2012 International Conference on Image, Vision and Computing (ICIVC 2012)
IPCSIT vol. 50 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V50.57

A High-efficiency Differential File Recovery Method

BIAN Xia[1], GAO Yuan[2+] and BIAN Yuan[3]

[1]School of Computer Science, Sichuan University, Chengdu，China

[2]School of Software Engineering, Sichuan University, Chengdu，China

[3]School of Management, Yulin College, Yulin，China

Abstract. For the problem of low efficiency and slow speed in file recovery area, a fast differential file recovery method∗ was proposed. The remote backup center keeps the reverse difference set and the latest full backup file,to achieve the recovery file, the reverse differential files are used to reconstruct the file we need by time reversal. Experiment results show that this method significantly reduces the reconstruction time，and improves the efficiency of file recovery evidently in comparison with the traditional differential recovery method.

Keywords: file recovery; reverse difference; file reconstruction

1. Introduction

As information technology is widely used in the communications, financial, military and other fields, computer data has become more and more important for enterprises [1], the security of data is essential to social development, and thus the data disaster recovery technology [2] has become an increasing concern. The main research of disaster recovery technology is to ensure minimal loss of data and make the system up and running in the shortest possible time after a disaster.In daily life, most of the data are stored as files. Therefore, a high-efficiency file recovery method is needed.

2. File Recovery Analysis

The file recovery system consists of a local data center and a remote backup center. When a disaster occurred in the local data center, the service was switched to the remote backup center, then the recovery processes begun.

For the traditional differential recovery method, the remote backup center only saved the earliest full backup file, and the subsequent forward difference files. This effectively saves storage space, However, when recovering files, the earliest full backup file and each forward difference files are used to reconstruct the recovery file one by one, and usually we want to get the latest full backup file, when there are a lot of forward difference files, reconstruction will be carried out repeatedly, a lot of disk operations are needed and the server's performance will inevitably decline. So this paper proposes to store both the forward and reverse difference files [3] in the remote backup center, the latest full backup file was reconstructed [4] by the forward difference files, and when recovering, the latest full backup file and the reverse difference files are used, and if the required file is the latest full backup file, the remote center directly transmit it to the client.

+ Corresponding author. Tel.: +13688321454.
E-mail address: shuoyu1124@163.com.

3. High-efficiency File Recovery Method

3.1. Definition of Terms

1) f_i represents the user backuped file.
2) $F = \{f_1, f_2, \cdots, f_i, \cdots, f_n\}$, represents the backup file set, it is a collection of multiple files or directories.at time t_1 it is denoted by F_1,at time t_2 it is denoted by F_2
3) $\Delta F = F_2 - F_1$, represents the forward difference file set, records the difference between backup file sets at time $t_2$and t_1, here $t_1 < t_2$.
4) $<f_P, f_D>$,represents the forward difference file, and $\langle f_P, f_D \rangle \in \Delta F$ .f_P represents the index file that records the length of the difference data and the numbers of the data blocks that are the same, they are expressed by positive and negative integers respectively. f_D represents the forward difference data file,it is a a stream file.
5) $\Delta F' = F_1 - F_2$, represents the reverse difference file set, records the difference between backup file sets at time t1 and t2, here t1< t2 .
6) $<f'_P, f'_D>$ represents the reverse difference file, and $\langle f'_P, f'_D \rangle \in \Delta F'$ .f'_P represents the reverse index file that records the length of the difference data and the offset of the same data block at the new file, they are expressed by negative and positive integers respectively. f'_D represents the reverse difference data file, it is a stream file.
7) f'_{MT} represents reverse match of the files at adjacent backup time. f'_{MT} records the sub-block size of the old file(f_i), the sub-block number and the offset of the match file at the old file and new file(f'_I) respectively.

3.2. Difference Calculation Principle

Suppose there are two similar files f_1 and f_2, the difference data blocks can be found out and record by calculating the weak 32-bit rolling checksum and 128- bit rolling checksum. Rolling checksum first generates a data block checksum, and then move forward a byte, calculate the next data block checksum. One advantage of this algorithm is that when the checksum of a given data block is known, you can quickly calculate the checksum of the next offset data.

Suppose that a file b, size m, the checksum from byte l to l + k (0 <l <m) has been calculated ,then to get the checksum from byte l+1 to byte l + k +1 will be very easy, the specific steps are as follows.

(1) First calculate the checksum from byte l to byte l+ k:

$$a(l, l+k) = (\sum_{i=l}^{i=l+k} X_i) \bmod M$$

$$b(l, l+k) = (\sum_{i=l}^{l+k} (l+k-i+1) X_i) \bmod M$$

$$s(l, l+k) = a(l, l+k) + 2^{16} b(l, l+k)$$

Where M is a random integer, and its value is generally 2^{16}.

(2)Use the result of （1） to calculate the checksum from byte l+1 to byte l+k+1:

$$a(l+1, l+k+1) = (a(l, l+k) - X_l + X_{l+k+1}) \bmod M$$

$$b(l+1, l+k+1) = (b(l, l+k) - (k+1) X_l + a(l+a, l+k+1)) \bmod M \quad s(l+1, l+k+1) = a(l+1, l+k+1) + 2^{16} b(l+1, l+k+1)$$

3.3. Rreverse Difference File Set Generation

Using the high-efficiency file recovery method, the forward and reverse difference files can be generated with only one comparison scan. The reverse difference files can be used to reconstruct the file we need. There is two steps for the reverse difference file set generation: reverse matching file generation and reverse difference file generation.

3.3.1. Reverse Matching File Generation

f'_{MT} records the sub-block size of the old file(f_i), the block number and the offset of the match file at the old file and new file(f'_I) respectively. The steps are as follows:

Block division: divide the file into a series of data blocks size of k that have no overlaps .and record the the number k in f'_{MT} .Because differences in computational complexity and precision depends largely on the sub-block size, so the value of k is generally dynamic, mainly determined by the file size

Checksum calculation: For each sub-block , calculate the rolling checksum calculation and the MD4[5] checksum, expressed as $h_j\langle R_j, M_j\rangle$, where R_i is the rolling checksum, M_i is the MD4 checksum..

Rolling checksum sorting: hash sort the the rolling checksums and sort the results into a 16bit hash table .And then create a 16 bit index table, each index value points to an entry point.

Scan and search f'_I to generate the forward and reverse difference index files . The steps are as follows:

1) Calculate the rolling checksum, denoted as *r(l,l+k),* for the data block that has an offset *l* and size of *k* in f'_I as well as the hash value, and then search the index table and hash table to find the matched checksum.
2) If the checksum does not has a matched checksum, record the *l*th byte of f'_I in f_D ,and write 1 in f_P, means there is a byte difference, then $l = l + 1$, goto (1).
3) If the checksum has a matched checksum, calculate the MD4 checksum of this data block, denoted as M_i. and compares it with the corresponding M'_I of the data block in f'_i. If $M_i = M'_I$ means that the two data blocks are exactly the same. record the negative block number of f_i in f_P, and record the block number of f_i along with the offset of this data block in f'_i into f'_{MT} ,then go to (1).
4) If $M_i \neq M'_I$ record the *l*th byte of f'_I in f_D , ,and write 1 in f_P, then $l = l + 1$, go to (1).
5) When finishing the search for f'_i, the reverse matching file is generated successfully. For example: The contents of the new file f'_i is x i a b i s n 1, the contents of the old file f_i is 1 b i a n x i s, the block size is 2. After scanning and searching steps, we can find that the contents of the forward difference file f_D is x b n 1, the contents of the forward difference index file f_P is 1 -2 1 -4 1 1,and the contents of the reverse matching file f'_{PT} is 2 2 1 4 4. As shows in Figure 1.

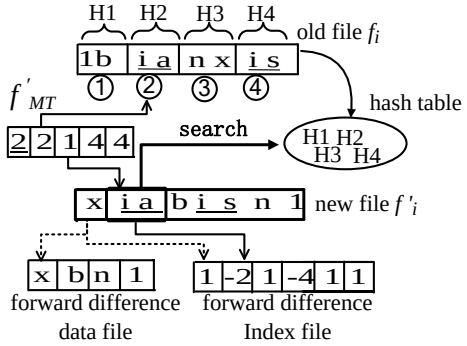

Fig. 1 Procedure of reverse matching file generation

3.3.2. Reverse difference file generation

In the remote backup center, the reverse matching file f'_{MT} is used to generate the reverse difference file, including the reverse difference data file f_d and the the reverse difference index file f_p'. The specific steps are as follows:

(1) Scan the reverse matching file $f'_{MT,}$ the sub-block size *k* of the old file, the matched block number of the old file f_i and the offset of the block in the new file f'_I are stored in $f'_{MT.}$ Read data from the file f'_{MT} to obtain the value of *k* of the old file, and write it to *fp'*.

(2) Scan each sub-block of the old file *fi* from start to finish, if a block number is the same as the block number that has been recorded in the reverse matching file $f'_{MT,}$ then write the corresponding offset value in the new file into *fp'*, otherwise write -1 into *fp'* and write the contents of this block into *fd'*. For example: go on the example of Figure 1,the contents of the reverse matching file f'_{MT} is: 2 2 1 4 4， from this we can read that the sub-block size of the old file is 2,the block number 2 and 4 in the old file matches the data with the offset of 1 and 4 in the new file respectively. As shows in Figure 2:

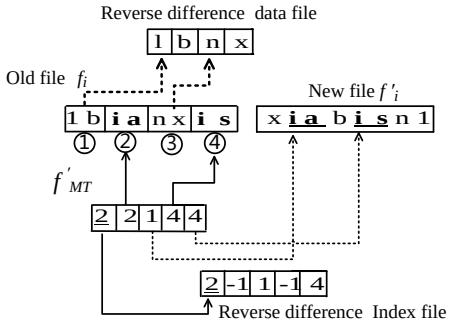

Fig. 2 Reverse difference files generation process

3.4. The Latest Full Backup File Generation

The latest full backup file stored in the remote backup center is generated by the earliest full backup file and each forward difference files with file reconstruction operations. Reconstruction is the process of using old file f_i and the forward difference file to generate the new file f'_i. f_T is a temporary file, the processes of the reconstruction operations are as follows:

1) Fetch data from f_P until to the end of the file, the reconstruction operation is complete, and cover f_i with f_T to get f'_i otherwise, continue to fetch the data x.
2) if x> 0, fetch x bytes of data from the current file pointer and write them to f_T, goto(1).
3) If x <0, fetch the block of number -x, write it in f_T, goto(1).

3.5. File Recovery

File recovery is the process of using the new file f'_I and the reverse differential file $<f'_P, f'_D>$ to construct the previous backup file f_i .f_T is a temporary file, the processes of reverse reconstruction are as follows:

1) Read data from f'_P and get the sub-block size *k* of the old file.
2) Continue to read data from f'_P, if to the end of the file, the reverse reconstruction is complete, and cover f_i with f_T, otherwise fetch the data *x*;
3) if x = -1, fetch *k* bytes of data from the current file pointer of *fd'* and write them in f_T, go to (2);
4) If $x > 0$,put the pointer of f'_I to x, fetch *k* bytes of data, and write them in f_T ,go to (2);

The reconstructed file f_i is the previous full backup file, with the reverse reconstruction we can get any full backup file we need, then transmitt the file to the client can finish the file recovery.

4. Comparative Experiment

We used both the traditional differential recovery method and the method we proposed to recover the backup point to the latest full backup file in the situation of growing point in time, then compared which is time-consuming.

Figure 3 shows a clear and intuitive relationship between the number of backup points in time and the time consumed to recover the backup point to the latest full backup file.

Figure 3 shows that, with the backup point increase in recovery, using this method the time is to a constant, while the traditional method is linear growth. Compared with the traditional recovery method, The method can significantly shorten the time to recover the backup point to the latest full backup file, thus increasing the efficiency of the backup file recovery.

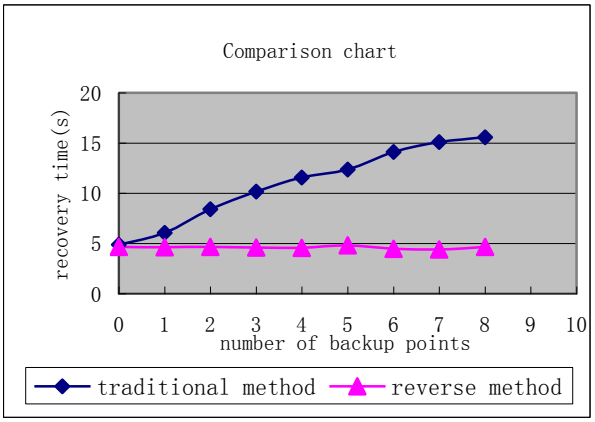

Fig.3 Comparison of file recovery in the latest backup point

5. Conclusion

This paper presents and implements a high-efficiency differential file recovery method based on Internet. With this method, the remote backup center keeps both the forward and reverse difference files, the latest full backup file was reconstructed by the forward difference files, while the reverse difference files are used for recovery. At last, we compared the method with the traditional recovery method, result showed that the method can significantly shorten the time to restore the backup point to the latest full backup file and improves the efficiency of file recovery evidently.

6. References

[1] Li Tao. Introduction to Network Security [M]. Beijing: Electronic Industry Press ,2004.

[2] Soubir. Acharya, Susan G. Friedman, Backup strategies for networked storage, http://www. infostor.com/index/articles/display/126595/s-articles/s-infostor/s-volume-5/s-issue-11/s-features/s-backup- strategies-for-networked-storage.html, 2001

[3] Andrew Tridgell. Efficient algorithms for sorting and synchronization [D]. Canberra : The Australian National University, 1999.

[4] Andrew Tridgell, Paul Mackerras. The rsync algorithm[R]. Canberra : The Australian National University, 1996.

[5] RFC 1320, The MD4 Message-Digest Algorithm, http://www.ietf.org/rfc/rfc1320.txt,1992