

A Design of a Transparent Backup System Using a Main Memory Database

Hiroki TAKAKURA Yahiko KAMBAYASHI

Integrated Media Environment Experimental Lab.,
Faculty of Engineering, Kyoto University
Sakyo, Kyoto 606-01 JAPAN

Abstract

In this paper, we mainly focus on a problem to realize high-performance systems without sacrificing system reliability. One solution for high-performance is to use main memory as a database storage. Since main memory cannot be free from software bug or electricity hazard, we cannot avoid archive storages such as disks. To reduce the time of checkpoint and recovery for a main memory database, some mechanism is required. For this purpose we have developed continuous backup RAMs which can store their contents periodically to an archive storage while they are used for usual process. By extending this concept continuous backup disks are also designed. A transparent backup system is organized using such RAMs and disks. As a backup process is performed by hardware mechanism during the execution of usual database operations, a backup process does not require to stop a system or to keep transactions waiting to record system status at each checkpoint and thus, users are not aware of the existence of a backup process. Furthermore, an interval between two adjacent checkpoints can be shortened, since hardware control does not require much computation overhead. This short interval contributes to reduce recovery time.

1 Introduction

Recently, databases have been widely used in systems of various fields such as online process control. This kind of systems must have the property of a real-time system, i.e. all processes can be finished before predefined deadline, and also must have the property of a fault-tolerant system where control operations should not stop even if a system fails, since the process of a plant cannot be stopped instantaneously. If a system cannot decide an appropriate action to prevent a plant from being runaway within predetermined time or if a system stops for even a very short period, then it may become unable to control a whole plant and we may suffer a great loss. For such applications, a high performance and highly reliable system must be realized.

To achieve high performance, a lot of papers have been published on query processing, concurrency control, database machines and main memory databases. One should notice, however, that even if high speed database operations are realized

we must still stop a system for recovery when some serious failure occurs.

To realize high reliance a fault-tolerant system employs a dual system approach which avoids time-consuming recovery process[CER85][KIM84]. In that system data are available regardless of any single failure of a system. Repairs can be done without affecting availability of data and it is assumed that repairs can be completed before next failure. It is required, however, to record system status for multiple failure cases.

A fault-tolerant system is designed to be based on the disk resident database, since main memory in general does not have enough reliability compared with a disk, even if it is nonvolatile. The disk resident system has a serious drawback due to disk I/O bottleneck to realize a high performance system.

A highly reliable main memory database system which has both properties of high performance and high reliance should be developed to solve these problems. In order to realize such systems continuous backup RAMs are designed by the authors[KAM91]. These RAMs can store their contents to archive storage while they are used for usual process.

In this paper, continuous backup disks are introduced which are extensions of continuous backup RAMs. A system architecture to reduce overhead for recovery using such RAMs and disks is discussed.

We supposed only the case where current data are lost by some failure such as system failure and media failure, and at least one checkpoint data can be available. Transaction failure is not considered here, since it does not make data lost and should be recovered by database management system.

Conventional checkpoint and recovery processes are as follows.

- A) At each checkpoint, dirty pages or whole database pages are dumped to an archive storage, such as tapes or disks.
- B) While database operations are performed, log records are stored in a stable storage (tapes or disks).
- C) In case of system failure, the status at the latest checkpoint is transferred to a system from its archive storage and a system can restart from this checkpoint. In case of media failure, a system must be repaired at first.

- d) Using log records after this checkpoint, the status of a system can be recovered just before the point when failure happened.

Overhead at each step is as follows.

- a) A system have to be stopped while dumping checkpoint data or transactions which require to write dumping pages must wait until dump of these pages completes.
- b) To take log records requires CPU consumption and results in system overhead.
- c) After failure the checkpoint status needs to be reloaded to a system. The reloading usually takes time.
- d) In addition to the reloading time, the computation time is also required for D).

Use of reliable main memory which consists of the battery backed-up RAM or electrically erasable ROM (Flash Memory) can avoid the dumping operation. Combined with battery operated disks[COP89] a reliable storage system may be able to be organized. Even if such a system is used, recovery process cannot be avoided. Examples are failures caused by memory peripheral circuit failure and by software bugs

In order to reduce the time required for c) and d), incremental reloading[LEH87] can be used, where data are recovered only when they are required for the first time. Even in such a system the time to load essential data such as database management system, operating system, etc., is needed before restarting a system.

Our system reduces these overheads as follows.

- a) A continuous backup RAM and a continuous backup disk are used to get the status of each checkpoint without stopping a system or keeping transactions waiting, since they are controlled by simple hardware mechanism which works independently of database process.
- b) Although details are not discussed in this paper, we are currently designing hardware mechanism to get log records efficiently, which will contribute to reduce software overhead. The outline of log record structure will be mentioned in Section 2.
- c) Our system has redundant main memory and disk which store the status at the latest checkpoint. By connecting these main memory and disk, the status of a system at this checkpoint can be recovered. The reloading time is unnecessary.
- d) As a system is not required much overhead to take checkpoint data, the interval between two consecutive checkpoints, which contributes to reducing the average time required for D), can be reduced.

The organization of this paper as follows. In Section 2 basic concepts required to describe our system are given. Organization of continuous backup memories and that of continuous backup disks, which are the major components of our system, are discussed in Sections 3 and 4, respectively. The overall system architecture and the recovery process are discussed in Section 5. Performance evaluation is carried out in Section 6.

2 Basic Concepts

In this section necessary basic terms and a problem are discussed.

Hot Spot Data(80/20-Rule)

The access of database may concentrate in very small part of database. In typical database systems, it is known that about 80 percent of access concentrates in about 20 percent of database (80/20-Rule)[GAW85]. Such data are called hotspot data.

A Partial Main Memory Database

A main memory database originally means that whole database is resided in main memory[EIC86][EIC87]. Although a main memory database is one effective solution to realize high performance database systems, it is not currently realistic because of cost and capacity limitation.

In such a case, we propose a partial main memory database where hotspot data are resided in main memory and another data are read from a disk on demand. To use a partial main memory database may be one answer to solve these limitations and it can realize high performance without using the very large memory required for a main memory database. We expect, however, that a main memory database can be realized in the near future.

Write-Ahead Logging(WAL)

Some systems such as IBM ARIES[MOH89] are based on the write-ahead logging (WAL) protocol where a log record must save in a stable storage before a dirty page is reflected to database. In the system where physical log records are used, in-place updating that a dirty page is written back to the same nonvolatile storage location from where it was read must be performed[MOH89].

Checkpoint and Recovery

At recovery possible latest contents of main memory and a disk should be calculated using checkpoint data and log records. Although to reduce this calculation overhead checkpoint should be taken as frequently as possible, frequent checkpoint operations cause more overhead to a system, since in the conventional database system these operations are controlled by software and, thus, require CPU which is also used for database operations.

A Problem between Backup and Logging

Some papers on backup of a main memory database claim that they can be used in both original and partial main memory database systems. Most of them use physical logs and suppose that whole or a part of main memory is nonvolatile. The physical log can be used in a main memory database system because this system can guarantee in-place updating, but a partial main memory database system, like our system, cannot use such a physical log, since hotspot pages are removed from main memory when they become non-hotspot and, thus, in-place updating cannot be guaranteed.

Although logical log can be avoid this problem, since it can indicate data object by their logical name, this log causes another problem when a backup operation is performed asynchronous with database operations. In case that a page is backed up during it is being updated by one logical operation, a system cannot recover from failure using a logical log,

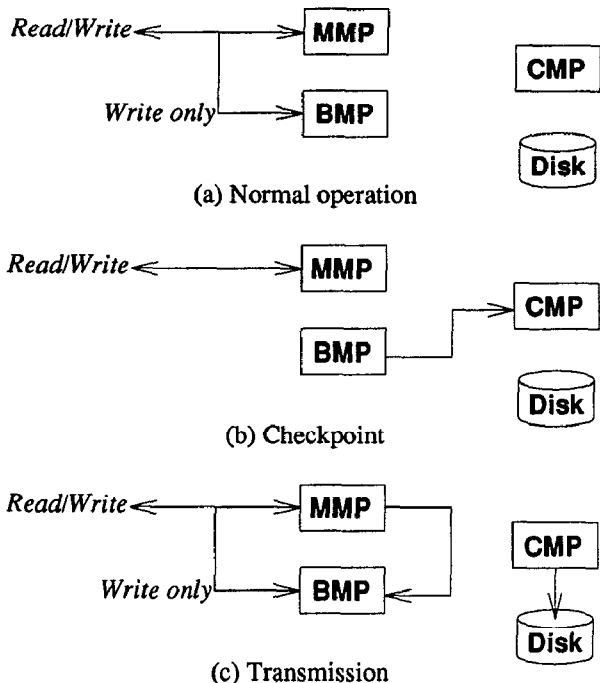


Figure 1: Two-Plane Backup System

since already reflected part cannot be distinguished from not reflected part.

To avoid these problem, we have been studying a new log management system. The outline of this system is as follows. To manage a log record of non-hotspot data, our system uses conventional log management method, like ARIES, except that only a physical log record is used. Every log record is assigned unique log sequence number (LSN) and every data page has page_LSN field which contains LSN that describe the latest update log record to the page[MOH89].

To manage the log record of hotspot data, our system uses a new log management system which is controlled by hardware to reduce the overhead. The log record has similar format as that of non-hotspot data. Each transaction produces at most a 256byte log record which contains transaction's ID, address of a main memory page, and after and before images of the record.

Our system needs log records describing data replacement between new hotspot page and non-hotspot page which were hotspot and resided in main memory. This log record contains disk address of the new hotspot page.

3 Continuous Backup RAM

In this section three methods to realize continuous backup RAMs are summarized[KAM91]. The first one is shown in Figure 1.

3.1 Two-Plane Backup System

A system consists of three memory planes, main memory plane (MMP), backup memory plane (BMP) and checkpoint memory plane (CMP). A disks is attached to CMP which is used to store main memory contents. Figure 1(a) shows the

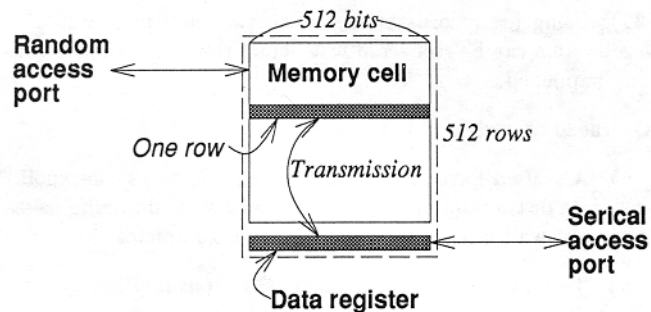


Figure 2: The Structure of Dual-Port RAM

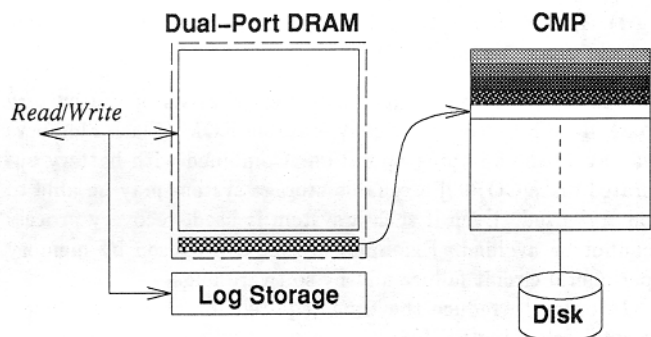


Figure 3: Organization of Continuous Backup RAMs Using Dual-Port RAMs

situation for normal access. At a checkpoint, BMP is disconnected from CPU which works for database process and starts to transmit its contents to CMP (Figure 1(b)). After transmission, BMP is again connected to CPU and CMP stores its contents in a disk (Figure 1(c)). The system performs write operations to both MMP and BMP, and read operations to MMP. Contents of MMP are transmitted to BMP by DMA (direct memory access) control concurrently with these read and write operations in a time-shared manner. After transmission contents of MMP and BMP become identical, since write operations during transmission are also applied to BMP. As the DMA transmission is performed simultaneously with database operations, database operation becomes slower when it competes with the DMA operation for the same memory module.

3.2 Continuous Backup by Dual-Port DRAMs

Figure 2 shows a structure of dual-port DRAM, which is developed for display memory. Memory contents can be modified through a random access port while a serial access port is used to display its contents. One row data of a memory cell are transmitted to a data register when row address is given to random access port. As this mechanism is realized by hardware, it is very much efficient. The organization of a continuous backup system using dual-port DRAM as MMP is shown in Figure 3. At one access only one row's data can be transmitted to CMP. Thus a row's data will be stored at different time. Let T_{Trans} be the time required to transmit all contents to CMP. The data transmission for i , i th checkpoint time, must be started before $t_i - T_{Trans}$. Write operations after $t_i - T_{Trans}$ are stored in

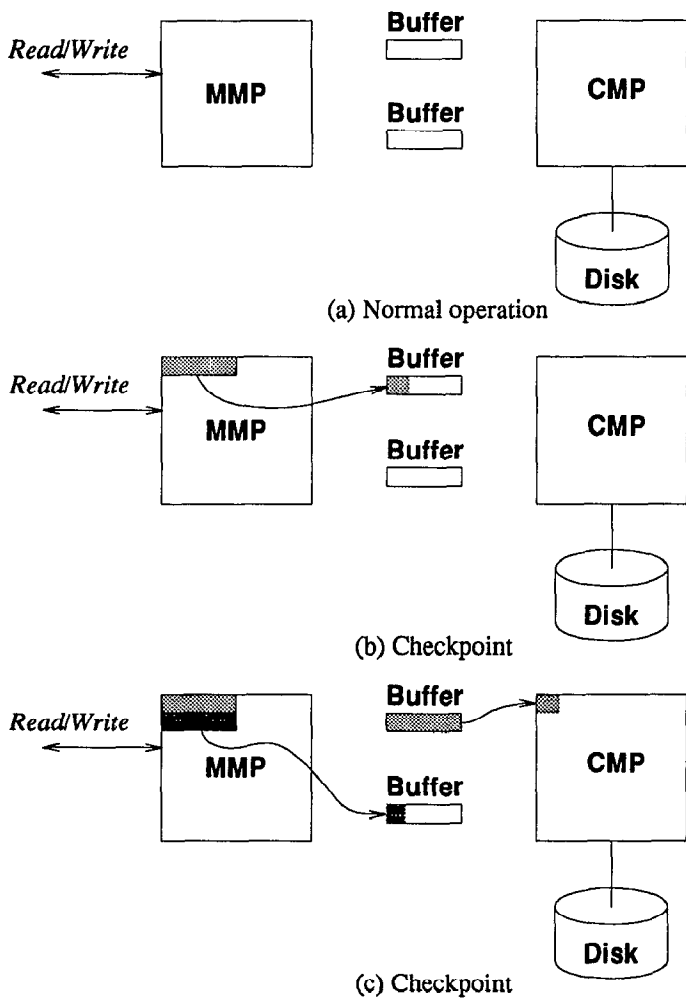


Figure 4: Double Buffer Backup System

Log Memory. At t_i , all write operations performed between $t_i - T_{Trans}$ and t_i are stored in Log Memory. Using contents of Log Memory, contents of CMP must be modified and then memory contents at t_i are obtained. After this operation CMP stores the data at the checkpoint t_i . During this modification process failure of dual-port DRAM may occur. In such a case both CMP and Log Storage must be used in order to get correct data values for recovery.

3.3 Double Buffer Backup Method

A problem of using currently available dual-port DRAMs is that access speed is slow, since it is designed for conventional TV systems such as NTSC. The access cycle for random access port is about 200[ns]. For HDTV (High Definition Television) random access port would become faster. We expect such dual-port RAMs will be available in the near future.

Double buffer backup is developed to simulate the operation of dual-port DRAM, in order to realize the high-speed operation. The architecture is shown in Figure 4(a). This is a hardware version of the double buffer algorithm. A part of MMP's contents is transmitted to one buffer and at the same time contents of the other are sent out to CMP. Two buffers

are alternatively used (Figure 4(c)). Compared with a two-plane backup system, hardware cost can be reduced, although modification by Log Memory is a little bit complicated.

4 Continuous Backup for Disk

We can design continuous backup disk similar to memories discussed in previous section when a partial main memory system is used. One typical characteristics of disk should be considered; one sequential access is much faster than a number of direct random access to retrieve the same amount of data.

4.1 Two-Disk Backup System

The organization is similar to the system shown in Figure 1. Here, MMP, BMP and CMP are disks and Log Memory discussed in Section 3.2 is required. After transmission, BMP catches MMP up using Log Memory data, since transmission from MMP to BMP competes usual data access and causes MMP many random accesses. As the sequential access is preferred, contents of Log Memory must be sorted in order to reduce the time required for catching up.

4.2 Simulation of Dual-Port DRAM

A dual-port disk can be developed, if it has one extra read head. One head is used for usual read/write purpose, while other head is used for sequential access to read backup data. One cylinder is considered to correspond to one row of dual-port DRAM. Differently from dual-port DRAMs, while sequential access port is used to read backup data, write operations using usual port must be prohibited and stored in main memory, since the magnetic field generated by usual port's head may affect data read from the sequential access port. Stored write operations are not lost by failure, since backup operations perform for both main memory and a disk at a checkpoint and thus they are also stored as checkpoint data of main memory. Read operations through both ports can be performed simultaneously.

Since to develop a new hardware is not the cost effective, currently available disk using disk cache can be used. Figure 5 shows an organization of such a system, here cache is assumed to have its own backup disk. During the sequential read to cylinders, all requests for the read and the write are stored in cache. Only data existing in cache can be read. Sequential

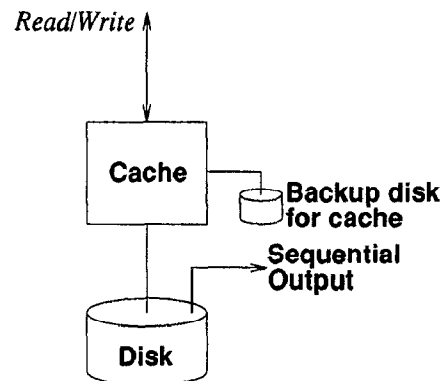


Figure 5: Simulation of Dual-Port RAMs by a Disk

access are done continuously and after sequential access completes all the requests in cache must be applied.

5 Organization of a Transparent Backup Database System

By combining previous discussion we will discuss how to organize a transparent backup database system in this section.

It is assumed that checkpoints are denoted by t_1, t_2, \dots, t_i . For simplicity it is assumed that the system status can be completely described by (1) contents of CPU registers, (2) contents of main memory, and (3) contents of disk. Let R_t, M_t , and D_t are these contents at time t .

Figure 6 shows basic organization of a transparent backup database system and its recovery process. The effective connection is shown by bold lines. Let us assume that the current time t satisfies $t_i \leq t < t_{i+1}$ (i.e., the latest checkpoint is t_i).

In Figure 6(a) the current contents are stored in disk and main memory shown in the left side. Right side disk and main memory store the data at the checkpoint t_i . The small disk attached to right main memory holds memory image of M_{t_i} , which will be used for recovery process of main memory. After system failure, main memory and disk storing checkpoint data are used as shown in Figure 6(b). Thus reloading is not required. Disk and main memory in the left side can be used to store the next checkpoint data. The status of CPU registers is assumed to be also stored in main memory as a part of checkpoint data, since only several steps of instructions are required to store them.

Since a system is not safe if the crash occurs while taking the next checkpoint, an additional set of main memory and disk is needed. There are two cases, i) the additional set also stores D_{t_i} and M_{t_i} (duplicated backup) or ii) $D_{t_{i-1}}$ and $M_{t_{i-1}}$.

The whole system organization is outlined in Figure 7. Memories in Systems II and III are alternatively used to store memory data at checkpoint (CMP), so that data at the two consecutive checkpoints can be stored. A memory in System I are used to store the current state of main memory (MMP and BMP). Log Storage stores all log records for main memory and disk after t_{i-1} . Since transmission of contents of System I is con-

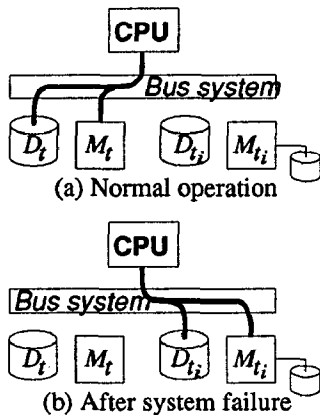


Figure 6: Basic Organization of a Transparent Backup Database System

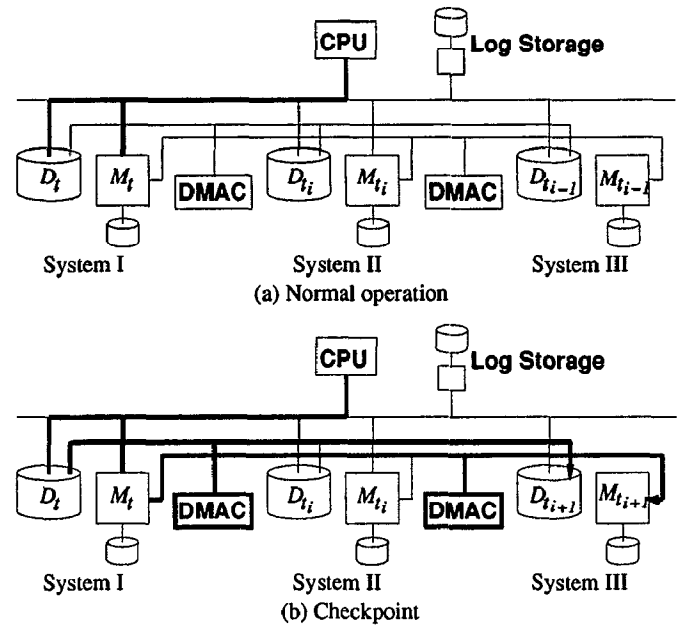


Figure 7: Organization of a Transparent Backup Database System

trolled by DMACs (Direct Memory Access Controller) which are connected with sub buses shown in Figure 7, this transmission does not cause any contention with main bus access.

If some crash occurs System II is used and system can restart from the checkpoint t_i . If another crash occurs before t_{i+1} System III has to be used. Using Log Storage contents of System III can be modified to very close to the crash time. If further crash occurs before t_{i+1} , a system must reload from an archive storage, which stores sampled checkpoint data. In such a case recovery becomes slow, but we believe that such a case rarely happens.

6 Evaluations of a Transparent Backup Database System

In this section we discuss performance evaluation of our system on the time to take checkpoint data and to recover from failure. The evaluation is based on TPC benchmark B[GRA91].

6.1 Outline of TPC Benchmark B

In TPC benchmark B each transaction updates four databases; Account, Branch, Teller and History. Basically, the size of databases, except for History, is in proportion to the number of transaction throughput(tps). History disk must be large enough to store records which are produced for the eight hour running. The profile of a transaction is as follows.

- Begin transaction.
- Update a record of Account.
- Update a record of Teller.
- Update a record of Branch.
- Insert a record into History.
- Commit transaction.

The definition of databases is shown in Table 1.

Database	Record length	Number of record/ <i>tps</i>
Account	100B	100000
Branch	100B	1
Teller	100B	10
History	50B	

Table 1: Definition of Databases

	Current	A few years later
Bit per inch	50000	70000
Track per inch	2000	3000
Rotation [rpm]	3600	5000
Serial data transfer rate	15Mb/s	35Mb/s

Table 2: Disk Technology

Number of tracks	Number of sector	Capacity per a surface
2000	108	100MB

Table 3: 3.5 Inch Disk Drive Specification

6.2 Specification of Disks

The current and near future disk technology are shown in Table 2. The future 3.5-inch disk specification expected form Table 2 is shown in Table 3. The seek time is supposed to be calculated with a following formula[STR83],

$$T_{Seek} = a + b\sqrt{i}$$

Here, $a = 5[\text{ms}]$, $b = 0.5[\text{ms}]$ and i denotes distance from starting cylinder to destination cylinder.

6.3 Performance of Checkpoint Operations

In this section, the time to take checkpoint is estimated. Main memory (disk) is supposed to be divided into several blocks (drives) and backup operation of each main memory block (drive) is also supposed to be performed parallel.

At first, the time to back up disk contents is estimated. It is supposed that contents stored in a track can be read while a disk makes one rotation. Time to transfer data between disks is,

$$N_{Cylinder} \cdot (T_{Seek} + T_{Rotation}) \cdot N_{Surface}$$

Here, $N_{Cylinder}$ denotes the number of cylinders and $N_{Surface}$ denotes the number of disk surfaces of a drive. T_{Seek} represents seek time calculated using the formula discussed section 6.2 where $i = 1$ and $T_{Rotation}$ represents rotation time. From Table 2, the expression above becomes,

$$35 \cdot N_{Surface}[s]$$

The result is shown in Figure 8.

Our checkpointing method need not to optimize disk access scheduling, since only sequential access is performed by hardware control.

Next, the time to back up main memory contents is estimated. Here, a dual-port DRAM backup system is supposed to be used. The size of a main memory block is denoted by S_{Block} . The access cycle of serial access port is denoted by T_{Serial} and the bus size of each block is denoted by S_{Bus} . When $Serial_Cycle = 40[\text{ns}]$ and $Bus_Size = 4[\text{byte}]$, the time to

take checkpoint is,

$$\frac{S_{Block}}{S_{Bus}} \cdot T_{Serial} \simeq 10 \cdot S_{Block}[\text{ns}]$$

The result is shown in Figure 9.

6.4 Performance of Recovery Operations

Considering the characteristic of the TPC benchmark, all records of Branch and Teller are hotspot and should be resided in main memory. In this paper 80 percent of transactions is supposed to access the hotspot records of Account. In respect to the TPC benchmark, the access pattern of Account is uniform, i.e. there is no hotspot record in Account, but when it is supposed, large main memory on which 20 percent of databases can be resided is needed. The evaluation of recovery under this supposition shows temperate results. Each record of History is produced and stored in the database only once, so History has no hotspot record.

The interval of two adjacent checkpoints is denoted by $T[s]$. During T , at most $tps \cdot T$ transactions commit. In our log management system, each transaction produces at most four 256-byte log records.

In our system, the incremental recovery where only hotspot data are recovered immediately is performed. In this paper, the time to recover hotspot data is discussed. The recovery is supposed to be the time to read log records from Log Storage (disk), because our system need not to reload hotspot data and this time is the most part of recovery procedure. Then, the recovery time becomes maximum when a failure occurs just before the checkpoint and its time is,

$$\frac{tps \cdot T \cdot (1 + 1 + 0.8) \cdot S_{Log}}{R_{Serial}} \simeq 156 \cdot tps \cdot T[\mu\text{s}]$$

Here, R_{Serial} represents the access rate to read disk data serially and S_{Log} represents the size of a log record.

The result from this expression is shown in Figure 10. The minimum recovery time is 0[s] when the latest checkpoint data can be used immediately (a failure occurs just after the checkpoint).

This result shows that if the system cannot stop even for a few seconds the incremental recovery where most needed hotspot data are recovered immediately must be performed, or the system must restart using only contents of the checkpoint abandoning log utilization. We cannot think, however, that such a system is reliable.

7 Concluding Remarks

In this paper a new approach to reduce the overhead of a recovery process is introduced. Since a highly available system is essential for various applications, we can design a super highly available system by combining the currently available technologies, for example a fault-tolerant system and a disk array technology. Using a disk array total time for backup can be reduced due to their parallel processing property. Although a triple redundant system is expensive, we expect this technology will be practically used, since the annual reduction of cost of the disk for one bit is said to be about 40% and that of the memory chip is about 25%.

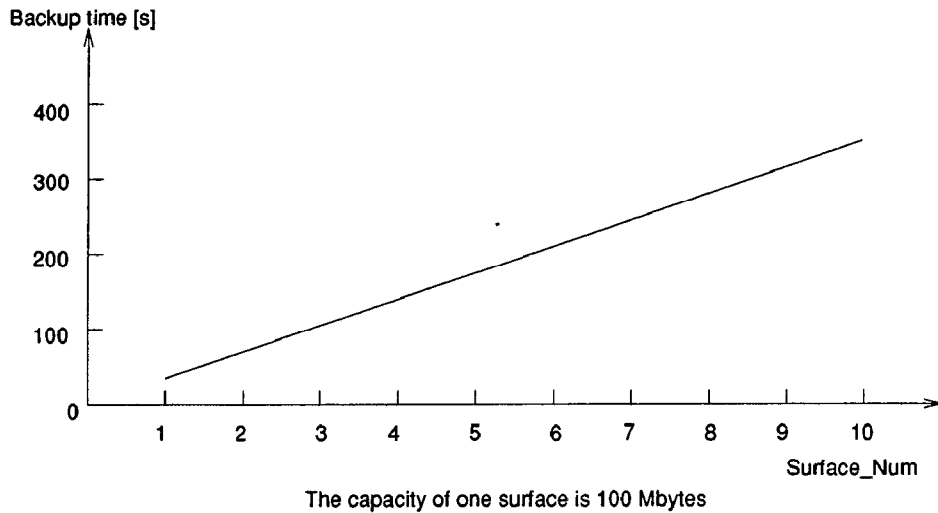


Figure 8: Surface Number vs Backup Time (Disk)

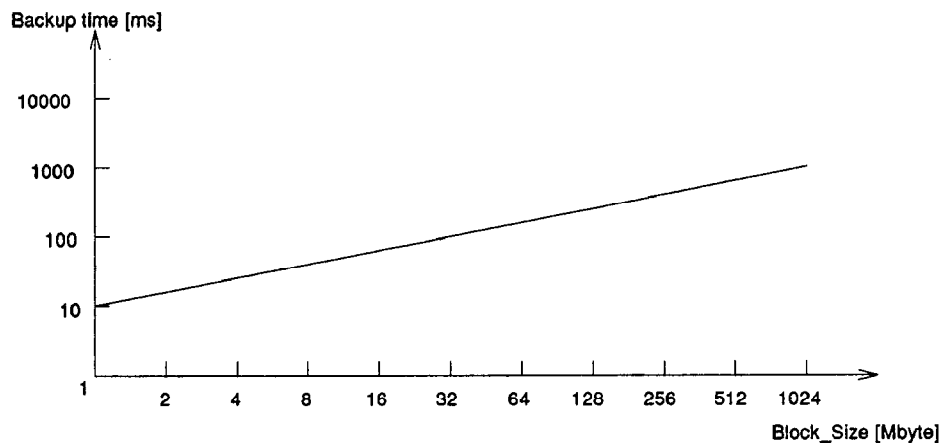


Figure 9: Block Size vs Backup Time (Main Memory)

The authors have been discussing on the use of flush memory which is a kind of EEPROM as an archive storage. This approach is expected to realize efficient checkpoint and recovery procedures[TAK93].

Acknowledgment

We would like to express our sincere appreciation to Professor Kazuo Iwama of Kyushu University for his constructive criticism and valuable insight.

References

- [BAR91] N.S. Barghouti, G.E. Kaiser, "Concurrency Control for Advanced Database Applications," ACM Computing Surveys, Vol.23, No.3, 1991, pp.269-317.
- [BER87] P.A. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison Wesley, 1987.
- [CER85] S. Ceri, G. Pelagatti, "Distributed Databases Principles and Systems," McGraw-Hill Book Company, 1985, pp.292-298.
- [COP89] G.Copeland, T.Keller, R.Krishnamurthy, M.Smith, "The Case For Safe RAM," Proc. of the 15th International Conf. on VLDB, 1989, pp.327-335.
- [DEW84] D. DeWitt, et al, "Implementation Techniques for Main Memory Database Systems," Proc. of ACM SIGMOD Conf., 1984, pp.1-8.
- [EIC86] M.H. Eich, "Main Memory Database Recovery," ACM FJCC, 1986, pp.1226-1232.
- [EIC87] M.H. Eich, "A Classification and Comparison of Main Memory Database Recovery Techniques," Proc. IEEE 3rd Conf. on Data Engineering, 1987, pp.332-339.

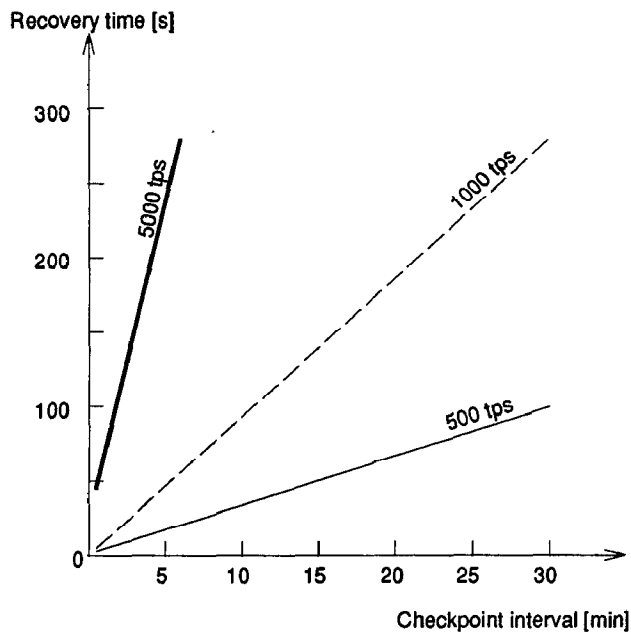


Figure 10: Checkpoint Interval vs Recovery Time
(Worst Case)

- [EIC88] M.H. Eich, "MARS : The Design of a Main Memory Database Machine," Database Machines and Knowledge Base Machines, Kluwer Academic Publishers, 1988, pp.325-338.
- [EIC89] M.H. Eich, "Main Memory Database Research Directions," Proc. 6th International Workshop, IWDM'89, 1989, pp.251-268.
- [GAW85] D. Gawlick, "Processing 'Hot Spots' in High Performance Systems," Proc. of IEEE Spring Computer Conference, 1985, pp.249-251.
- [GRA91] Jim Gray, "The Benchmark Handbook," Morgan Kaufmann Publishers, 1991, pp.19-117.
- [GRU91] L. Gruenwalld, M.H. Eich, "MMDB Reload Algorithms," Proc. of ACM SIGMOD International Conf. on Management of Data, 1991, pp.397-405.
- [HAG86] R.B.Hagmann, "A Crash Recovery Scheme for a Memory-Resident Database System," IEEE Trans. on Computers, Vol. C-35, No.9, September, 1986, pp.839-843.
- [KAM91] Y. Kambayashi, H. Takakura, "Realization of Continuously Backed-up RAMs for High-Speed Database Recovery," Database Systems for Advanced Applications '91, World Scientific, 1992, pp.236-242.
- [KIM84] W. Kim, "High Available Systems for Database Application," ACM Computing Surveys, Vol.16, No.1, 1984, pp.71-98.
- [KUM91] V. Kumar, A. Burger, "Performance Measurement of Some Main Memory Database Recovery Algorithms," Proc. 7th Int. Conf. Data Engineering, 1991, pp.436-443.
- [LEH87] T.J. Lehman, M.J. Carey "A Recovery Algorithm for A High-Performance Memory-Resident Database System," Proc. ACM SIGMOD Conf., 1987, pp104-117.
- [LEH89] T.J. Lehman, M.J. Carey "A Concurrency Control Algorithm for Memory-Resident Database Systems," Proc. 3rd International Conf, FODO, 1989, pp.490-504.
- [MOH89] C. Mohan, D. Haderle, B. Lindsay, H. Piatesh, P. Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," IBM Research Report RJ6649, IBM Almaden Research Center, January 1989; Revised April 1991.
- [MOH90] C. Mohan, K. Treiber, R. Obermarck, "Algorithms for the Management Remote Backup Data Base for Disaster Recovery," IBM Research Report RJ7885R, IBM Almaden Research Center, December 1990; Revised June 1991.
- [STR83] R.A. Scranton, D.A. Thompson, D.W. Hunter, "The Access Time Myth," IBM Technology Report, RC10197, September, 1983.
- [TAK93] H. Takakura, Y. Kambayashi, "Continuous Backup Systems Utilizing Flash Memory," Conf. on Data Engineering, 1993 (to appear).