로그 기반 백업 및 안정기억장치를 이용한 주기억장치 DBMS에서의 회복기법

최미선, 김영국
충남대학교 컴퓨터과학과
(mschoi, ykim)@cs.cnu.ac.kr

A Log-Driven Backup Method Using Stable Memory in Main Memory Database

Mi-Seon Choi, Young-Kuk Kim
Dept. of Computer Science, Chungnam National University

요 약

메모리 기술의 발전은 주기억장치 데이터베이스가 널리 사용될 수 있는 토대를 마련하였다. 주기억장치 데이터베이스에서는 주기억장치의 휘발성으로 인해 시스템이 파손될 경우 모든 데이터들을 잃어버리게 되므로 데이터베이스를 일관된 상태로 복구시켜주는 회복기법이 매우 중요하다. 본 논문에서는 비휘발성 메모리이면서도 RAM과 같이 빠른 접근속도를 가지는 FRAM과 같은 고성능 비휘발성 메모리를 이용하여 그림자 갱신과 로그 기반 백업을 실시함으로써 시스템 파손 후, 빠른 복구를 제공하면서도 기존의 로그 기반 백업이 가지는 문제점을 해결하는 회복기법을 제안한다.

1. 서론

반도체 기술 발전과 가격 하락에 따라 실시간 응용을 효율적으로 지원하기 위한 주기억장치 데이터베이스(Main Memory Database : MMDB)의 필요성이 증대되고 있다. MMDB는 데이터를 디스크로부터 주기억장치로 이동할 필요가 없고 데이터에 대한 직접 접근이 가능하므로 디스크 기반 데이터베이스(Disk-Resident Database : DRDB)보다 더 빠른 응답 시간과 높은 트랜잭션 처리 능력을 가지는 반면에, 시스템이 파손될 경우, 주기억장치에 저장되어 있던 데이터들을 모두 잃어버리게 되는 문제점을 가지고 있다.

데이터베이스 회복 기법은 시스템이 동작하는 중에 복구 과정에서 필요한 정보들을 안전한 저장 장소에 기록하여 두었다가 오류가 발생하면 기록된 정보를 이용하여 데이터베이스의 일관성을 유지시키기 위한 기법이다. 그러나 시스템 동작 중에 백업 디스크나 로그 디스크를 접근하는 것은 트랜잭션 수행 시간을 지연시켜서 전체 시스템 성능에 많은 영향을 주기 때문에 MMDB의 특성도 저하시킨다.

대부분의 MMDB 회복 기법들은 디스크에 저장되는 로그의 양을 최소로 줄여서 그 입출력 횟수를 줄이고, 체크포인트가 트랜잭션 수행에 미치는 영향을 최소로 하며, 시스템 복구 과정에서 빠른 재수행을 지원함으로써 MMDB 시스템의 성능을 높이고 있다. 최근에는 다중 프로세서를 이용하여 트랜잭션 및 로그 처리를 병렬로 수행함으로써 MMDB의 성능을 높이는 회복 기법에 대한 연구도 나오고 있다[6].

MMDB에 있어서 메모리 기술의 발전이 가져다 준 또 하나의 이득은 비휘발성 특징을 가지면서도, RAM과 유사한 동작 방식 및 접근 속도를 가지는 FRAM(Ferro-Electric Random Access Memory)과 같은 차세대 메모리의 출현이다. FRAM은 RAM 타입 메모리들의 치명적인 약점인 휘발성 문제를 해결한 비휘발성 메모리이면서도 기존의 비휘발성 메모리에 비해 매우 빠른 기록속도를 가지고 있다[8].

본 논문에서는 FRAM과 같은 고속 안정메모리(stable memory)를 활용하여 로깅과 백업을 하나의 프로세스로 통합하는 로그 기반 백업을 실시함으로써 시스템 파손 시 빠른 복구를 제공하면서도, MMDB 시스템의 성능을 유지할 수 있도록 하는 MMDB 회복 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2 절에서는 관련 연구로서 MMDB 회복 기법들에 대해 설명하고, 3 절에서는 시스템 구성 및 트랜잭션 처리와 회복 관련 알고리즘에 관하여 설명한다. 4 절에서는 결론을 맺고 향후 연구 방향을 기술한다.

2. 관련 연구

데이터베이스에서 데이터 갱신 방식은 직접 갱신(immediate update) 방식과 그림자 갱신(shadow update) 방식으로 분류된다. 직접 갱신 방식은 트랜잭션이 커미트(commit)되지 않은 상태에서 트랜잭션에 의해 변경된 내용을 데이터베이스에 직접 반영하는 방식이다. 그림자 갱신 방식은 변경 내용을 별도의 영역에 유지하였다가, 트랜잭션이 커미트되었을 때, 변경 내용을 데이터베이스에 적용하는 방식이다. 그림자 갱신 방식은 트랜잭션이 커미트될 때 변경된 데이터를 MMDB로 복사하는 과정이 필요하므로 트랜잭션 응답시간이 지연된다는 단점이 있는 반면에, redo 로그만을 기록하면 되므로 로그의 양이 절반으로 감소하여 복구 시간이 단축되는 장점이 있다. 일반적으로 DRDB에서는 직접 갱신 방식이 우수한 것으로 알려져있으나, MMDB의 경우에는 데이터 변경이 주기억장치에서 이루어지므로 상황이 다르다. 그림자 갱신 방식이 직접 갱신 방식보다 MMDB에 있어 보다 좋은 성능을 보인다는 연구 결과도 나온 바 있다[5].

[4]는 MMDB에서 데이터베이스 갱신 방식의 성능 비교를 위해 그림자 갱신 방식과 안정기억장치를 사용하는 대표적인 MMDB 시스템인 MARS를 대상으로 성능평가를 실시하였다[1]. 그 결과에 따르면, 정상적인 트랜잭션 처리 환경에서는 직접 갱신 방식이 더 좋은 평균 응답 시간을 보이는 반면, 시스템 파손이 발생한 경우에는 그림자 갱신 방식이 복구 시 적용되어야 할 로그의 양이 작기 때문에 더 빠른 평균 응답 시간을 나타내고 있다. 이 연구는 그림자 갱신 방식을 사용할 경우, 시스템 파손 후 복구 시간이 빠른 반면, 정상적인 경우에는 트랜잭션 처리 시간이 지연되는 문제점이 있다는 것을 보여주고 있다

본 연구에서는 실시간 응용을 위하여 데이터 갱신 방식으로

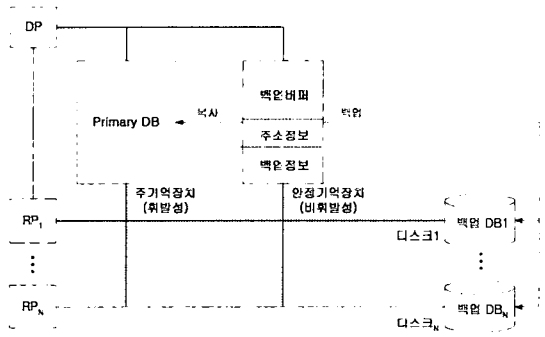
* 본 연구는 두뇌한국21(BK21) 사업의 지원을 받았습.

서 시스템 파손 후 빠른 복구가 가능한 그림자 갱신 방식을 채택하고, 트랜잭션 커미트 과정에서 요구되는, 변경 데이터의 MMDB 복사 과정 때문에 처리 시간이 지연되는 문제점을 해결하기 위한 방안을 제시하고자 한다. 기본 방안은 그림자 영역과 로그 버퍼를 별도로 분리하지 않고, 로깅과 백업 과정을 하나로 통합하는 것이다. 그림자 영역에 기록된 트랜잭션의 AFIM 를 삭제하지 않고, 그대로 redo 로그로 이용하여, 이 로그를 기반으로 백업을 실시한다. 이를 위해 비휘발성이면서도 RAM 과 같이 빠른 접근 속도를 제공하는 FRAM 을 그림자 영역 및 로그 버퍼로 사용한다.

3. 시스템 구성 및 회복 기법

이 절에서는 본 논문에서 제안하는 회복 기법을 위한 MMDB 시스템의 기본 구성을 설명하고, 트랜잭션 수행 방식과 백업 방식, 시스템 복구 과정에 관하여 기술한다.

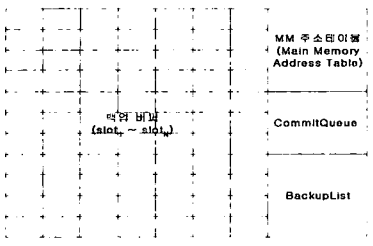
3.1. 시스템 구성



[그림 1] 회복 시스템 구조

본 제안하는 회복 시스템은 [그림 1]과 같이 데이터처리기(DP), 회복처리기(RP), MMDB, 백업 버퍼 및 주소 정보, 백업 정보를 저장하고 있는 안정기억장치, 백업 디스크의 백업 DB 로 구성된다. 백업 지연 시 발생할 수 있는 병목 현상을 방지하기 위해 RP 를 여러 개 두어 동시에 별도의 백업 디스크로 백업을 실시할 수 있다. 정상적인 경우에 DP 는 트랜잭션 처리를 담당하며, RP 는 로그 기반 백업을 담당하나, 시스템 복구과정에서는 DP 와 RP 가 병렬로 복구 작업을 수행한다.

주기억장치는 페이지들로 구성되며, 페이지는 다시 일정 크기의 슬롯들로 분할되고, 모든 페이지들은 동일한 수의 슬롯들을 가지는 슬롯 기반 페이지 방식을 사용한다.



[그림 2] 안정기억장치 구조

안정기억장치는 [그림 2]와 같이 백업 버퍼와 MM 주소 테이블(MMAT), CommitQueue 와 BackupList 로 이루어진 백업 정보 저장 영역으로 구성된다. 백업 버퍼도 주기억장치 페이지 슬롯 크기와 동일한 크기의 슬롯들로 분할된다. MM 주소 테이블은 각 백업 버퍼 슬롯에 저장되어 있는 데이터들의 주기억장치 내의 주소 정보를 (MM pageID, MM slotID)와 같은 형식으로 가지고 있다.

BackupList 는 RP 가 디스크로 백업할 백업 버퍼 슬롯 번호들을 저장하고 있는 리스트로 효율적인 백업을 위하여 주기억장치 페이지 별로 분류되어 주기억장치 페이지 수만큼의 $BackupList_{pageID}$ 가 존재한다. CommitQueue 는 커미트된 트랜잭션이 변경한 슬롯들 중, 아직 BackupList 에 추가되지 않은 슬롯 번호들을 저장하고 있는 FIFO 큐(queue)이다. 시스템 복구 작업과 트랜잭션 처리를 위해 유지되어야 할 정보들로는 이밖에도 현재 수행 중인 트랜잭션이 사용하고 있는 백업 버퍼 영역 내의 슬롯들에 대한 정보를 가지고 있는 TPList 와, 백업버퍼 영역 내의 빈 슬롯들을 관리하는 SFreeList 이 있으며, 이들은 비휘발성인 안정기억장치에 저장될 필요 없이 주기억장치에 저장될 수 있다.

- TPList : 현재 트랜잭션에 의해 변경된 백업 버퍼 슬롯 번호들의 리스트
- SFreeList : 안정기억장치 내의 빈 슬롯 번호들의 리스트
- CommitQueue : 트랜잭션에 의해 커미트된 후, 백업을 기다리는 슬롯번호들의 FIFO 큐로 트랜잭션이 변경한 백업버퍼 슬롯번호들이 CommitQueue 에 추가되어야 커미트가 완료됨.
- BackupList : RP 가 CommitQueue 에서 슬롯번호들을 가져와서 백업 처리를 위해 재구성한 리스트로 주기억장치 페이지별로 하나씩 존재.

회복 정보의 관리에 있어서 트랜잭션들의 동시 수행을 지원한다면 페이지에 대한 잠금을 설정해야 하므로 이로 인해 오버헤드가 발생할 수 있다. MMDB 에서는 주기억장치 접근 속도가 디스크 접근 속도보다 훨씬 빠르므로 잠금을 획득하기 위한 경쟁이 DRDB 보다 낮은 특성을 보인다. 이로 인해 잠금의 단위를 전체 데이터베이스로 하여 트랜잭션들을 순차적으로 수행함으로써 동시성 제어를 위한 비용을 제거하는 것이 오히려 전체 MMDB 시스템의 성능 향상을 가져올 수 있다는 것이 여러 연구들을 통해 증명된 바 있다[5, 7]. 본 논문에서는 트랜잭션의 순차수행을 가정하여 동시성 제어로 인해 발생할 수 있는 오버헤드를 없애고자 한다.

3.2. 트랜잭션 처리

본 논문의 회복 기법을 지원하기 위한 트랜잭션의 수행 알고리즘은 다음과 같다.

1. 트랜잭션 시작
2. 트랜잭션의 각 연산 op_i 에 대하여,
   a. op_i 가 $(page_k, mslot_l)$에 대한 읽기 연산인 경우,
      - TPList 에 존재하는 각 백업버퍼 SlotID 에 대하여, $(page_k, mslot_l)$의 MM 주소를 가지는 엔트리가 있는지 찾음
        - 엔트리가 있고, 백업버퍼 SlotID 가 $sslot_p$ 라면, $sslot_p$ 에 해당하는 백업버퍼 슬롯에서 데이터를 읽음.
        - 그렇지 않다면, $mslot_l$ 에 해당하는 MMDB 슬롯에서 데이터를 읽음.
   b. op_i 가 $(page_k, mslot_l)$에 대한 쓰기 연산인 경우,
      - TPList 에 존재하는 각 백업버퍼 SlotID 에 대하여, $(page_k, mslot_l)$의 MM address 를 가지는 엔트리가 있는지 찾음
        - 엔트리가 있고, 백업버퍼 SlotID 가 $sslot_p$ 라면, $sslot_p$ 에 해당하는 백업버퍼 슬롯의 데이터를 변경.
        - 그렇지 않다면,
          - SFreeList 에서 백업버퍼 슬롯 $sslot_q$ 을 하나 할당 받음
          - MMAT 의 q 번째 엔트리의 (MM pageID, MM slotID) 필드

를 (*k, l*)로 변경.
- $sslot_q$에 데이터를 기록.

3. 트랜잭션 커미트 과정
   a. 트랜잭션 철회(abort) 시,
      - TPList 에 존재하는 각 SM SlotID 를 SFreeList 에 추가
   b. 트랜잭션 커미트 시,
      - TPList 에 존재하는 각 백업버퍼 SlotID 에 대하여,
        ■ 백업버퍼 SlotID 를 CommitQueue 에 Add
        ■ ($page_m$, $mslot_n$) 의 AFIM 인 백업버퍼 SlotID 에 해당하는 슬롯 내용을 MMDB 의 ($page_m$, $mslot_n$) 슬롯에 복사
4. 트랜잭션 종료

3.3. 백업 관리

백업은 RP 에 의해서 수행된다. 시스템 성능을 높이기 위한 방안으로 RP 와 백업 디스크를 여러 개 두어, 각 RP 가 $BackupList_{pageID}$ 들을 분할 담당하여 별도의 백업 디스크로 독립적으로 백업을 실시할 수 있다.

1. RP 가 CommitQueue 에서 하나의 Slot ID 를 꺼냄(dequeue)
2. Slot ID 에 해당하는 (*MM pageID,mslotID*)를 MMAT 에서 얻음.
3. (*mslotID, sslotID*)를 $BackupList_{pageID}$에 추가
   - *mslotID* 가 이미 $BackupList_{pageID}$ 에 추가되어 있는 슬롯의 *mslotID* 와 같다면, 이전에 추가된 엔트리의 *sslotID* 를 SFreeList 에 추가하고 $BackupList_{pageID}$ 에서 삭제. (가장 최근에 변경된 백업버퍼 슬롯만 남겨놓음)
4. $BackupList_{pageID}$ 중에서 백업을 대기중인 슬롯의 수가 정해진 임계치를 넘어가는 경우, 해당 페이지에 대한 백업을 실시.
   - 백업이 완료되면, $BackupList_{pageID}$ 에 존재하는 백업 버퍼들의 *sslotID* 들을 SFreeList 에 추가.
5. goto 1.

3.4. 복구 관리

데이터베이스 복구는 로그를 적용할 필요가 없으므로 신속히 이루어진다. 아직 백업되지 않은 슬롯들이 시스템 파손 후에도 그대로 안전기억장치에 유지되므로 복구 시 undo 나 redo 연산은 필요치 않다. 따라서 다음과 같은 방법으로 신속하게 트랜잭션 처리를 재개할 수 있다.

1. DP 가 디스크에서 백업 DB 를 페이지별로 MMDB 에 적재
2. RP 는 1 번 단계와 병렬로 CommitQueue 에 남아 있는 엔트리들을 각 MMDB 페이지별 $BackupList_{pageID}$ 에 추가
3. DP 가 MMDB 적재를 완료한, MMDB 페이지들에 대해, RP 가 $BackupList_{pageID}$ 에 존재하는 백업버퍼 슬롯들의 내용을 MMDB 에 적용(1 번과 병렬 수행)
4. 3 번이 완료되면, SFreeList 에 모든 백업버퍼 slotID 들을 추가
5. 트랜잭션 처리 시작

시스템 재시작 후 백업 버퍼에 남아있는 백업되지 않은 페이지들을 MMDB 로 적재하는데 이것은 안정기억장치와 주기억장치간에 이루어지므로 짧은 시간에 처리할 수 있다.

5. 결론 및 향후 연구 방향

회복 관련 작업이 MMDB 성능에 미치는 영향을 최소화하기 위해, 본 논문에서는 안정기억장치를 사용하여 로그 기반 백업을 실시함으로써 트랜잭션 처리 시간을 지연시키지 않으면서도 시스템 파손 시 빠른 복구를 지원하는 회복 기법을 제안하였다.

본 논문에서 제안하는 MMDB 회복 기법은 첫째, 그림자 갱신 방식을 사용하여 undo 로그를 생성하지 않음으로써 로그 양을 경감시켰다. 둘째, 안정기억장치를 백업버퍼로 사용한 로그 기반 백업을 실시하여, 로깅과 체크포인팅 프로세스를 하나로 통합하였다. 따라서 별도의 로그 디스크가 필요없으며, 시스템 복구 시, redo 로그를 디스크에서 적재하여 MMDB 에 적용할 필요 없이 곧바로 안정기억장치에서 MMDB 로 복사하면 되므로, 신속한 트랜잭션 재개가 가능하다. 또한 보다 원활한 로그 기반 백업을 위해 다중 프로세서로 하여금 MMDB 페이지들을 분할하여 해당 페이지들에 대한 백업을 실시할 수도 있도록 하였다.

본 회복기법에서는 백업 버퍼 용도의 안정기억장치를 필요로 하는데, MMDB 의 크기가 커질 경우 트랜잭션 처리에 영향을 주지 않기 위해서는 안정기억장치 필요량이 증대될 수 있다. 그러나, 데이터베이스를 접근하는 80%의 경우가 20%의 데이터에 대해서 이루어진다는 80-20 rule 에 따를 경우, 대부분의 트랜잭션들이 MMDB 의 일부 페이지들에 대해 집중되므로 비교적 작은 안정기억장치만을 사용하더라도 원활한 트랜잭션 처리가 가능하다[3].

앞으로는 본 회복 기법을 기존에 연구된 MMDB 회복 기법들과의 성능 평가를 수행하여 본 회복 기법의 적합성에 대한 연구를 실시할 예정이다.

참고문헌

[1] Margaret H. Eich, "MARS: The Design of a Main Memory Database Machine," In proceedings of the International Workshop on Database Systems (IWDM), Oct. 1987, pp.325-338, Database Machines and Knowledge Base Machines, Kluwer Academic Publishers, 1988, pp. 325-338.

[2] Hector Garcia-Molina, Kenneth Salem, "High Performance Transaction Processing with Memory Resident Data", International Workshop on High Performance Transaction Systems, Paris, Dec. 1987.

[3] J.Gray, B.Good, D.Gawhck, P.Homan and H.Sammer, "One Thousand Transactions Per Second", Tandem Computers, Technical Report 85, Nov. 1984

[4] Le Gruenwald, YuWei Chen, Jing Huang, "Effects of Update Techniques on Main Memory Database System Performance", IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 5, Sep. 1998

[5] Vijay Kumar, Albert Burger, "Performance Measurement of Main Memory Database Recovery Algorithms Based on Update-in-Place and Shadow Approaches", IEEE Transactions on Knowledge and Data Engineering, Vol.4, No.6, Dec. 1992

[6] Juchang Lee, Kihong Kim, Sang K. Cha, "Differential Logging:A Commutative and Associative Logging Scheme for Highly Parallel Main Memory Database", ICDE 2001, pp.173-182

[7] S.H. Han, B.S. Jeon, C.H. Ryu, Y.K. Kim, and S.I. Jin, "Performance Comparison of Sequential Transaction Processing and 2PL-based Concurrency Control in a Main Memory Database System", In *Proc. of Korea Information Processing Society* ('99 KIPS), Vol. 6, No. 1, Apr. 1999, pp. 159-161.

[8] "About FRAM Memory", http://www.ramtron.com