2012 2nd International Conference on Industrial Technology and Management (ICITM 2012)
IPCSIT vol. 49 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V49.52

A Parallel Transmission Remote Backup System

Che Yu[+]

College of Computer Science, Sichuan University, Chengdu, China

Abstract. A parallel transmission remote backup system is presented in this thesis. This system could offer real-time monitoring for data of user's disk partition, and transmit the changed data to remote backup server by parallel approach. When data loss occurs, this system could recover the backup data to the local disk partition. This system also offer a write order constraints and data consistence check to makesure the data is correct.

Keywords: Backup; parallel; recover; consistence

1. Introduction

The attention of the data security is growing in today's informatization highly developed society. Data loss or corruption will bring inestimable loss in many walks of life. The disaster recovery system become more and more common.

The principle of most remote disaster recovery system is create the backup of target disk on a removable disk or networked location, data in the target disk and the backup disk become consistent by resynchronization.

This paper propose a parallel transmission remote backup system[1]. Different from other backup system is this system uses a parallel transmission and use write order constraints to ensure the data is correct. This system also offer a data consistency check to make sure the backup data is useful.

2. System Module

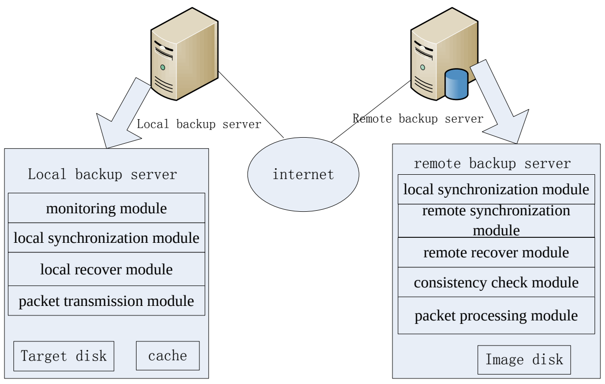

Fig. 1: System module structure

The system is divided into monitoring module, synchronization module, recovery module, packet transmission module, packet processing module, consistency check module.

[+] Corresponding author.
E-mail address: lance.cheyu@gmail.com.

Synchronization module divided into local synchronization module and remote synchronization module[2]. This module copy target disk data into image disk. Ensure the target disk consistent with the image disk when the task start running.

Recovery module divided into local recovery module and remote recovery module. When data loss occurs this module lock the local disk and copy image disk data into target disk. Ensure disk data recovery to the state before the disaster.

Monitoring module intercepts disk write operation by volume filter driver[3], storage the corresponding data into the cache on the local server.

Packet transmission module transfer data packet to remote server, use the parallel transmission to improve the speed.

Packet processing module receive the data packet from the local server. Read information from the packet and write it into the image. Use write order constraints to response data association and data dependent.

3. Task Process

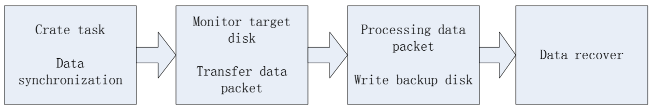

Fig. 2: Task running process

This backup system create task for every target disk, synchronize、monitoring and recover the disk by operate the task.

User create a task for target disk on the operate platform, then the system create a image disk in the remote server storage medium. Local synchronization module create a buffer memory for the task to restore the task information. [4]Remote synchronization module create a image disk in the remote server, then make the target disk and image disk data consistence by synchronization.

When the synchronization is complete, the monitoring module to begin the monitoring of I/O operate at the corresponding disk partition, and transfer the monitoring data to remote server by internet.

Data packet processing module read data packet’s information and write it into the image, make sure the image disk and target disk be data consistence.

When data loss occurs, user begin disaster recover operate. Recover module locking target disk. The data in the image disk will be copied to the target disk, to make disk data recovery to the state before the disaster.

4. Data Synchronism

Beginning in the monitoring module to monitor the corresponding disk partition, you need to create a storage medium in the mirror, and through data synchronization to mirror data corresponding to the same disk partition. The synchronization include complete synchronization and diversity synchronization.

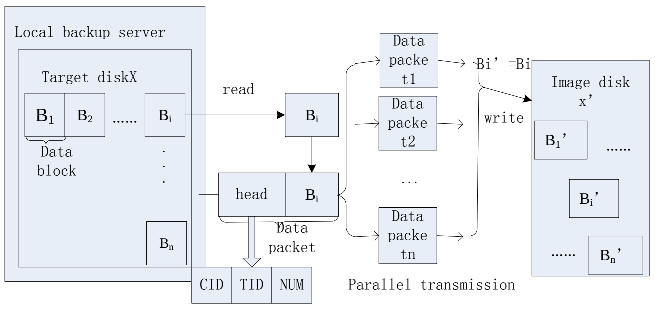

Fig. 3: Synchronism process

In the beginning of synchronization, local synchronization module partition the target disk into numerous data block[5]. Suppose the target disk named X, the aggregate Ld include disk X’s data, divided Ld into N-data block $B_1 B_2 \cdots B_n$, and content the following conditions:

$$Ld = \bigcup_{i=1}^{n} B_i, \bigcap_{i=1}^{n} B_i = \phi$$
.

Meanwhile, create a data mirroring in the storage image for the task. Suppose the image disk named X', the aggregate Rd include the data of disk X', divided Rd into M-data block $B'_1\ B'_2 \cdots B'_m$, and content the following conditions:

$$Rd = \bigcup_{i=1}^{m} B'_i, \bigcap_{i=1}^{m} B_i' = \phi \quad , \quad m = n\ 。$$

Local synchronization module read data block Bi, add a header file HEAD, HEAD is defined as: HEAD={CID，TID，NUM}. The CID is the socket file descriptor of the connect the task which the data block corresponding belongs. The TID is the task id of the task which the data block corresponding. The NUM is the number of the data block, NUM=i. Sent the data block with header file to the remote server one by one. Remote synchronization module read the header file first, find the image disk location by CID and TID, and used NUM to find the corresponding data block in the image disk B_i' which content NUM=i, copy the data in the B_i into B_i'. When all the data blocks are sent to the remote server and be processed, the data in the image disk and the target disk be consistency.

The diversity synchronization can be used to improve the speed if the task isn’t first run. The diversity synchronization got the same data copying process with the complete synchronization. The difference is the diversity synthronization calculated digest value for all data block before data copy. Define a structure $S(B_i)$ to store the data block Bi’s digest value. Local synchronization module calculate all the data block both in the target disk and image disk. Obtain two queue store the digest value $S(B_i)$ from target disk and $S(B_i')$ image disk. The module compare the value of $S(B_i)$ and the value of $S(B_i')$ with the same number, for example $S(B_5)$ and $S(B_5')$. If the value of $S(B_i)$ and the value of $S(B_i')$ are equal, ignore the data block Bi. In the opposite case the module transfer the block B_i and copy the data of B_i into B_i', just like the complete synchronization.

5. Write Order Constraints

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper.

This system use parallel transmission to enhance transfer speed. The packet order which packet processing module received isn’t the correct order.

Applications maintain the consistency of their data by keeping track of modifications of their data sets in special places.

In case of a system crash, the information in these journals is sufficient to bring the data set into a consistent state again. The application must ensure that the write operation to update the journal is finished before the update of the associated data set takes place.

The applications use the fsync or the fdatasync[6] system calls to impose write order constraints. By using these calls the applications are delayed until all pending write requests have been processed.

If the primary node fails, the system have to restart from the current state of the secondary node's disk.

For example, the system don’t need a defined order of the write operations of data blocks B_0, B_1, B_2, B_3 and B_4, since they were issued before any of these IO requests had been down. But it is possible that the application issued the write request for block B_5 because the write operations of B_0 and B_3 were finished. Only the application knows if this is a real write-after-write dependency, therefore we must not violate a single possible dependency.

This system approach is to keep a history of all recently sent write requests. As soon as a single block of these finishes IO, a write barrier must be issued, and the history is cleared.

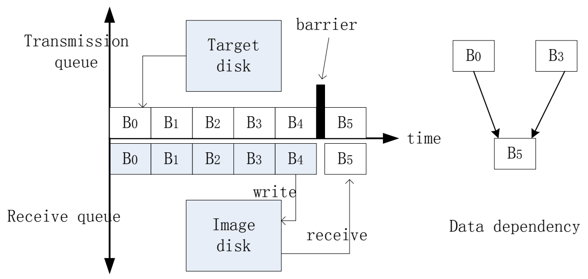

Fig. 4: Write order constraints

In the above example the blocks B_0, B_1, B_2, B_3, and B_4 are added to the history set. With the completion of B_3's write operation the history set gets cleared and a write barrier is issued after block B_4.

The secondary node can now receive blocks and commit them to its local IO subsystem immediately. If it receives a write barrier, it waits until all its pending write requests are finished before it processes any further requests.

6. Data Consistency Checking

After the synchronization has been completed, the consistency checking module checking the data between target disk and image disk. The definition of data consistency checking as follows:

Target data collection P={a,b,c}, backup data collection P'={a',b',c'}. Status of target data collection P at time T_i is $P_i=(a_i,b_i,c_i)$. Status of backup data collection P' at time T_i' is $P_i'=(a_i',b_i',c_i')$.

If $\forall T_i = T_i' \rightarrow P_i = P_i'$, means consistency checking pass, Or $\forall T_i = T_i' \rightarrow P_i \neq P_i'$ means data in the targer disk and image disk isn't consistence.

6.1. Algorithm description

Set current monitoring disk data collection is D, remote backup disk data collection is D'. Therefore D' is a image for D and $D \rightarrow D'$ is a bijective mapping. Aggregate $D_1,D_2,\cdots D_n$ is subaggragate of D, $D_1',D_2',\cdots D_n'$ is subaggragate of D', and contentment follow conditions:

$$\bigcup_{i=1}^{n} Di = D, \bigcap_{i=1}^{n} Di = \phi, \bigcup_{i=1}^{n} Di' = D', \bigcap_{i=1}^{n} Di' = \phi .$$
$$\forall D = D' \Rightarrow D_1 = D_1', D_2 = D_2', \cdots, D_n = D_n' .$$

Define function f(x) to find a summary of the value of the aggregate and get following expression:

$$\forall D = D' \Rightarrow f(D_1) = f(D_1), \cdots, f(D_n) = f(D'_n) .$$

The formulation of the consistency check is:

Branch A:

$\forall T_i = T_j', \forall D_j \in D, D_j' \in D$,

$\forall i = j \Rightarrow f(D_i) = f(D_j') \Leftrightarrow D = D'$ means the target data as same as the image data, the data consistency check be successful.

Branch B:

$\forall T_i = T_j', \exists D_i \in D, D_i' \in D$,

$\exists i = j \Rightarrow f(D_i) \neq f(D_i') \Leftrightarrow D \neq D'$ means the target data and the image data isn't consistence.

For backup system based on internet, because of net transfer, the change of data collection at time point T will reflect to the backup data at the $T_i'=T_i+\Delta T(\Delta T>0)$ time. The fomulation of data consistency check on the internet is:

Branch A:

$\forall T_i = T_j' - \Delta T, \forall D_j \in D, D_j' \in D$,

$\forall i = j \Rightarrow f(D_i) = f(D_j') \Leftrightarrow D = D'$ means the target data as same as the image data.

Branch B:

$\forall T_i = T_j' - \Delta T, \exists D_i \in D, D_i' \in D$,

$\exists i = j \Rightarrow f(D_i) \neq f(D_i') \Leftrightarrow D \neq D'$ means the target data and the image data isn’t consistence.

6.2. Consistency check implement

All the data consistency check include volume copy service， packet transmission module and consistency check module, it’s follows the process:

Begin consistency check module, require packet transmission module stop transfer data.

Consistency check module make sure the data dispatch is stopped, inform the volume copy service create a snapshot[7].

The cache get the time of create snapshot, send the data which the store time before create snapshot.

Data consistency check module begin consistency check between local disk and remote disk.

The check complete, inform the volume copy service destroy the snapshot and note packet transmission module begin transfer data packet.

Data consistency check process like the figure five, Practical application, to achieve the following:

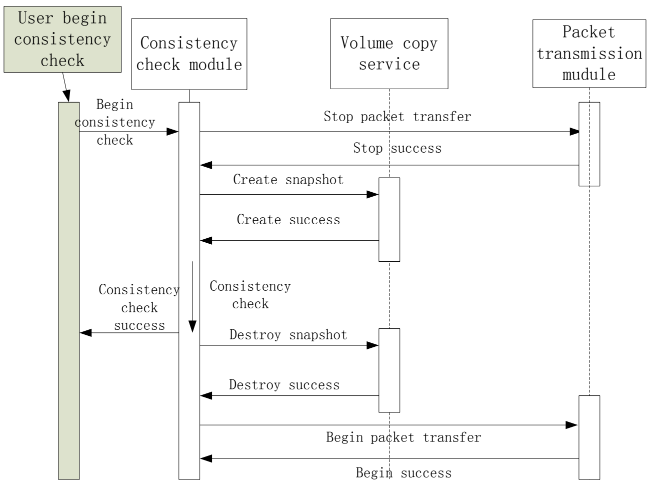

Fig. 5: Data consistency check

Arranged raw data collection and backup data collection into a progression, accordance with the data offset at the volume.

Make the data both in the two aggregate to a bijective mapping accordance with the data offset on the disk.

Divided each aggregate order data into 512 Byte, constitute the aggregate $D_1, D_2, \cdots D_n$ which is subaggragate of D, $D_1', D_2', \cdots D_n'$ which is subaggragate of D'.

Use function f(x) to calculate each aggregate summary value. Compare summary value obtain a result of data consistency check.

7. Conclusions

This paper introduces a parallel transmission remote backup system. The system can backup local disk data in remote server. When data loss occurs this system can recover data from remote server immediately. This system used parallel transmission to enhance synchronization speed, and make sure the data between local disk and remote disk is consistence.

8. References

[1] LAWLER C M, SZYGENDA S A, THORNTON M A. Techniques for disaster tolerant information technology systems. Proceedings of the 1st Annual 2007 IEEE Systems Conference[C]. Honolulu, HI, United States, 2007. 333-338.

[2] Lloyd S J, Joan P, Jian L, “An Economic and Efficient Solution for Real-time Online Remote Information

Backup,”. Journal of Database Management, 2003, 14(3): 56-73.

[3] M Ji, A Veitch, J Wilkes. Seneca, “Remote Mirroring Done Write”. Proceedings of the 2003 USENIX Technical Conference. San Antonio, TX, USA, June, 2003:254-258.

[4] R Cegiela. “Selecting technology for disaster recovery.” Proc of the International Conference on Dependability of Computer Systems 2006:160-167.

[5] Peter M. Chen , Edward K. Lee , Garth A. Gibson , Randy H. Katz , David A. Patterson, “RAID: high-performance, reliable secondary storage,” ACM Computing Surveys (CSUR), v.26 n.2, p.145-185, June 1994 .

[6] Andrew Tridgell,”The rsync algorithm[R]”.Canberra,The Australian National University,1996.

[7] Shaikh F，Shivani Z．”Snapshot service interface (ssi),”a generic snapshot assisted backup framework for linux, international conference on digital information management, Bangalore, India, December 6-8,2006.USA:EEE Computer Society Press,2006:228.