A Method of Deduplication for Data Remote Backup

Jingyu Liu, Yu-An Tan, Yuanzhang Li, Xuelan Zhang, Zexiang Zhou

▶ To cite this version:

Jingyu Liu, Yu-An Tan, Yuanzhang Li, Xuelan Zhang, Zexiang Zhou. A Method of Deduplication for Data Remote Backup. 4th Conference on Computer and Computing Technologies in Agriculture (CCTA), Oct 2010, Nanchang, China. pp.68-75, 10.1007/978-3-642-18333-1_10 . hal-01559563

HAL Id: hal-01559563
https://inria.hal.science/hal-01559563

Submitted on 10 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Method of Deduplication for Data Remote Backup

Jingyu Liu[1,2], Yu-an Tan[1], Yuanzhang Li[1], Xuelan Zhang[1], Zexiang Zhou[3]

[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081, P. R. China
[2]School of Computer Science and Engineering, Hebei University of Technology, Tianjin, 300010, P. R. China
[3]Toyou Feiji Electronics CO., LTD, Beijing, 100081, P. R. China
Liujy01@gmail.com, victortan@yeah.net

Abstract.

With the ever-growing amount of data, the network bandwidth requirement has increased greatly in Remote Backup System. The network bandwidth became the bottleneck of the whole system. The paper describes the Remote Backup System using Hash to identify and avoid sending duplicate data blocks between the Primary Node and the Secondary Node, thereby, to reduce the data replication network bandwidth, decrease overhead and improve network efficiency. On both nodes, some extra storage spaces (the Hash Repositories) besides data disks are used to record the Hash for each data block on data disks. We extend the data replication protocol between the Primary Node and the Secondary Node. When the data, whose Hash exists in the Hash Repository, is duplication, the block address is transferred instead of the data, and that reduces network bandwidth requirement, saves synchronization time, and improves network efficiency.

Keywords: Disaster Recovery, Deduplication, Hash, Duplicate Data

1. INTRODUCTION

Today, the ever-growing volume and value of digital information have raised a critical and mounting demand for long-term data protection through large-scale and high-performance backup and archiving systems. The amount of data requiring protection continues to grow at approximately 60% per year[1]. The massive storage requirement for data protection has presented a serious problem for data centers. Typically, data centers perform weekly full backups for weeks to months. Local hardware replication techniques can mask a significant number of failures and increase data availability. For example, RAID can protect against single disk-failure. Furthermore, certain raid levels even survive multiple simultaneous failure[2],[3],[4]. However, local hardware replication techniques are inadequate for extensive failures or disasters, which may be caused by environmental hazards (power outage, earthquake, and fire), malicious acts or operator errors. To ensure continuous operation even in the presence of such failures, the secondary node (a backup copy of the primary node) is often maintained up-to-date at a remote geographical location and administered separately. When disaster strikes at the primary node, the secondary node takes over transaction processing. The geographic separation of the two copies reduces the likelihood of the backup also being affected by the disaster. Disaster Recovery is such technique.

Data Disaster Recovery is an important measurement to ensure the integrity and availability of computer systems. With the remote replication technology, an offsite independent backup for the local data is stored via the network[5]. When the local node is damaged, the data can be recovered from a remote system immediately[4],[6].

At first, all data blocks on the disk of the local source server (the Primary Node) are duplicated to the remote target server (the Secondary Node) to complete the initial data synchronization. From then on, the data in the Primary Node changed is duplicated to the Secondary Node synchronously or asynchronously via the network[7]. If the data is transferred offsite over a wide area network, the network bandwidth requirement can be enormous[8].

The Primary Node and the Secondary Node are usually deployed separately in two buildings that one is very far apart from the other, or even two cities. There are two ways to transfer the data packet between the nodes, one is then common IP network, the other is the Fibre Channel[9]. Because the private network is expensive, data replication between the Primary Node and the Secondary Node usually uses the common IP network[10]. When the updates are frequent, and the amount of data gets massive, the performance degrades and the backup data maybe lost because of the low network bandwidth and the latency.

Some data blocks on the disk are duplication, for example, a file on disk may have multiple copies or different versions, and the great majority is same[11],[12]. In the data disaster recovery system, all the data blocks of the file need to be transferred to the Secondary Node when the Primary Node creates a duplicate of the file or updates it. However, the

Secondary Node already contains most sections of the data blocks, and the data blocks transferred through the network is duplication of the section of the Secondary Node[13],[14],[15].

The paper describes the Remote Data Disaster Recovery System using Hash to identify and reduce the amount of data transmitted over the network, and at the same time, the technique assures the reliability and availability of the data, and reduces network bandwidth requirement.

The rest of this paper is organized as follows. Section 2 describes the architecture of Disaster Recovery System we used, highlighting the features important for performance. Section 3 discusses implementation of the deduplication method. Section 4 discussed the design and performance evaluation of the solution. Section 5 summarizes and describes future work.

2. ARCHITECTURE

The existing Disaster Recovery System duplicates the data between the Primary Node and the Secondary Node to maintain data consistency of the two nodes through the IP network. An extra storage space is used as a Hash Repository to record the Hash of each data block on the disk. The Hash Repositories of the Primary Node and the Secondary Node are keep consistency, and both of them update synchronously with the data blocks on the disks, as shown in Fig. 1. Generally, when the Primary Node receives the write request, it writes through the local disk immediately and sends the data packets to the Secondary Node. The Secondary node receives the data packets and writes through the disk. After it completes the event, it sends back ACK to the Primary Node, then the event is completed, as shown in Fig. 2[10].

Typically, the size of the data block is 4KB (4096 Bytes), and the Hash is 16 Bytes (128 bits) that is calculated according to MD5. The Hash Repository storages the Hash of each data block in sequence. Each data block takes 16 Bytes, 16/4096=1/256, so the storage space that Hash Repository takes is 1/256 of the storage space that data blocks take.

The architecture of the Hash Repository is shown in Fig. 3.

After the Primary Node receives write request to a data block (the Destination Block), it writes the data to the Destination Block, and calculates the Hash of data block to match with the Hash Repository.

If they do not match, the Primary Node transfer the data block to the Secondary Node and the Secondary Node writes it to the disk.

On the other hand, if they match, it means that the disk of the Primary Node has the same data. This block (the Source Block) is duplication data. It has been delivered to the Secondary Node during the previous initialization or data replication, and it also means that the Secondary Node's disk already contains data of the source block. In this case, the Primary Node needs to transfer the Source Block Address and the Destination Block Address to the Secondary Node Only. Then, the Secondary Node reads the data from the Source Block Address of its local disk and writes it to the Source Block Address.

When the transmission succeeds, both the Primary Node and the Secondary Node update their Hash Repositories.

3. IMPLEMENTATION

The Primary Node receives a write request to write the data A to the destination block PD_B. Correspondingly, the data A should also be written to the destination block SD_B(PD_B=SD_B) of the Secondary Node, as shown in Fig. 4. The Primary Node does as follows: 1) the Primary Node writes data A to the destination block PD_B; 2) calculates the

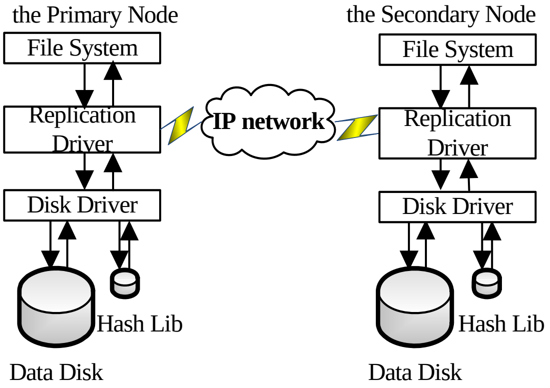


Figure 1. System Architecture

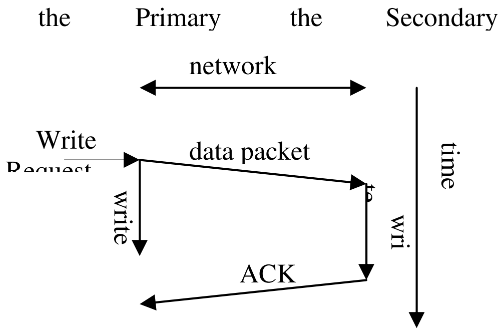


Figure 2. Write Simultaneously

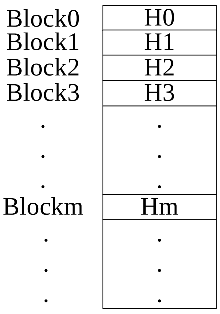


Figure 3. Hash Repository

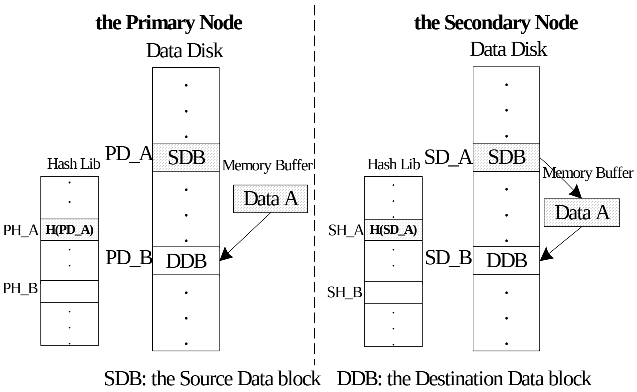


Figure 4. Example of Replication

Hash of the data A; 3) matches the Hash Repository; if it matches with the value in SH_A in the Hash Repository where the Hash of the data block PD_A is placed, it means that the date in PD_A is the same as data A. it is the duplication data. Similarly, the duplication data exists in the Secondary Node also. We suppose that the address is SD_A(SD_A=PD_A). Therefore, when the two nodes synchronize, the data A need not to be transferred. 4) only transfers the source address (PD_A) and the destination address (PD_B) to the Secondary Node; 5) the Secondary Node gets the network package and extracts the addresses from it; 6) reads the data from the source address; 7) writes the data to the destination address; lastly, 8) updates the Hash Repository.

For example: in the existing Disaster Recovery Systems, the file F which size is 8MB (8192KB) is replicated from the A to B. In the 64-bit addressing file system, each data block size is 4KB. The address of each block consists of 8 Bytes (64bit) component, so the file F contains a total of 8192KB/4KB = 2048 data blocks. A total amount of data of synchronization between the Primary Node and the Secondary Node is all the data blocks and their destination addresses, 2048×(4KB+8B) = 8208KB. With the method this paper provides, only the source address, the destination address and the identification information (the size is 1B/block) are transferred because the file F already exists in the Secondary Node. A total amount of data to be transferred is 2048×(8B×2+1B)=34KB, so the data to be transferred is $34KB/8208KB \approx 1/241 \approx 0.415\%$ of the former, and that significantly reduces the network bandwidth requirement for data transmission overhead.

When the Primary Node is malfunction, the Secondary Node can start the remote service system to take over the Primary Node service. Before the Primary Node restored, the Secondary Node's change is written to the disk and the Hash Repository updates at the same time, but this update cannot be synchronized to the Primary Node until it restores. Similarly, When the Secondary Node is malfunction, the Primary Node's update cannot be synchronized to the Secondary Node. Date resynchronize-tion must be performed after the Secondary Node is restored.

Comparing Hash Repositories of the two nodes, we can get the collection of the changed data. The normal node sends these data blocks to the node which used to be malfunction to maintain the consistency between two nodes. During the transfer process, we can use the deduplication technology also.

Each data block size is 4KB, and the Hash size is 16 Bytes, so the Hash Repository size is 1/256 of the data disk size.

The Address of data block's Hash in the Hash Repository can be calculated with the following formula:

Hash Address=Block Address×16

Similarly, find the Address of Hash in the Hash Repository, the block address can be calculated with the following formula:

Block Address= Hash Address/16

When the Primary Node receives write request, it does as follows:

A. For the Primary Node

1) Write the data to the disk.
2) For all data block needed to be written, perform the following steps 3) to 5).
3) Calculate the Hash of each data block.
4) Match the Hash in the Hash Repository.

- If it does not match, the Primary Node constructs the network packet, the structure is shown in Fig. 5(a), and transfers it to the Secondary Node. The "ID" in the package is "0" which means the package includes data and the destination address.
- If it matches, the Primary Node calculates the source address with the formula above, constructs the network package, the structure shown as Fig. 5(b), and transfers it to the Secondary Node. The "ID" in the package is "1" which means the package includes the source address and the destination address.

5) Update the Hash Repository.

B. For the Secondary Node

1) Receive the network packet from the Primary Node.
2) According to the ID in the network, implement as follows:

- if ID=0, extract the destination block address and the data block from the network package, and write them to the data disk.
- if ID=1, extract both the destination block address and the source block address from the network package. Read the source data block from the source block address, and write them to the destination block address.

3) Calculate the Hash of data blocks.
4) Update the Hash of the destination data block in the Hash Repository.

4. EVALUATION

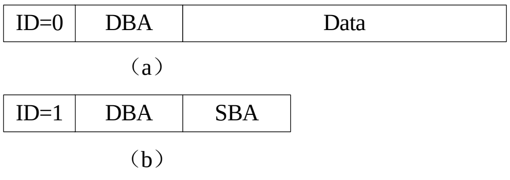
To evaluate the performance of our solution, we build an experimental system under Linux and compare it with the existing Disaster Recovery System of our lab. To compare fairly, we try our best to create similar experiment environment. In the experiment, there are two machines with the same configurations. We divided the four computers into two groups. One Group is for the existing Disaster Recovery system, the other is for the system with our solution. Each group has two computers. One is the Primary Node and the other is the Secondary Node. The machine's CPU is Pentium(R) Dual-Core. We deployed 2.0GB RAM in the machine. The disk of the machine is ST3500418AS (500GB) and NIC is Atheros AR8132. The OS we used is Fedora 10 (Linux Kernel 2.6.27). The two nodes are connected with Fast Ethernet.

In the experiment, we designed two projects: one is that the transmitted data is divided into blocks which sizes are 4KB, the other is that the transmitted data is divide into blocks which sizes are 2KB. In these experiments, we employ a software packet to simulate to normal operation and record the amount of transmitted data.

The results of the experiment are shown in Fig 6. The figure shows that amount of transmitted in the new system is only 22.32% maximum and 12.31% minimum of the former system while divided the date into 4K blocks and it is reach approximately 19% over a period of time (about 20 days). It is only 20.15% maximum and 12.30% minimum of the former system while divided the date into 2K blocks, and it is reach approximately 15% over a period of time. That shows the smaller size of the block can get the higher performance of deduplication. However, there is a question: the smaller size of the block means the greater workload of CPU. It lowered system's performance.

5. CONCLUSION

According to the data replication protocol which is extended between the Primary Node and the Secondary Node, When the Primary Node receives a write request to the Destination Block, the Primary Node identifies if it is the duplicate data block according to the Hash. While the data block to be written is the duplication data block, it will not be the data block to be transferred to the Secondary Node but the block addresses which includes the Source Block Address and the Destination Block Address. The Secondary Node reads the data from the Source Block address of it's local disk and writes to the Destination Block Address. Therefore, when the data is duplication, the Block address is transferred



DBA: the Destination Block Address
SBA: the Source Block Address

Figure 5. Network Package Architecture

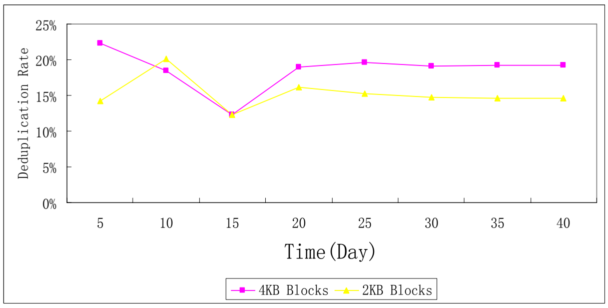


Figure 6. the Rate of Deduplication

instead of the data block, and that reduces network bandwidth requirement, saves synchronization time, and improves network efficiency.

To judge the duplication data makes the CPU's workload increased and this may make the CPU the bottleneck of the system. Future work includes designing a new method to reduce the CPU's workload to improve the system's performance.

6. REFERENCES

1. Yang T, Jiang H, Feng D, et al. "DEBAR: A Scalable High-Performance De-duplication Storage System for Backup and Archiving", [J], CSE Technical reports, 58, 2009.
2. H. Garcia-Molina, H. Halim, R. P. King, and C. A. Polyzois, "Management of a remote backup copy for disaster recovery", ACM Transactions on Database Systems, vol. 16, 338-368, 1991.
3. Polyzois C A, Molina H G. "Evaluation of remote backup algorithms for transaction-processing systems", [J], ACM Transactions on Database Systems (TODS), 19(3): 423-449,1994.
4. Lars Ellenberg, "DRBD 8.0.x and beyond Shared-Disk semantics on a Shared-Nothing Cluster", http://www.drbd.org, 2007.
5. Ao Li, Shu Jiwu, Li Mingqiang, "Data Deduplication Techniques", [J], Journal of Software, Vol. 21, No 5, 916-929, May, 2010.
6. Philipp Reisner, "DRBD – Distributed Replicated Block Device", http://www.drbd.org, Augest. 2002.
7. Patterson R H, Manley S, Federwisch M, et al. "SnapMirror: file-system-based asynchronous mirroring for disaster recovery", [C], USENIX Association, 2002.
8. Zhu B, Li K, Patterson H. "Avoiding the disk bottleneck in the Data Domain deduplication file system". In Proceeding of the 6th USENIX Conference File and Storage Technologies, pages 1-14, California, USA, Feb. 2008.
9. Tan Y A, Jin J, Cao Y D, et al. "A high-throughput fibre channel data communication service", [C], Institute of Electrical and Electronics Engineers Computer Society, Dalian, China, 2005.
10. P. Reisner and L. Ellenberg, "Drbd v8 – replicated storage with shared disk semantics" , http://www.drbd.org, 2005.
11. D. R. Bobbarjung, S. Jagannathan, and C. Dubnicki, "Improving duplicate elimination in storage systems", ACM Transactions on Storage (TOS), vol. 2, 424-448, 2006.
12. J. Barreto and P. Ferreira, "Efficient locally trackable deduplication in replicated systems", presented at Middleware, Urbanna , Illinois, United States, 2009.
13. W. G. Aref and H. Samet, "Hashing by proximity to process duplicates in spatial databases", presented at Information and knowledge management, Gaithersburg, Maryland, United States, 1994.
14. M. Y. Eltabakh, M. Ouzzani, and W. G. Aref, "Duplicate Elimination in Space-partitioning Tree Indexes", presented at Scientific and Statistical Database Management, 2007.
15. You LL, Pollack KT, Long DDE. "Deep Store: An Archival Storage System Architecture", In: Proc. Of the 21st Conf. on Data Engineering (ICDE 2005), Washington: IEEE Computer Society Press, 804-815, 2005.