A Database Redo Log System Based on Virtual Memory Disk*

Haiping Wu, Hongliang Yu, Bigang Li, Xue Wei, and Weimin Zheng

Department of Computer Science and Technology, Tsinghua University,
100084, Beijing, P.R. China
{wuhp, hlyu, lbg01, xuewei, zwm-dcs}@tsinghua.edu.cn

Abstract. Redo log of database must be written to permanence storage like disks. When database is heavily loaded, the crowded redo log writing queue will become a performance bottleneck. In this paper, the operation principle of database redo log is analyzed. It is found that if redo log is stored on virtual memory disk directly, the database will demonstrate better performance. In addition, the reliability of redo log system based on virtual memory disk is analyzed. At the end of the paper, a contrastive performance measurement result is given.

1 Introduction

Modern database is capable of storing several TB (10^{12}) data. But it is still necessary to improve a database's performance while gaining a much larger capacity. Traditional redo log of database is stored in permanence storage like hard disks. When redo log is created, it is written in redo log buffer firstly. After redo log buffer is filled up, items of redo log are moved from buffer to log files stored in disk. Because the I/O speed of disk is much slower than CPU and memory, frequent writing operation of redo log files will become the performance bottleneck [1].

Many large databases are running on storage network area (SAN). General speaking, SAN has high performance in cluster systems. On the other hand, the redundant memory of clusters' node servers can be separated to be a virtual memory disk. The capacity of virtual memory disk can only reach 4GB in IA-32 servers; and using an IA-64 server, 100GB capacity can be gotten. It is for sure that the capacity of virtual memory disk is not enough to store all massive data of database. But, the small redo log files which are frequently written can be stored in virtual memory disk. It can improve a database's performance.

This paper describes operating principle of database's redo log in section 2. Then, a virtual memory disk based on SAN is designed. Because memory is more vulnerable than traditional disks, mirror technique is used. The reliability analysis indicates it can be used to store redo log files. At last, two databases' performances are compared, with redo log files of one database stored in virtual memory disk and

* This research was partially supported by the Chinese "863" project, sponsored by the Chinese Ministry of Science and Technology under contract 2003AA1Z2330.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 656 – 662, 2006.

those of the other are stored in SAN's permanence disk. The result indicates that the performance of the former is better than the latter.

2 Summary of Database's Redo Log

Information about database's transaction is recorded by the red log. Each item of redo log describes database users' writing operation: WRITE(Q). The redo log record is usually composed of name of transaction, name of data item Q, original value of Q before WRITE(Q) is commit and new value of Q after WRITE(Q) is commit.

Redo log also includes other items such as beginning point of transaction, end point of transaction, etc. Before a transaction commits WRITE(Q), the redo log items are created. When needed, the value of Q stored in redo log can update the value of Q stored in database, or, can be used to recover the original value [2].

Because redo log should be always usable, it must be stored in permanence storage equipment such as hard disks. For example, each Oracle database has two or more online redo log files. Oracle writes these files by circulation way: after the first file is fulfilled, redo log items are written in the second file, then, the rest may be deduced by analogy. When all online redo log files are used up, the first redo log file is written again. The database can be recovered to any point by using redo log files [3].

Redo log is so important that it must be written in permanence disk before the transaction is committed. As the size of a redo log item is much smaller than I/O unit of disk, writing single redo log item is expensive. So, the redo log buffer is created in memory and the items of redo log are written in buffer firstly. After buffer is filled up, items of redo log are moved from buffer to log files stored in disk. More over, the disk is a sort of mechanism. Its speed is far slower than CPU or memory. Frequent writing operation of redo log files will become the performance bottleneck. Acceleration of writing redo log becomes more important [4].

3 Virtual Memory Disk Based on SAN

3.1 Design of Virtual Memory Disk

With the development of data storage technique, many new storage structures come into being. SAN is a new representative storage structure. It is composed of storage control cluster, massive disk pool and high speed network. Application server which connects SAN can access data stored in network disk pool through high speed network. Compared with traditional disk, SAN is more flexible and more reliable. It can provide higher performance storage with much larger capacity.

Shu[5] has designed a SAN system. In order to improve the performance and reliability, storage control server usually adopts a cluster which is composed of several nodes. Every node in the cluster owns a big memory. The redundant memory of clusters' nodes can be separated to form a virtual memory disk. Figure 1 shows the structure of virtual memory disk.

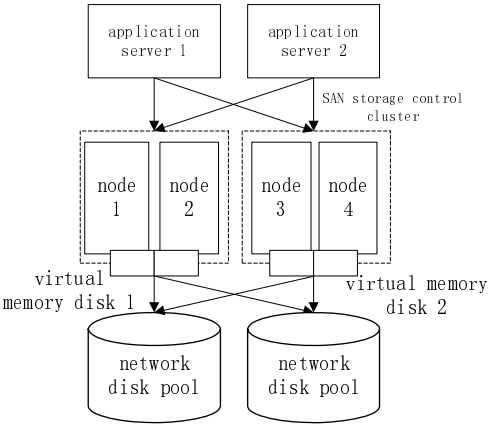

Fig. 1. Structure of virtual memory disk based on SAN

The interface between virtual memory disk and server is the same as general hard disk. The data is read from or written in memory disk directly. Because virtual memory disk is embedded in server's memory, the time which is spent finding track of disk is saved.

3.2 Reliability Analysis

Memory is more vulnerable compared with the traditional disk. When server's power is off or there is a technical problem in writing to files on the storage medium, data in the virtual memory disk will be lost. Any node server in the storage control cluster of SAN may encounter failure at any time, including hardware failure, software failure and user error. If it can't recover on time after these failures appear, the usability of system would decline greatly.

An easy solution is making a hard copy in permanence disk while data is written in virtual memory disk. Its reliability is not very good, for data's writing is asynchronous and data's consistency can not be kept well. This method will also affect data's writing performance.

The better way is using mirror disk and snapshot disk technique to improve virtual memory disk's reliability. In this method, the node servers in the storage control of SAN are divided into N groups, with every group composed of two nodes. The virtual memory disk of these two nodes are mirrored each other. When an error occurs in one node, the other node will provide virtual memory disk service in place of it and at the same time, one hard copy is made in performance disk of SAN. If two nodes in the same group encounter failure concurrently, data will be read from or written in the hard copy through other groups in the cluster. At this time, the performance of virtual memory disk will be the same as that of permanence disk. When all nodes in the cluster fail, data will be lost in virtual memory disk. Figure 2 shows mirror and snapshot copy of virtual memory disk.

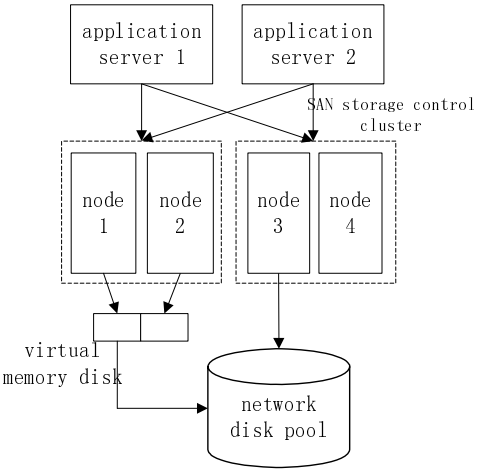

Fig. 2. Mirror and snapshot copy of virtual memory disk

It is supposed that average failure cycle of every node in the cluster is 1 week, and it takes 1 minute to recover. When the cluster has 8 nodes which can be divided into 4 groups, the chance of 8 nodes' concurrent failure is 10^{-30}, which can be ignored.

4 Database Redo Log System Based on Virtual Memory Disk

4.1 Design of Redo Log File System

Figure 3 shows a database redo log system based on virtual memory disk. The SAN in the figure 3 has four node servers and is divided into two groups, with each group

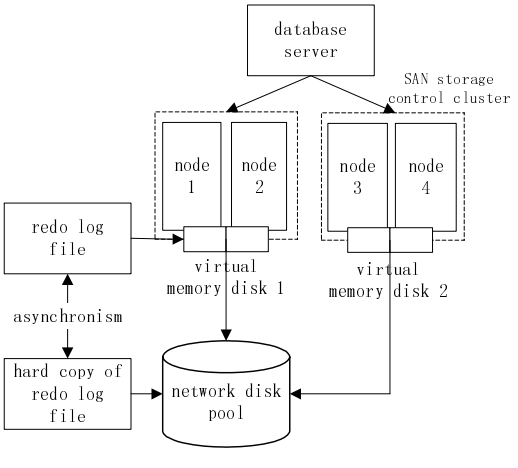

Fig. 3. Database's redo log system based on virtual memory disk

having two nodes. Some memory of node server has been partitioned and forms virtual memory disk. The virtual memory disk of two nodes in the same group are mirrored each other. So, there are two virtual memory disks in figure 3. The database redo log files are stored in these virtual memory disks.

Although virtual memory disk has better performance than permanence disk, it is more vulnerable. In order to prevent its data from being lost when servers' power is off, a hard copy in permanence network disk pool is created. When node server is idle, it writes redo log file in the virtual memory disk to its hard copy in network disk pool of SAN.

There is a log buffer in a traditional database. When a virtual memory disk is used, the disk's performance is the same as that of memory. So, redo log buffer can be canceled.

4.2 Performance Analysis

The performance of a hard disk is greatly influenced by delay time greatly. Delay time includes tracking time and rotating time. A disk rotates to find data's sector in rotating time and find data's track in tracking time. Formula 1 shows a disk's delay time [6]:

$$T_{write\_disk\_rand} = aL_{data} + b \tag{1}$$

Coefficient A is a constant. When an accessing disk's pattern is fixed, coefficient B can be taken as a constant, too.

Delay time of virtual memory disk mainly includes network transportation time, memory writing time and node processing time. It is expressed in formula 2:

$$T_{write\_mem\_band} = T_{data} + T_{mem-write} + T_{operation} \tag{2}$$

In formula 2, $T_{operation}$ is node processing time; $T_{mem\text{-}write}$ is memory writing time. They are decided by performance of a cluster's node servers and can be taken as a constant. T_{data} is network transportation time, and is proportional to size of data package.

For testing the performance of redo log system stored in virtual memory disk, an Oracle database is installed in TH_MSNS[5], a self-designed SAN system. Performances are compared when redo log file is stored in virtual memory disk and permanence disk. The system configurations can be seen from Table 1.

Table 1. System configurations

Storage Capacity	4T
Network bandwidth in the SAN	2Gbps
Operating system	Redhat Linux 2.4.18
Database	Oracle9i
Database server	Intel Pentium4 Xeon 2.4Ghz, 2Gb memory
Storage control cluster node	Intel Pentium4 Xeon 2.4Ghz, 2Gb memory

From formulae 1 and 2, it is known that delay time is related to the size of data package. Database data block can be 4KB, 8KB, 16KB, and 32KB, etc. According to

the size of data block, delay time is tested separately. The result is lineal approximation, which can be expressed in the following formulae:

$$T_{write\_disk\_rand} = 0.27L_{data} + 66 \tag{3}$$

$$T_{write\_disk\_rand} = 0.064L_{data} + 0.025 \tag{4}$$

From formula 3 and 4, the performance of a virtual memory disk is much better than a permanence disk. When data package size is 4KB, the ratio of delay time of a permanence disk and a virtual memory disk is 1:237; when data package size is 64KB, the ratio is 1:20. It is obvious that the smaller data package size is, the better performance of a virtual memory disk is.

The database's performance is measured by Benchmark Factory V3.3 which is developed by Quest Corporation. The tool adopts TPC-C benchmark established by Transaction Processing Performance Council. As an OLTP system benchmark, TPC-C simulates a complete environment where a population of terminal operators executes transactions against a database. The benchmark is centered on the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. We simulate 100, 200, 300, 400 and 500 users to access database concurrently and get database's transaction numbers in time unit. The result is shown in Table 2 and 3.

Table 2. The transaction number in time unit when stored in permanence disk

Users	4KB	8KB	16KB	32KB	64KB
100	380	434	450	471	506
200	762	870	896	1206	1208
300	1151	1291	1390	1415	1524
400	1528	1598	1673	1737	1895
500	1873	2187	2184	2289	2409

Table 3. The transaction number in time unit when stored in virtual memory disk

Users	4KB	8KB	16KB	32KB	64KB
100	426	473	485	501	528
200	847	942	1062	1209	1284
300	1275	1402	1488	1537	1603
400	1701	1823	1836	1906	2018
500	2104	2309	2298	2413	2536

It is obvious that the transaction number increases with increase of the number of users. The performance of the system improves lineally. The result also indicates that the performance is better when redo log files are stored in a virtual memory disk rather than in a permanence disk. When data block is 4KB, the average ratio of

improvement is 1.12: 1; when data block is 8KB or 16KB, the average ratio is 1.09: 1; when data block is 32KB or 64KB, the average ratio is 1.06: 1.

Because the database performance is not only related to redo log files, but also related to the size of data block buffer. Data block buffer is used to store data which database users usually access to. So, the bigger a data block is, the more beneficial it is to improve a database's performance. We can see from the experiment that when the block size is increased, the ratio of the performance gained from virtual memory disk declines.

5 Conclusion

When a database is heavily loaded, frequent writing operation of redo log files will become the performance bottleneck. This paper designs a database redo log file system which is stored in a virtual memory disk. The average ratio of a database's performance get an improvement of 6 to 11 percent when redo log files are stored in a virtual memory disk compared with a permanence disk.

In traditional database, the system tablespace and rollback segment are also the factor which influences database's performance. It is also possible to store them in a virtual memory disk, and more performance gains can be expected. Further studies need to be carried out in terms of this.

References

1. Lomet, David B.: Persistent applications using generalized redo recovery. Proceedings of the 1998 14th International Conference on Data Engineering, (1998) 154-163.
2. Haerder, Theo, Reuter, Andreas.: Optimization of logging and recovery in a database system. ACS Symposium Series, (1979) 151-168
3. Lance Ashdown, Valarie Moore.: Oracle9i Backup and Recovery Concepts, Release 2 (9.2). Oracle Corporation. Oracle Parkway, Redwood City (2002).
4. Kumar, Vijay, Moe, Shawn, D.: Performance of recovery algorithms for centralized database management systems. Information Sciences, V.86, n1-3, (1995) 101-147.
5. Shu, J.W., Xue, W., Li, B.G., Zheng, W.M.: TH-MSNS: A High Scalable Network Storage System. Chinese Journal of Computers. V.28, n3, (2005), 326~333.
6. Jin C., Zheng, W.M., Mao, Y., Wang, D.S.: Two methods to improve the performance of synchronous disk I/O in Linux environment. The Third LCI International Conference on Linux Clusters, St. Petersburg, USA, 2002.