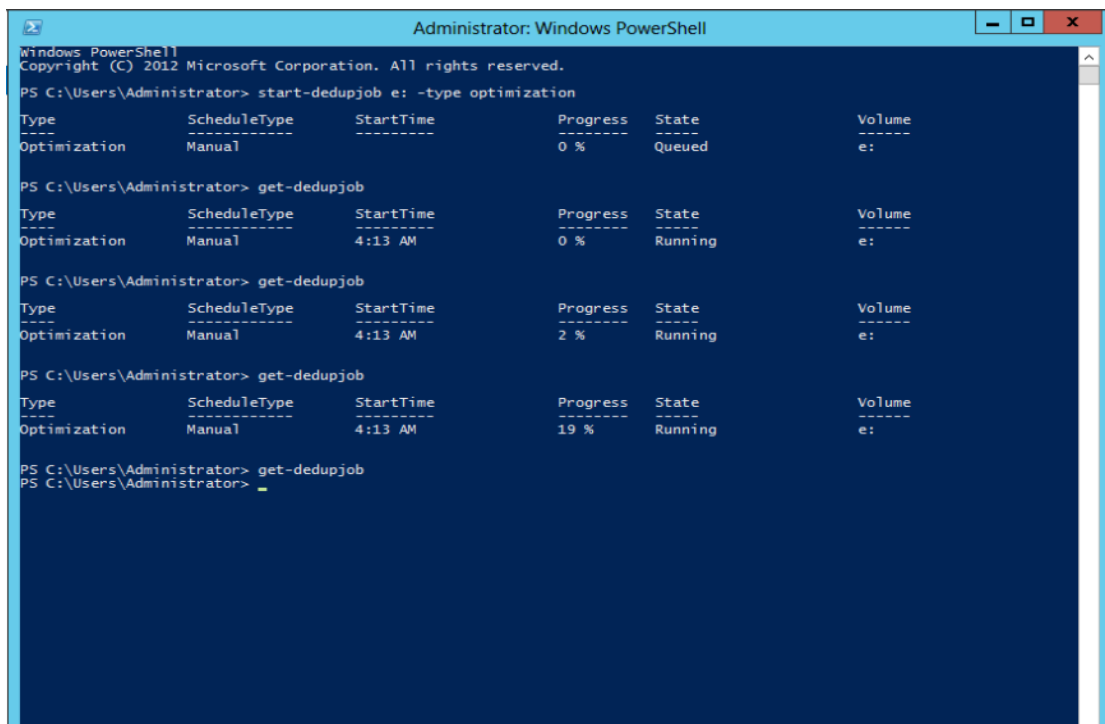


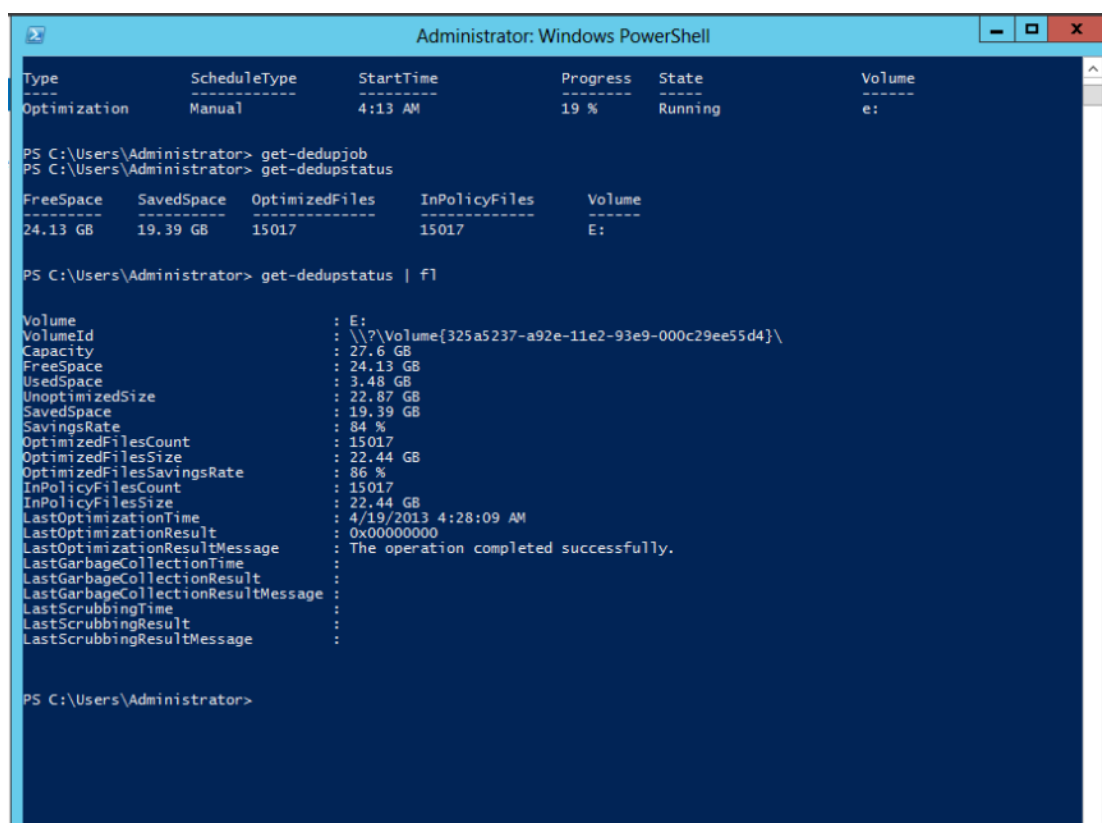
FIGURE 38

The result can be checked by typing the command below:

get-dedupstatus

The result shows that the freespace now is 24.13 GB, saving 19.39 GB, as shown in FIGURE 39. More details can be found by typing:

get-dedupstatus /fl



```
Administrator: Windows PowerShell

Type           ScheduleType   StartTime      Progress      State         Volume
-----
Optimization    Manual         4:13 AM       19 %         Running      e:

PS C:\Users\Administrator> get-dedupjob
PS C:\Users\Administrator> get-dedupstatus

FreeSpace      SavedSpace     OptimizedFiles  InPolicyFiles  Volume
-----
24.13 GB      19.39 GB      15017           15017          E:

PS C:\Users\Administrator> get-dedupstatus | fl

Volume                : E:
VolumeId              : \\?\Volume{325a5237-a92e-11e2-93e9-000c29ee55d4}\
Capacity              : 27.6 GB
FreeSpace             : 24.13 GB
UsedSpace             : 3.48 GB
UnoptimizedSize      : 22.87 GB
SavedSpace            : 19.39 GB
SavingsRate           : 84 %
OptimizedFilesCount   : 15017
OptimizedFilesSize    : 22.44 GB
OptimizedFilesSavingsRate : 86 %
InPolicyFilesCount    : 15017
InPolicyFilesSize     : 22.44 GB
LastOptimizationTime  : 4/19/2013 4:28:09 AM
LastOptimizationResult : 0x00000000
LastOptimizationResultMessage : The operation completed successfully.
LastGarbageCollectionTime :
LastGarbageCollectionResult :
LastGarbageCollectionResultMessage :
LastScrubbingTime     :
LastScrubbingResult   :
LastScrubbingResultMessage :
```

FIGURE 39

The Deduplication Rate and Deduplication Savings can also be found in VOLUMES, as shown in FIGURE 40, the Deduplication Rate is 84% and the Deduplication Savings is 19.4 GB.

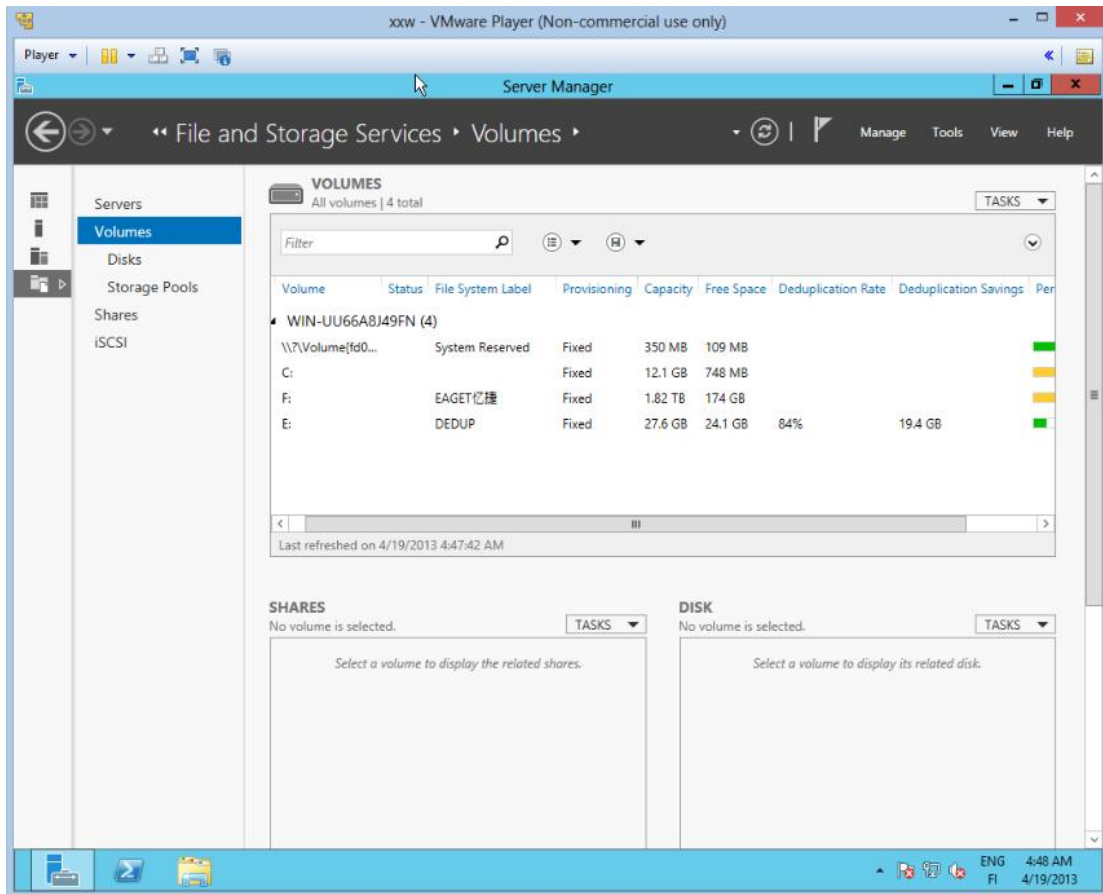


FIGURE 4

6 LINUX

Linux is a free and open source Unix-like system which is originally designed by Linus Benedict Torvalds in 1991. Linux is a leading system that it is able to run on the server and other platform, like mainframes and supercomputers. Linux is widely used in embedded systems ,such as phones, tablet computers and routers, as well.

Following the GNU General Public License, anyone can use all the underlying source code of Linux and modify or redistribute freely. Typically, Linux is packaged into a Linux distribution for the use of personal computers and servers, including Debian (and its derivative versions of Ubuntu, Linux Mint), Fedora (and its related versions of Red Hat Enterprise Linux CentOS) AND openSUSE.

6.1 Introduction of Ubuntu

Ubuntu is based on Debian releases and the GNOME desktop environment. Differing from Debian which publish a new version every two years, it will release a new version every six months. It aims to provide an up-to-date and fairly stable operating system which is mainly consisted of free software for average user. It fully comply with the principle of open source software development, encouraging people to use, improve and spread of open source software. However it does not just mean zero cost, the idea is that people should be free to use all socially useful software. And the free software does not just mean that you do not need to pay for it, it also means that you can use the software the way you prefer: you can download the software in any way you want and modify it with your requirement. Besides, the technical advantage is apparent that others achievement can be your useful reference when your programs are developed, without doing everything by yourselves from the very beginning. ^[36]

Ubuntu is a Unix-like system with Linux core, which is a very popular Linux distribution. Since it has open source, it is totally free for users. Differing from Windows system, almost everything need to be done by commands although Ubuntu itself has a GUI too. ^[29]

6.2 Data Deduplication by ZFS

ZFS is a file system first designed by Sun Microsystems. It is a new lightweight file system with a high storage capacity, conceptual integration of file system and volume management and also a convenient storage pool management system.

6.2.1 Introduction of ZFS

Unlike traditional file system residing on a separate device or a volume management system to use more than one device, ZFS establish it in a virtual device, known as the zpool. Each storage pool includes number of virtual devices. These virtual devices can

be the original disk, a RAID 1 mirror device, or non-standard RAID level multi-disk group. Therefore the file system on zpool could use a total storage capacity of these virtual devices. ^[35]

Storage pools and deduplication are two significantly important features in ZFS ^[30] which we are going to use next.

6.2.2 Deduplication in ZFS

ZFS owns the deduplication since the end of October 2009^[33]. ZFS deduplication can be classified as inline process, as we introduced in previous chapter, it will delete the duplicates and remain the unique data before writing data in to disk. This method can be very space-saving since all the optimization are finished before the storage, however it is highly demanding on RAM capacity, virtual memory thrashing could be the result if physical memory is not sufficient. ^[34]

6.3 Installing

VMware is still going to be used this time and the process of installing is similar to the previous one, so I will drop several steps in this case. The main difference is I am going to use the ISO of 12.10 desktop version this time, as shown in FIGURE 41

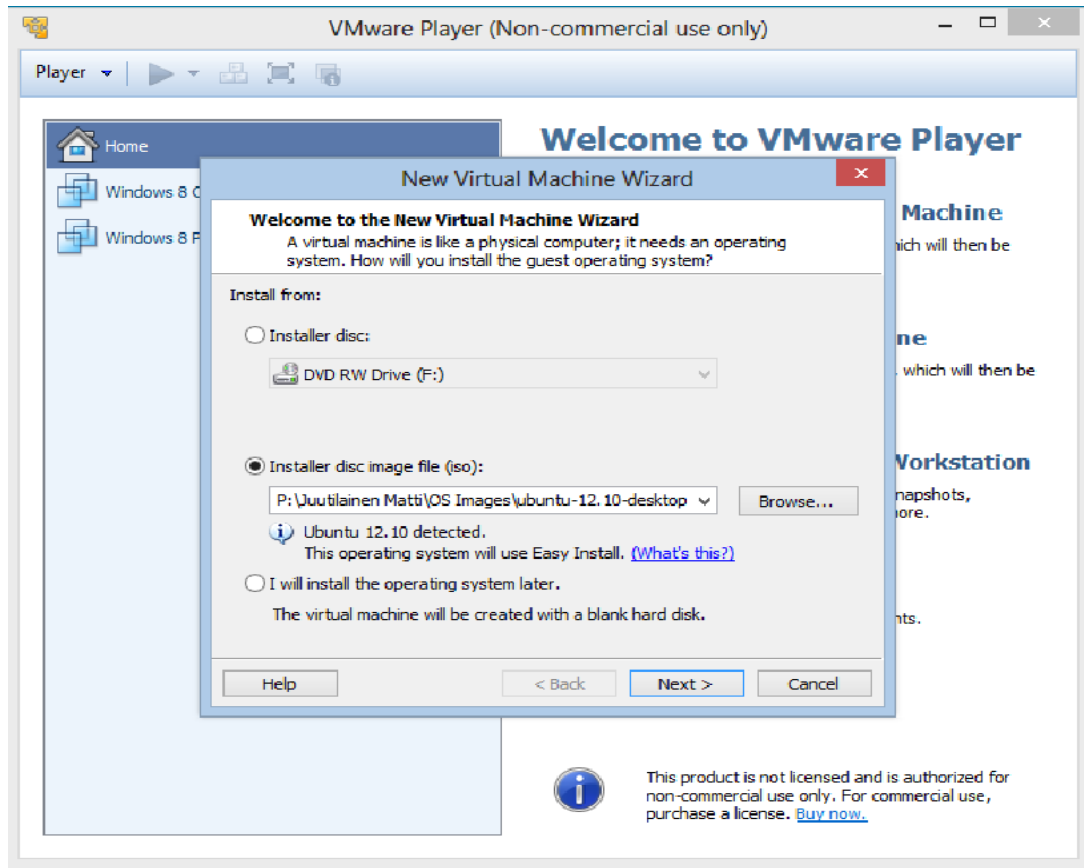


FIGURE 41

It takes about 20 minutes to install this OS, as shown in FIGURE 42



FIGURE 42

6.4 Data Deduplication with ZFS

STEP 1 open the terminal and type the command that you need. Here, we first type:

```
sudo -i
```

Then, terminal will request you to input the password that you set during the configuration. When you input the password correctly you will log in as root which means you can just type the command without typing `sudo` before the command.

STEP 2 Install ZFS. First type the command:

```
apt-get install zfs-fuse
```

Then when asked if you want to continue, press `Y` and enter, and the installing of ZFS will continue till end, as shown in FIGURE 43

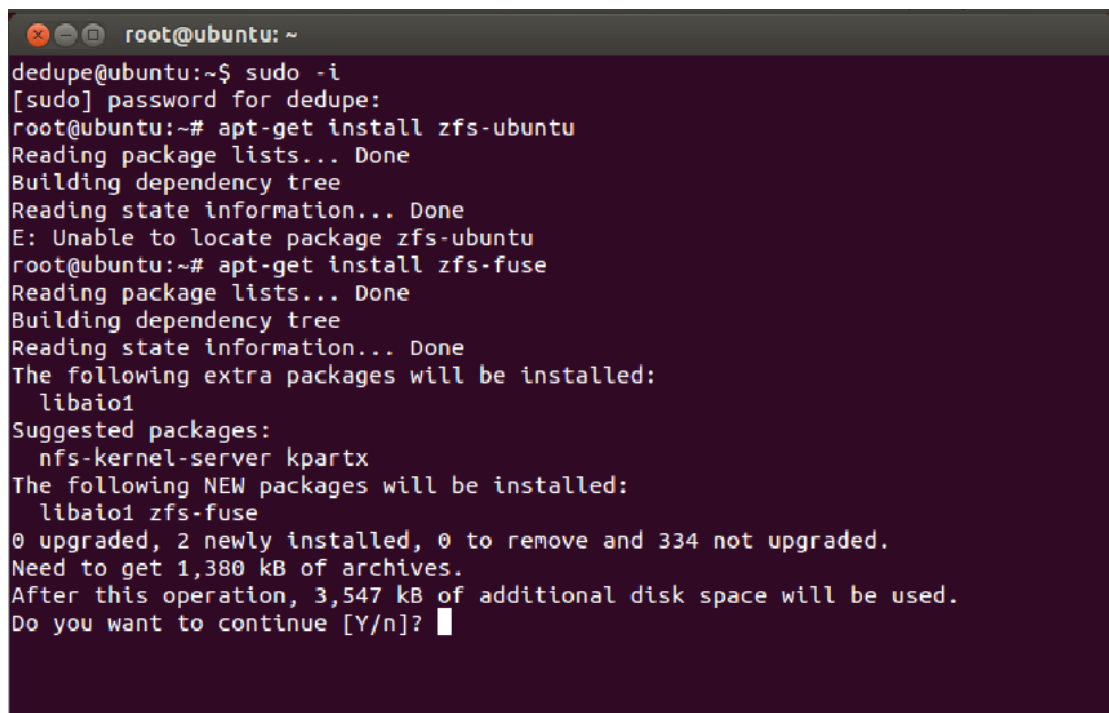
A terminal window screenshot showing the installation of zfs-fuse. The user starts as 'dedupe' and runs 'sudo -i' to become root. Then they run 'apt-get install zfs-ubuntu', which fails with an error 'Unable to locate package zfs-ubuntu'. They then run 'apt-get install zfs-fuse', which succeeds. The terminal output shows the package lists, dependency trees, and disk space requirements. The user is prompted to continue with 'Y/n' and the cursor is on 'Y'.

FIGURE 43

As you can see in FIGURE 44, ZFS has been successfully installed. Next type this command:

```
zpool status
```

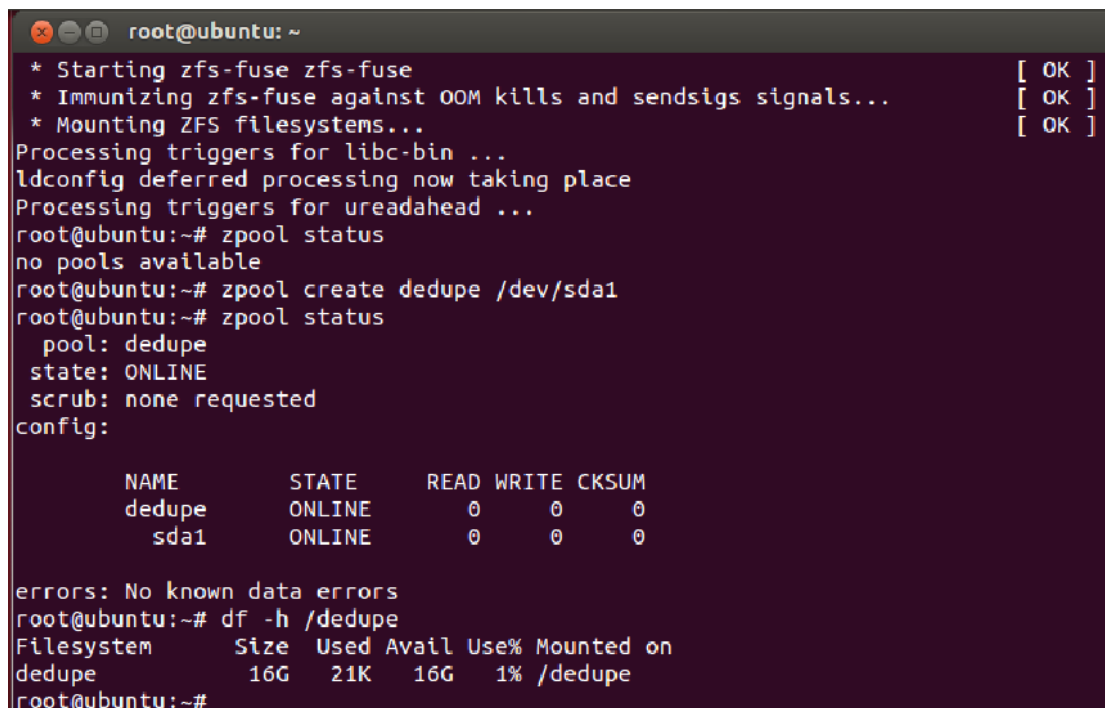
You can see what is shown is no pools available, which means that we need to create one:

```
zpool create dedupe /dev/sda1
```

Now, a pool has been created .type the command below to check the utility of dedupe:

```
df -h /dedupe
```

The size of dedupe is 16 GB.

A terminal window showing the installation of ZFS and the creation of a dedupe pool. The terminal output includes messages about starting zfs-fuse, immunizing against OOM kills, and mounting ZFS filesystems. It shows the 'zpool status' command being run twice: first, it reports 'no pools available', and second, after running 'zpool create dedupe /dev/sda1', it shows the pool 'dedupe' in an 'ONLINE' state. A table displays the pool's configuration, and a 'df -h /dedupe' command shows the pool's size as 16G and its usage as 1%.

```
root@ubuntu: ~
* Starting zfs-fuse zfs-fuse [ OK ]
* Immunizing zfs-fuse against OOM kills and sendsigs signals... [ OK ]
* Mounting ZFS filesystems... [ OK ]
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
Processing triggers for ureadahead ...
root@ubuntu:~# zpool status
no pools available
root@ubuntu:~# zpool create dedupe /dev/sda1
root@ubuntu:~# zpool status
  pool: dedupe
state: ONLINE
scrub: none requested
config:

    NAME      STATE    READ WRITE CKSUM
    dedupe    ONLINE      0     0     0
    sda1      ONLINE      0     0     0

errors: No known data errors
root@ubuntu:~# df -h /dedupe
Filesystem      Size  Used Avail Use% Mounted on
dedupe          16G   21K   16G   1% /dedupe
root@ubuntu:~#
```

FIGURE 44

Type the command below to move to Dedupe:

cd /dedupe

Then, enable the deduplication function of ZFS by typing:

zfs dedup=on dedupe

To check the status of deduplication , type:

zpool list dedupe

Now , we are going to produce some random data to file1 ,typing :

dd if=/dev/urandom of=file1 bs=2M count=100

About 210 MB data has been copied to file1,as shown in FIGURE 45.

Then, we copy all the data in file1 to filecopy by typing:

cp -a file1 filecopy

Then we check the status of deduplication again, finding the allocated space is still about 200 MB rather than 400 MB.

```

root@ubuntu: /dedupe
root@ubuntu:~# cp -a file1 file1copy
root@ubuntu:~# zpool list dedupe
NAME      SIZE  ALLOC   FREE   CAP  DEDUP  HEALTH  ALTROOT
dedupe    15.9G  78K    15.9G   0%  1.00x  ONLINE  -
root@ubuntu:~# cd /dedue
-bash: cd: /dedue: No such file or directory
root@ubuntu:~# cd /dedupe
root@ubuntu:/dedupe# dd if=/dev/urandom of=file1 bs=2M count=500
500+0 records in
500+0 records out
1048576000 bytes (1.0 GB) copied, 136.688 s, 7.7 MB/s
root@ubuntu:/dedupe# cp -a file1 file1copy
cp: reading `file1': Input/output error
cp: failed to extend `file1copy': Input/output error
root@ubuntu:/dedupe# cp -a file1 file2
cp: reading `file1': Input/output error
cp: failed to extend `file2': Input/output error
root@ubuntu:/dedupe# zpool list dedupe
NAME      SIZE  ALLOC   FREE   CAP  DEDUP  HEALTH  ALTROOT
dedupe    15.9G  1004M  14.9G   6%  1.39x  ONLINE  -
root@ubuntu:/dedupe# df -h /dedupe
Filesystem      Size  Used Avail Use% Mounted on
dedupe           17G  1.4G  15G   9% /dedupe
root@ubuntu:/dedupe#

```

FIGURE 45

Performing data deduplication in Ubuntu, compared to Windows Server 2012, is a little bit difficult since almost all the assignments are finished by the commands. But the user could get more intuitive results. Just as we can find out, PowerShell in Windows Server 2012 is just similar to terminal in Linux system, and it is generally better to commands rather than GUI we try to get more information of what we have done so far.

7 CONCLUSION

In this paper we mainly discuss about the definition and classification of deduplication. Among them the most important concept is hash-based deduplication which refers to that the files are first divided into pieces of chunks, and a hash will be generated with each chunk and stored in chunk table with which the whole deduplication system works.

Besides, the basic use of deduplication in Windows Server 2012 and Ubuntu are

covered in the thesis. With GUI, deduplication in Windows Server 2012 is very easy to grasp because you do not need to memorize too many commands. But to process data deduplication successfully, you have to store all the data into the storage disk which requires the disk a high capacity and then it is possible for user to carry out the deduplication. Compared to Windows Server 2012, users have to use many commands to conduct the deduplication, but it is faster in speed because the deduplication is performed before writing the data to disk which saves a lot of time for transmitting data.

Data deduplication is playing a more and more important role in this information society where new data grows in a unusual high speed and we need to get ourselves ready in order to not get lost in it.

BIBLIOGRAPHY

【B1】 mcKnight J, Asaro T, Babineau. Digital archiving: end-user survey and market forecast 2006-2010 [EB?OL].

【2】 Clements A, Ahmad I, Vilayannur M, et al. Decentralized deduplication in SAN cluster file systems[C]// Proc of the USENIX ATC 09. Berkeley, CA: USENIX, 2009: 101-114

【3】 data deduplication technology review
<http://www.computerweekly.com/report/Data-deduplication-technology-review>
[Referred 23.3.2013].

【4】exchange single-instance storage and its effect on stores when moving mailboxes
<http://support.microsoft.com/kb/175481> [Referred 23.3.2013]

【5】 a Microsoft exchange history lesson& other fun factoids
<http://www.sherpasoftware.com/blog/microsoft-exchange-history-lesson/> [Referred 23.3.2013]

【6】 Evolution of Data Centers
<http://data-centers.in/evolution-of-data-centers/> [Referred 23.3.2013]

【7】 deduplicatio internals-source side & target side deduplication
<http://pibytes.wordpress.com/2013/03/09/deduplication-internals-source-side-target-side-deduplication-part-4/> [Referred 23.3.2013]

【8】Paulo Ricardo, Motta Gomes. Distributed Deduplication in a Cloud-based Object Storage System. [Referred 23.3.2013]

【9】 primary dedupe where are you?
<http://storagegaga.wordpress.com/tag/primary-storage-deduplication/> [Referred 23.3.2013]

【10】 Mike Dutch, Understanding data deduplication ratios

【11】 primary VS secondary storage de-deduplication
<http://www.securstore.com/blog/primary-vs-secondary-storage-de-duplication/>[Referred 23.3.2013]

【12】 data compression
<http://www.dspsguide.com/ch27/4.htm> [Referred 23.3.2013]

【13】 Keren Jin, Deduplication on Virtual Machine Disk Images

【14】 Chris Cassano, A comparison study of deduplication implementations with small-scale workloads.

【15】 John Richard Black, Compare-by-Hash

【16】 Qinlu He, Zhanhuai Li, Xiao Zhang, Data Deduplication Techniques

【17】 Dutch T. Meyer, William J. Bolosky, A Study of Practical Deduplication

【18】 File-level vs Block-level vs byte-level deduplication

<http://www.starwindsoftware.com/features/file-level-vs-block-level-vs-byte-level-deduplication>

[Referred 1.4.2013]

【19】 William J. Bolosky, Scott Corbin, David Goebel*, and John R. Douceur, Single Instance Storage in Windows 2000

【20】 Stephen Mkandawire, Improving Backup and Restore Performance for Deduplication-based Cloud Backup Services

【21】 The pros and cons of file-level vs. block-level data deduplication technology

<http://searchdatabackup.techtarget.com/tip/The-pros-and-cons-of-file-level-vs-block-level-data-deduplication-technology> [Referred 1.4.2013]

【22】 Partho Nath, Bhuvan Urgaonkar, Anand Sivasubramaniam, Evaluating the usefulness of content addressable storage for high-performance data intensive applications [C] // Proc of HPDC'08. New York: ACM, 2008;35-44

【23】 hash collision probabilities

<http://preshing.com/20110504/hash-collision-probabilities> [Referred 1.4.2013]

【24】 Heckel C. Philipp, Minimizing remote storage usage and synchronization time using deduplication and multichunking: Syncany as an example

【25】 Zhu Benjamin, Kai Li, Patterson Hugo. Avoiding the disk bottleneck in the Data Domain deduplication file system [C] // Proc of the USENIX FAST'08.

Berkeley, CA: USENIX, 2008: 269-282

【26】 Windows Server 2012 Overview

<http://www.microsoft.com/en-us/server-cloud/windows-server/overview.aspx>

[Referred 20.4.2013]

【27】 Data Deduplicatio Overview

<http://technet.microsoft.com/en-us/library/hh831602.aspx> [Referred 20.4.2013]

【28】 Data Deduplication in Windows Server 2012 explained from A to Z

<http://jeffwouters.nl/index.php/2012/01/disk-deduplication-in-windows-8-explained-from-a-to-z/> [Referred 20.4.2013]

【29】 An Introduction to Ubuntu Linux

<http://www.linuxeum.com/Distros/osIntroToUbuntu.php> [Referred 20.4.2013]

【30】 FreeBSD Handbook

<http://www.freebsd.org/doc/en/books/handbook/filesystems-zfs.html> [Referred 20.4.2013]

【31】 An Efficient Data Deduplication Design with Flash-Memory Based Solid State Drive

【32】 Kruus Erik, Ungureanu Cristian, and Cezary Dubnicki. Bimodal content defined chunking for backup streams

【33】 ZFS Deduplication

https://blogs.oracle.com/bonwick/entry/zfs_dedup [Referred 6.5.2013]

【34】 Why ZFS? : Pools & Deduplication

<http://www.smallnetbuilder.com/nas/nas-features/32079-why-zfs-pools-a-deduplication>
[Referred 6.5.2013]

【35】 Oracle Solaris ZFS Administration Guide

<http://docs.oracle.com/cd/E19253-01/819-5461/zfsover-2/> [Referred 6.5.2013]

【36】 The Ubuntu story

<http://www.ubuntu.com/about/about-ubuntu> [Referred 6.5.2013]