플래시 메모리에서의 회복 방안 고찰

정교성, 배덕호, 김상욱
한양대학교 전자컴퓨터통신공학과
e-mail : purenim@agape.hanyang.ac.kr

A Reconsiderations of Recovery Techniques for DBMS in Flash Memory Environment

Kyo-Sung Jeong, Duck-Ho Bae, Sang-Wook Kim
Dept. of Electronics and Computer Engineering, Hanyang University

요 약

플래시 메모리는 기존 저장 매체와는 달리 읽기 연산보다 쓰기 연산의 수행비용이 매우 크고, 덮어쓰기가 불가능한 특성이 존재한다. 본 논문에서는 플래시 메모리의 고유의 특성들이 기존의 디스크 기반 로그 기반 회복 방안과 그림자 페이지 기반 회복 방안의 성능에 미치는 영향을 분석한다. 더 나아가, 플래시 메모리 환경을 위해 제안된 회복 방안들의 장, 단점을 분석하고, 이를 바탕으로 추후 플래시 메모리 환경을 위한 회복 방안 설계 시 고려해야 할 사항들을 제안한다.

1. 서론

플래시 메모리는 기존의 하드 디스크와는 다른 고유한 특성을 가진다. 첫째, 쓰기 연산은 읽기 연산에 비해 매우 느리다[1,2]. 둘째, in-place update가 불가능하다[2,3]. 따라서 데이터를 갱신하기 위해서는 기존 데이터가 저장된 페이지를 무효화(invalid)하고, 갱신된 데이터를 새로운 페이지에 저장하여야 한다. 셋째, 수행 속도가 매우 느린 소거 연산이 존재한다[4]. 소거 연산은 무효화된 페이지를 재사용하기 위해 수행되는 연산으로 빈번한 쓰기 연산에 의해 발생한다. 따라서 플래시 메모리 환경에서는 *쓰기 연산 횟수를 줄이는 것*이 매우 중요하다.

최근 플래시 메모리가 대용량화됨에 따라 플래시 메모리를 위한 DBMS 기술들이 연구되고 있다. DBMS 기술에는 인덱싱, 버퍼 관리, 레코드 관리, 동시성 제어, 회복 등이 존재한다. 특히, 회복 기술은 예기치 못한 오류가 발생하여도 최신의 안정된 데이터를 유지할 수 있도록 도와주는 DBMS의 핵심 기술이다[5].

본 논문에서는 플래시 메모리 환경이 기존의 디스크 기반 회복 방안에 미치는 영향력을 분석한다. 이를 통해, 디스크 환경과는 달리 플래시 메모리 환경에서 새롭게 나타나는 회복 방안들의 장, 단점을 도출한다. 더 나아가, 플래시 메모리 환경을 위해 제안된 회복 방안들의 장, 단점을 분석한다. 끝으로, 위의 분석들을 바탕으로 추후 플래시 메모리 환경을 위한 회복 방안 설계 시 고려해야 할 요소들을 제안한다.

2. 플래시 메모리 환경이 회복 방안에 미치는 영향력 분석

예기치 못한 오류의 발생으로부터 최신의 안정된 데이터를 유지하기 위해서는 완료된 트랜잭션이 갱신한 데이터는 반드시 저장 장치에 반영되어 있어야 한다[5]. 기존의 회복 방안은 크게 로그 기반 회복 방안(이하 로그 방안)과 그림자 페이지 기반 회복 방안(이하 그림자 페이지 방안)으로 구분할 수 있다[5,6,7].

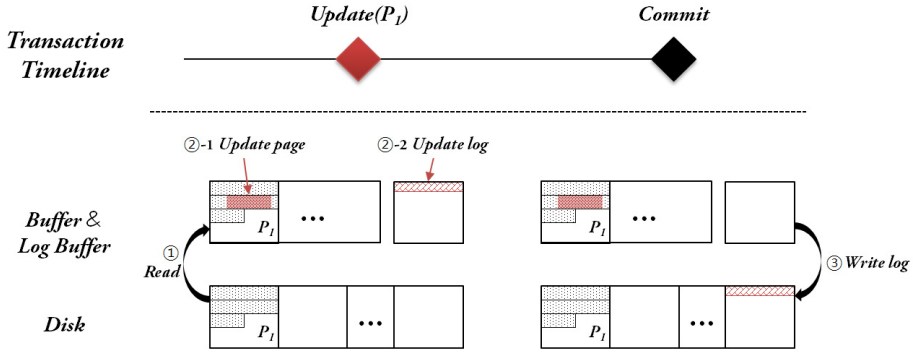
2.1 로그 방안

로그 방안은 완료된 트랜잭션이 갱신한 데이터 정보를 별도의 로그 레코드를 이용하여 저장하는 방안이다[1][5,8]. 그림 1(a)는 로그 방안의 트랜잭션 처리 과정을 나타낸다. 로그 방안은 오류 발생 시, 저장된 로그 레코드를 이용하여 데이터가 저장된 페이지(데이터 페이지)를 최신의 안정된 데이터를 갖도록 재구축하는 방안이다.

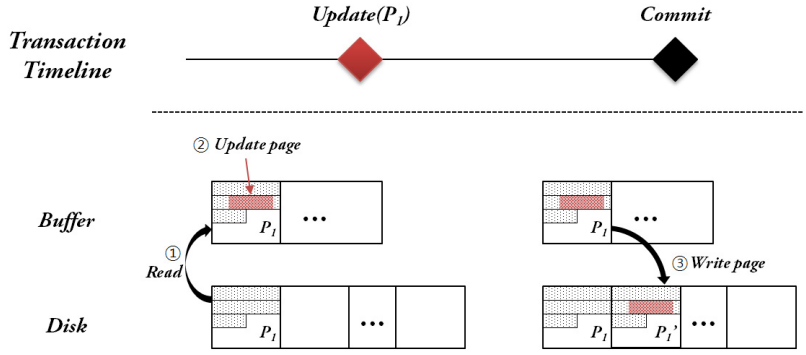
따라서 로그 방안에서는 로그 레코드들은 트랜잭션이 완료되는 시점에 반드시 디스크에 저장되어야 하지만, 데이터 페이지는 회복 방안에 의해 영향을 받지 않는다[8]. 이렇듯, 로그 방안은 사용자 쿼리를 처리하기 위한 일반적인 DBMS 동작에는 전혀 영향을 미치지 않지만, 로그 레코드를 저장하기 위한 추가적인 쓰기 연산이 발생하는 특징이 있다.

그러나 플래시 메모리 환경에서 로그 방안은 추가적인 쓰기 연산으로 인한 오버헤드가 커진다. 또한, in-place update가 안됨에 따라 하나의 레코드만 업데이트하더라도 한 페이지의 쓰기 연산이 발생하게 되고, 이로 인해 공간 낭비가 발생한다. 이는 플래시 메모리의 페이지 소모가 빨라 소거 연산 또한 많이 유발하게 된다.

1) 로그 레코드는 회복에 필요한 모든 갱신 정보를 기록하는 별도의 레코드로, 트랜잭션 수행의 최소 단위인 레코드 단위로 기록된다.



(a) 로그 방안 트랜잭션 처리



(b) 그림자 페이지 방안의 트랜잭션 처리

(그림 1) 기존 회복 방안들의 트랜잭션 처리

2.2 그림자 페이지 방안

그림 1(b)는 그림자 페이지 방안의 트랜잭션 처리 과정을 나타낸다. 그림자 페이지 방안은 회복을 위해 별도의 로그 레코드가 아닌 최신의 안정된 데이터가 저장된 페이지 전체를 유지하는 방안이다[5,9]. 다시 말해 트랜잭션이 완료 시점에 로그 방안은 갱신된 데이터만 기록하는 것이고 그림자 페이지 방안은 갱신된 데이터를 포함하는 페이지 전체를 기록하는 것이다. 따라서 회복을 위해 그림자 페이지 방안은 현재 수정이 발생한 데이터 페이지를 단순히 최신의 그림자 페이지로 교체한다[9].

그림자 페이지 방안은 다음과 같은 근본적인 한계점들이 존재한다. 첫째, 그림자 페이지들을 유지하기 위한 별도의 공간이 필요하다. 둘째, 페이지 단위 잠금(page-level locking)이 필요하다. 그림자 페이지를 생성하기 위해서는 하나의 페이지에 저장된 데이터들은 하나의 트랜잭션만 접근하여야 한다. 이로 인해, DBMS의 트랜잭션 처리 성능이 로그 방안에 비해 전반적으로 낮다. 이러한 한계점으로 인해, 디스크 환경에서는 일반적으로 로그 방안이 널리 사용되었다[5,7].

그러나 플래시 메모리 환경에서 그림자 페이지를 유지하기 위한 오버헤드가 줄어들게 된다. 이는 in-place update가 불가능한 플래시 메모리 특성 상, 수정이 발생한 페이지는 다른 공간에 저장하여야 한다. 따라서 디스크 환경과 달리 수정이 발생하여도 그림자 페이지가 그대로 존재하게 된다. 이렇듯, 플래시 메모리 환경에서는 그림자 페이지 방안이 갖는 이점이 존재한다.

3. 플래시 메모리 환경을 위한 회복 방안 분석

3.1 TIPL

Transactional In-Page Logging(TIPL)은 대표적인 플래시 메모리 환경에서의 로그 방안이다[1,6]. 그러나 플래시 메모리에서 발생하는 로그 방안의 문제점을 해결하기보다는 플래시 메모리의 특성을 고려하여 일반적인 DBMS 동작 시 쓰기와 소거 연산 횟수를 줄이는데 초점을 맞추었다[1,6].

TIPL은 다음과 같은 두 가지 제약 조건이 있다. 첫째, 부분 쓰기(partial write)가 가능한 single-level cell(SLC) 플래시 메모리를 사용하여야 한다. 둘째, 데이터가 저장될 페이지의 물리적 위치를 TIPL이 직접 관리할 수 있어야 한다.

3.2 Flag Commit

Flag Commit은 플래시 메모리 환경을 위한 그림자 페이지 방안으로, 페이지마다 메타 데이터 공간을 확보하고 트랜잭션이 완료되었을 때 부분 쓰기를 사용하여 메타 데이터를 변경함으로써 그림자 페이지로 전환시키는 방식을

사용한다[7]. 이를 통해, 그림자 페이지 관리 오버헤드 및 생성되는 무효 페이지 수를 줄일 수 있다.

Flag Commit 또한 다음과 같은 두 가지 제약 조건이 있다. 첫째, 부분 쓰기가 가능한 SLC 플래시 메모리를 사용하여야 한다. 둘째, 효율적인 그림자 페이지 관리를 위해 페이지 매핑 테이블의 정보를 Flag Commit 방안이 관리할 수 있어야 한다.

4. 플래시 메모리 환경에서의 회복 방안 연구 시 고려해야할 요소

본 장에서는 기존의 회복 방안들에 대한 분석을 기반으로 향후 플래시 메모리 환경을 위한 회복 방안 연구 시 고려해야할 요소들을 제안한다.

- **플래시 메모리의 물리적 특성 고려**

  플래시 메모리는 디스크와 다른 고유한 물리적 특성을 가진다. 따라서 플래시 메모리 환경을 위한 회복 방안은 반드시 플래시 메모리의 특성을 고려하여 효율적인 동작을 수행하여야 한다.

- **플래시 메모리 종류와 무관한 회복 방안**

  플래시 메모리는 물리적 구조에 따라 SLC와 MLC로 구분할 수 있다. 최근 플래시 메모리 환경을 위해 제안된 회복 방안들은 부분 쓰기를 사용한다. 그러나 부분 연산은 SLC 플래시 메모리에서만 사용 가능하다. 최근, 쉬운 대용량화와 경제성으로 인해 MLC 메모리가 널리 사용되고 있다. 따라서 플래시 메모리 환경을 위한 회복 방안은 SLC에서만 가능한 부분 쓰기를 이용하지 않고도 효율적으로 동작하여야 한다.

- **FTL과 독립적인 회복 방안**

  Flash Translation Layer(FTL)은 파일 시스템의 논리적 페이지 주소를 실제 플래시 메모리에 저장된 물리적 페이지와 매핑 시켜주는 역할을 한다[3,7]. FTL은 현재 플래시 메모리를 기반으로 한 디스크인 SSD에 기본적으로 내장되어 있다. 따라서 범용성을 위해서는 회복 방안은 데이터가 저장될 위치를 직접 관리하지 않고 FTL과 독립적으로 수행하여야 한다.

- **일반적인 DBMS 동작 성능을 고려한 회복 방안**

  회복 방안은 일반적인 DBMS 동작에도 많은 영향을 미친다. 따라서 회복 관련 동작들을 효율적으로 수행할 뿐 아니라, 일반적인 DBMS 동작도 효율적으로 수행하여야 한다. 그림자 페이지 방안의 경우, 회복 관련 동작에 있어 플래시 메모리 환경에서 효율적이나, 페이지 단위 잠금 및 그림자 페이지 생성을 위한 강제적인 버퍼 flush 전략으로 인해 일반적인 DBMS 동작의 성능이 좋지 않은 문제점이 있다.

5. 결론

본 논문에서는 기존의 로그 방안과 그림자 페이지 방안의 특징들을 분석하고, 해당 방아들이 플래시 메모리 환경에 그대로 적용되었을 때 변화에 대해서 분석하였다. 더 나아가, 플래시 메모리를 위한 회복 방안들의 문제점을 분석하였다. 끝으로, 이를 바탕으로 향후 플래시 메모리에서 회복 방안 연구 시 고려해할 요소들을 제안하였다.

감사의 글

본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업 (NIPA-2013-H0401-13-1001)과 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단 (No. NRF-2012047724)의 지원을 받아 수행되었다.

참고문헌

[1] S. Lee and B. Moon, “Design of Flash-based DBMS: An In-page Logging Approach,“ *ACM SIGMOD*, pp. 55-66, 2007.

[2] E. Gal and S. Toledo, "Algorithms and Data Structures for Flash Memories," *ACM Computing Surveys*, 37 (2), pp. 138-163, 2005.

[3] S. Lee et al., “A log buffer-based flash translation layer using fully-associative section translation,” *ACM Transactions on Embedded Computing Systems,* Vol. 6, No. 3, pp. 1 - 27, 2007.

[4] D. Bae et al, “An Efficient Method for Record Management in Flash Memory Environment,” *JSA*, pp. 221-232, 2012.

[5] R. Ramakrishnan, and J. Gehrke. *Database Management Systems*, McGraw-Hill, 2003.

[6] S. Lee, and B. Moon, “Transactional In-Page Logging for Multiversion Read Consistency and Recovery” *ICDE*, pp. 876-887, 2011.

[7] S. On et al., “Flag Commit: Supporting Efficient Transaction Recovery on Flash-Based DBMSs” *IEEE TKDE*, 2 (9), pp. 1624-1639, 2011.

[8] C. Mohan et al., “ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging” *ACM TODS*, 17 (1), pp. 94 - 162, 1992.

[9] J. Gray et al., “The Recovery Manager of the System R Database Manager,” *ACM Comput. Surv.*, 13 (2), pp. 223 - 242, 1981.