한국항행학회 논문지 제14권 제5호 2010년 10월

동적 로그 페이지 할당을 이용한 플래시-고려 DBMS의 스토리지 관리 기법

Flash-Conscious Storage Management Method for DBMS using Dynamic Log Page Allocation

송석일*, 길기정*, 최길성**

Seok-Il Song*, Ki-Jeong Khil* and Kil-Seong Choi**

요 약

NAND 플래시 메모리는 높은 입출력 성능, 내장애성, 저전력 소모 등 여러 가지 장점을 가지고 있어서 하드디스크를 대체할 수 있는 새로운 저장 장치로 관심을 받고 있다. 전통적인 DBMS들은 FTL (Flash Translation Layer)를 이용하면 전혀 수정 없이 플래시 메모리 위에서 동작한다. 그러나, 대부분의 FTL은 DBMS가 아닌 파일 시스템에 최적화 되어있다. 또한, 전통적인 DBMS는 플래시 메모리의 특징 (erase-before-write) 을 고려하지 않고 있다. 이 논문에서는 플래시 메모리를 이차 저장장치로 사용하는 DBMS를 위한 플래시 고려하는 스토리지 시스템을 제안한다. 제안하는 플래시 고려 스토리지 시스템은 비용이 높은 변경 연산을 피하기 위해 로그 레코드를 이용한다. 마지막으로, 실험을 통해서 제안하는 스토리지 시스템이 기존에 제안된 플래시를 고려하는 DBMS에 비해 우수함을 보인다.

Abstract

Due to advantages of NAND flash memory such as non-volatility, low access latency, low energy consumption, light weight, small size and shock resistance, it has become a better alternative over traditional magnetic disk drives, and has been widely used. Traditional DBMSs including mobile DBMSs may run on flash memory without any modification by using Flash Translation Layer (FTL), which emulates a random access block device to hide the characteristics of flash memory such as “erase-before-update”. However, most existing FTLs are optimized for file systems, not for DBMSs, and traditional DBMSs are not aware of them. Also, traditional DBMSs do not consider the characteristics of flash memory. In this paper, we propose a flash-conscious storage system for DBMSs that utilizes flash memory as a main storage medium, and carefully put the characteristics of flash memory into considerations. The proposed flash-conscious storage system exploits log records to avoid costly update operations. It is shown that the proposed storage system outperforms the state

Key words : DDR-SSD, Software RAID, RMW

I. 서 론

NAND 플래시 메모리기술의 발전은 NAND 플래시 기반의 스토리지 장치를 전통적인 하드디스크 드라이브에 대한 대안으로 만들고 있다. 플래시 메모리는 비휘발성, 낮은 접근 지연, 저전력 소모, 소형, 저 소음, 빠른 읽기 성능 및 내장애성 등과 같은 다양한 장점들을 가지고 있다 [1, 2, 3]. 여러 장점들 때문

* 충주대학교 컴퓨터공학과
** 동아방송예술대학 미디어기술학부 방송통신과
· 제1저자 (First Author) : 송석일
· 투고일자 : 2010년 10월 14일
· 심사(수정)일자 : 2010년 10월 14일 (수정일자 : 2010년 10월 23일)
· 게재일자 : 2010년 10월 30일

에 플래시 기반의 스토리지 장치들은 모바일 기기들의 주 스토리지로 사용되어 왔고, 최근에는 플래시 기반의 SSD (Solid State Disk) 들이 랩탑, 데스크탑, 엔터프라이즈 서버의 입출력 성능의 향상과 전력 소모량을 줄이기 위해 2차 저장장치로 사용되고 있다.

플래시 메모리는 전통적인 하드디스크와 구분되는 특성을 가지고 있다. 첫 번째, 기계적인 헤드의 움직임이 불필요한 플래시 메모리의 I/O 성능이 하드디스크에 비해 훨씬 우수하다. 특히, 랜덤 읽기 연산이 순차 읽기 연산만큼 빠르다. 두 번째, 플래시 메모리는 비대칭 읽기 및 쓰기 속도를 가지고 있다. 메모리 셀에 전하를 삽입하는 것이 상태를 읽는 시간보다 길기 때문에, 2KByte 에 대한 읽기 시간이 일반적으로 80us 이며 같은 양에 대한 쓰기 시간은 200us 이다. 세 번째, 플래시 메모리는 섹터에 대한 변경을 하기 위해서는 반드시 데이터가 포함된 블록을 먼저 삭제해야 한다. 삭제를 위해서는 변경하는 섹터가 아닌 다른 섹터들은 다른 블록들로 옮겨져야 한다. 이로 인해, 삭제 연산은 많은 I/O 연산들을 일으키게 되며 전체적인 성능을 저하시킨다. 마지막으로, 플래시 블록은 제한된 횟수에 대해서만 삭제가 가능하다. 따라서, 잦은 블록 삭제는 플래시 메모리의 지속시간을 줄이게 된다.

이상의 플래시 메모리의 특성들을 살펴 볼 때, 플래시 메모리는 OLAP, DSS, 데이터 웨어하우스 등과 같은 읽기 집중적인 데이터베이스 응용들에 적합하다. 반면에 플래시 메모리는 OLTP, 이동객체 응용등과 같은 변경 집중적인 데이터베이스 응용에는 적합하지 않다. 이러한 응용들은 섹터크기 보다 작은 작은크기 또는 중간크기의 임의 쓰기 연산들에 대한 처리를 요구한다[2]. 한번 작은 데이터가 플래시 섹터에 기록되면, 섹터의 사용되지 않은 영역은 섹터를 포함하는 블록이 삭제될 때 까지는 사용되지 못한다. 이것은 삭제 연산의 횟수를 증가시키게 되고 쓰기 성능을 크게 저하시킨다.

FTL (Flash Translation Layer) [4] 는 플래시 메모리의 특성을 감추어서 플래시 메모리가 임의 접근 블록 장치처럼 동작하게 해주는 소프트웨어이다. FTL을 이용하면 기존의 DBMS들이 전혀 수정없이 플래시 메모리 위에서 동작한다. 그러나, FTL은 작은 크기에서 중간크기의 임의 쓰기 연산들을 효과적으로 처리할수 있도록 최적화 되어 있지 않다. 따라서, 이 접근법은 플래시 메모리로부터 최대의 성능을 이끌어 내는데 어려움이 있다. 일부 연구 [2, 5]에서 플래시를 고려하는 스토리지 관리 기법을 제안하여 이 문제를 해결하려 하였다.

조사한 바에 따르면, IPL (In Page Logging)이 가장 잘 알려진 최신의 DBMS용 플래시 고려 스토리지 관리 기법이다. IPL은 플래시 메모리의 삭제단위인 블록의 한 페이지를 그 블록의 변경 로그를 저장하기 위한 용도로 예약한다. 데이터 페이지가 변경이 될 때, 이 변경에 대한 로그는 메인 메모리의 로그 섹터에 먼저 저장이 된다. 데이터 페이지가 버퍼 관리 정책에 따라서 교체되거나, 로그 섹터가 차게 되면, 로그 섹터에 대한 변경 로그는 대응하는 로그 페이지로 플러시 된다. 읽기 연산을 처리하기 위해서, IPL은 데이터 페이지와 이와 대응하는 로그 페이지를 읽고 결합하여 데이터 페이지에 대한 최신 버전을 만들어 낸다. 블록의 로그 페이지가 차게 되면, IPL은 병합 연산을 수행한다. 이때, 로그 페이지의 모든 로그 레코드들은 블록의 모든 데이터 페이지들과 병합이 된다. 그리고, 병합된 데이터 페이지들은 새로운 블록으로 이동한다.

IPL이 여러 장점이 있음에도 불구하고, 잦은 변경이 발생되는 핫-페이지들에 대한 처리가 유연하지 못한 단점이 있다. IPL에서는 한 블록에 예약되어 있는 로그 페이지들의 개수가 고정되어 있다. 따라서, 핫-페이지들을 포함하고 있는 블록은 상대적으로 많은 로그 레코드들을 만들어 낸다. 이것은 블록에 대한 잦은 병합 연산으로 이어지게 된다. 반면에 변경이 거의 발생하지 않는 페이지들을 가진 블록은 로그 페이지들의 활용도가 떨어지게 될 것이다.

이 논문에서는 DBMS를 위한 새로운 플래시 고려 스토리지 관리 기법을 제안한다. 제안하는 방법에서는 각 블록이 포함하고 있는 데이터 페이지들의 변경 정도에 따라서 블록에 할당하는 로그 페이지들의 개수를 달리 한다. 또한, 페이지 수준의 매핑 테이블을 사용하여 변경된 데이터 페이지에 대해서 만 변경연산을 수행한다. 이 논문의 구성은 다음과 같다. 2절에서는 제안하는 플래시 고려 스토리지 관리 기법

에 대해 설명한다. 3절 에서는 제안하는 기법과 기존 기법을 실험을 통해서 비교한 결과를 제시한다. 4절에서는 관련된 연구들을 요약하고, 5장에서 결론을 맺는다.

Ⅱ. 제안하는 플래시 고려 스토리지 관리 기법

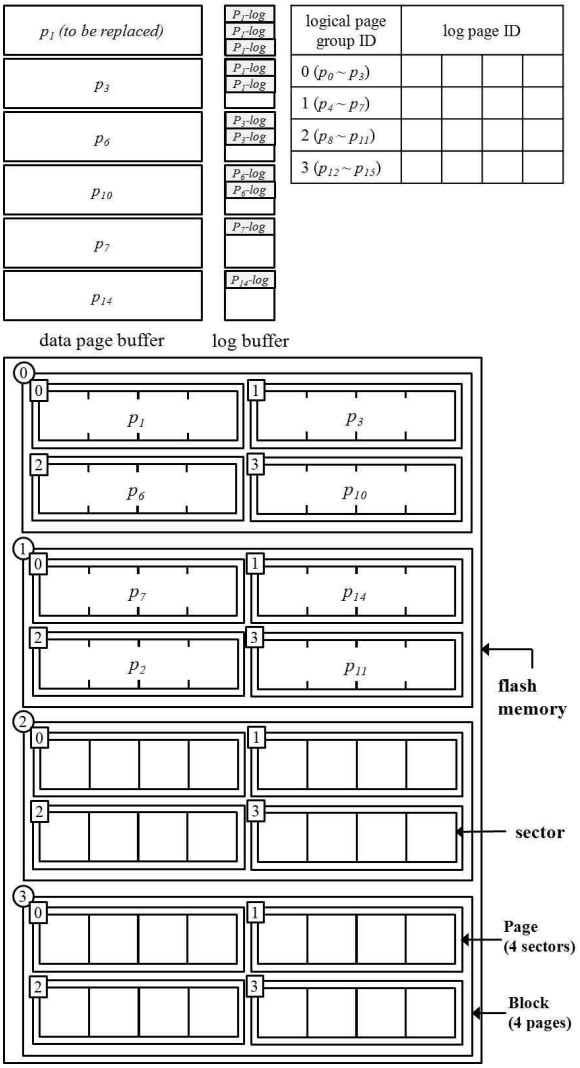

그림 1. 제안하는 스토리지 관리기법의 구조
Fig. 1. Architecture of proposed storage management method

그림 1에서 제안하는 플래시 고려 스토리지 관리 기법의 전체적인 구조를 보여주고 있다. 제안하는 스토리지 관리 기법은 IPL과 유사한 구조를 가지고 있다. 데이터 페이지 버퍼는 고정된 크기의 메인메모리로 플래시 메모리의 데이터 페이지들을 보관한다. 로그 버퍼역시 고정된 크기의 메인 메모리로서 변경된 로그를 저장한다. 로그 버퍼는 플래시 메모리의 섹터 크기과 같은 크기를 갖는 로그 섹터들로 구성이 된다. 이 그림에서 로그 버퍼에 포함된 로그 섹터의 개수는 데이터 페이지 버퍼의 데이터 페이지 개수와 동일하다. IPL의 경우, 데이터 페이지 버퍼에 포함된 데이터 페이지 하나에 로그 섹터를 하나씩 할당 한다. 하지만, 제안하는 방법에서는 로그 섹터의 개수는 사용자에 의해 주어지고, 버퍼의 데이터 페이지는 하나 또는 그 이상의 로그 섹터를 가질 수 있다.

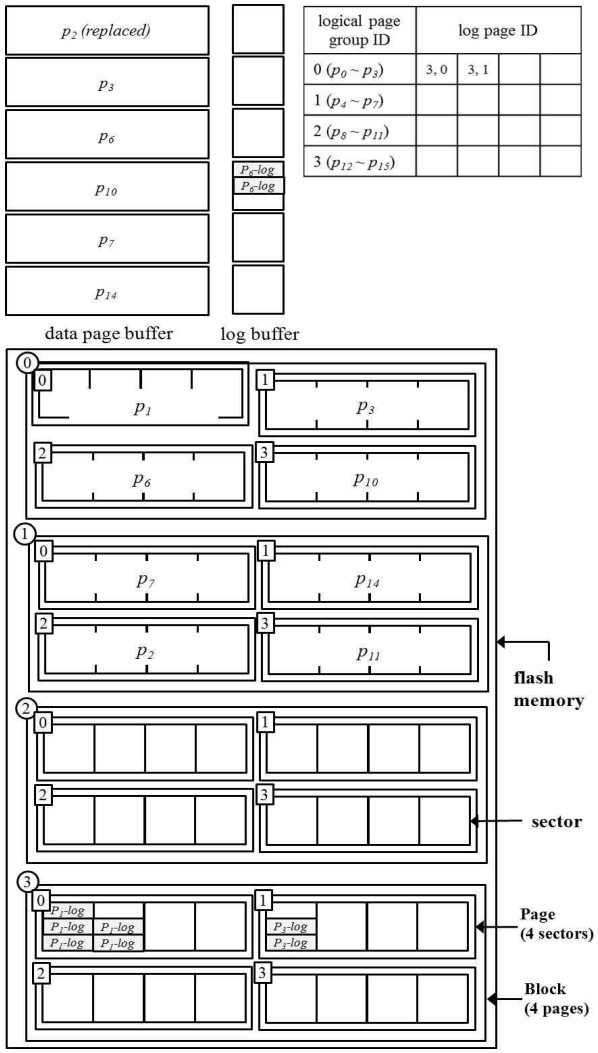

그림 2. 로그 버퍼의 로그레코드들을 로그 페이지로 플러시
Fig. 2. Flush log records in log buffer to log pages in flash memory

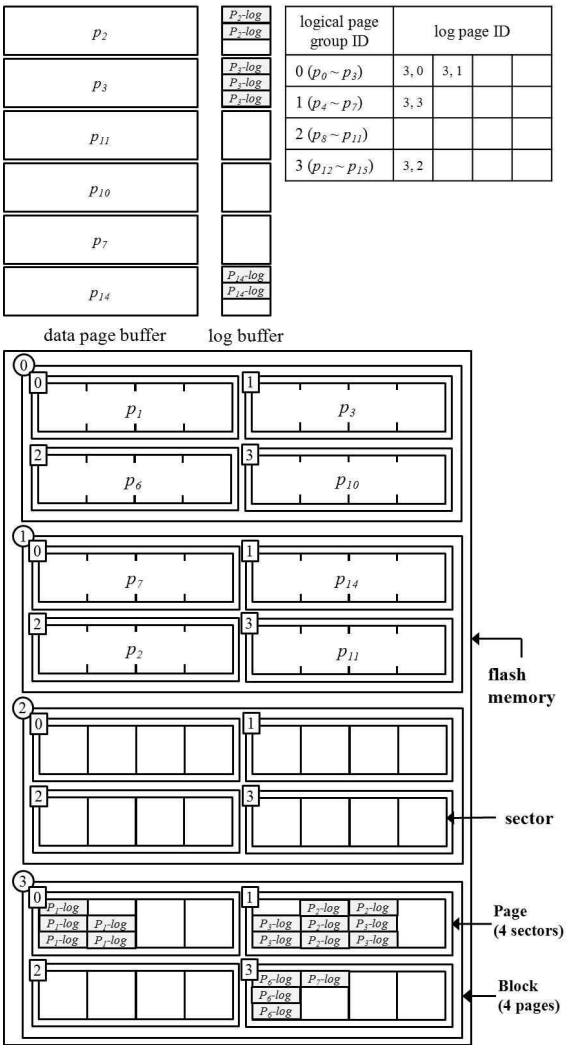

그림 3. 로그 페이지의 병합 연산 수행
Fig. 3. Log page merge operation

그림 1~4에서 pi 는 논리적인 페이지 ID를 나타내고, (블록 번호, 페이지 번호)의 쌍은 물리적인 페이지 ID를 나타낸다. 플래시 메모리는 4개의 블록을 포함하고 있고, 각 블록은 다시 4개의 페이지를 포함하고 있으며, 각 페이지는 4개의 섹터를 포함하고 있다. 논리적인 페이지들은 논리적인 페이지 ID에 의해 그룹이 지어지며 각 논리적 페이지 그룹에는 4개의 논리 페이지들이 포함된다.

제안하는 방법은 매핑 테이블을 가지고 있다. 매핑 테이블은 논리적인 페이지 그룹과 대응하는 로그 페이지들간에 대한 사상 정보를 가지고 있다. 하나 이상의 로그 페이지가 각 논리 페이지 그룹에 할당될 수 있다. 어떤 논리 페이지 그룹이 포함하고 있는 페이지들에 잦은 변경이 발생한다면, 그 그룹은 상대적으로 많은 로그 레코드들을 만들어 내게 된다. 이러한 논리 페이지 그룹에는 두 개 이상의 로그 페이지들을 할당할 수 있다. 논리 페이지 그룹에 로그 페이지들을 할당할때는 로그 버퍼에 포함된 로그 레코드들의 총수에 대한 해당 논리 페이지 그룹의 로그레코드들의 수의 비율을 구한다. 이때 비율이 어떤 기준을 초과하게 되면, 두 개 이상의 로그 페이지들을 할당하고 그렇지 않은 경우에는 하나의 로그 페이지를 할당한다. 그림 1 ~ 4에서 예를 보여주고 있다. 이 예에서는 각 논리 페이지 그룹에 하나나 두 개 이상의 로그 페이지들을 할당한다. 여기서 기준은 50%로 하였다.

로그 버퍼가 차게 되거나 데이터 페이지 버퍼의 데이터 페이지가 교체가 될 때에는 관련된 로그 레코드들이 플래시 메모리의 로그 페이지들에 기록된다. 그림 1에서 데이터페이지 p1이 교체 대상으로 선정이 되었다. 할당할 로그 페이지 개수를 계산하기 위해서 p1이 속해 있는 논리 페이지 그룹의 총 로그 섹터의 수를 구한다. 해당 논리 페이지 그룹의 로그 섹터의 수는 3 이며, 전체 로그 섹터의 수 6의 50% 에 해당한다. 이에 따라, 이 예에서는 플래시 메모리의 논리 페이지 그룹에 2개의 로그 페이지를 할당한다. 그리고 나서, 논리 페이지 그룹의 p0, p1 페이지의 로그 레코드들과 p2, p3의 로그 페이지들을 물리적 페이지인 (3, 0) 과 (3. 1) 에 각각 기록한다.

읽기 연산은 최대 두 개의 물리적 페이지를 읽어서 최신 버전의 데이터 페이지를 만들어 낸다. 그림 2에서, 논리 페이지 그룹 0 (p0 ~ p3) 는 두 개의 로그 페이지들을 가지고 있다. 데이터 페이지 p1을 플래시 메모리로부터 읽기 위해서는 매핑 테이블을 먼저 살펴야 한다. 매핑 테이블에서, p1이 속해 있는 논리 페이지 그룹 0은 두 개의 로그 페이지들 가지고 있다. 이것은 논리 페이지 그룹의 첫 번째 반이 물리 페이지 (0, 3) 과 두 번째 반은 (3,1) 과 연관이 되어 있다는 것을 말한다. p1은 첫 번째 반과 연관이 되어 있으므로, 로그 페이지 (3, 1)와 데이터 페이지를 플래시 메모리에서 읽어서 결합하여 최신 버전의 데이터 페이지를 만들어 낸다. 이 경우에, 읽기 연산은 두

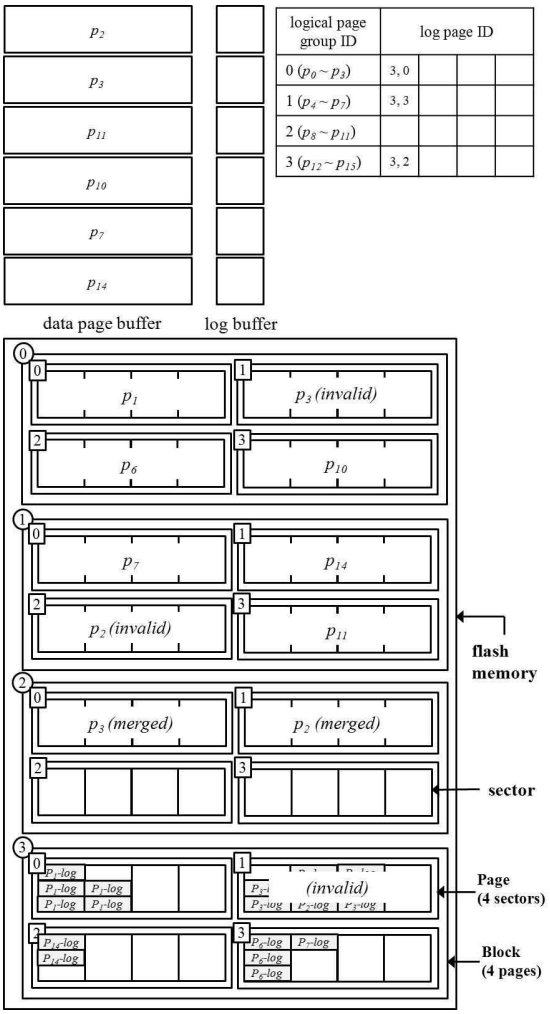

그림 4. 병합 연산의 완료

Fig. 4. Completion of merge operation

개의 물리 페이지 (하나의 데이터 페이지 와 하나의 로그 페이지)를 읽는다. 로그 페이지와 연관되지 않은 p11을 읽어올 때는 하나의 물리적 페이지 (1, 3)를 읽는다.

병합 연산은 논리 페이지 그룹의 로그 페이지들이 찰 때 수행된다. 그림 3에서, 데이터 페이지 p3이 교체대상으로 선택이 되었고, p3가 소속된 논리 페이지 그룹의 로그 레코드들을 로그 페이지로 쓰려고 한다. 하지만, 논리 페이지 그룹의 로그 페이지들은 로그 레코드들을 수용할수 없다. 따라서, p2, p3를 로그 페이지 (3,1) 과 병합하여야 한다. 그림 4에서처럼, 병합된 p2 와 p3는 새로운 물리 페이지인 (2,0) 과 (2,1) 로 옮겨지고, 기존의 물리 페이지 (0,1) 과 (1,2) 는 “invalid" 상태로 표시된다. 블록의 모든 페이자가 ”invalid" 로 표시되면, 가비지 콜렉터 (Garbage Collector) 가 블록을 삭제한다.

그림 5는 쓰기 연산의 알고리즘을 의사코드로 보여준다. 쓰기 연산의 입력은 로그 레코드 (logRec) 이다. logRec은 lpid, tid, data, size, lsn 으로 구성된다. lpid는 논리 페이지 ID를, tid는 트랜잭션 ID, data 는 이전/이후 데이터 이미지, size는 logRec의 크기, lsn은 로그 순차 번호를 뜻한다. 쓰기 연산은 먼저 데이터 버퍼 (dataBuffer)로부터 해당 데이터 페이지 (page)를 찾는다. 그리고 나서, page가 로그 섹터 (logSectors)를 가지고 있고 logRec을 수용할 수 있는 공간이 있는지 확인한다. logSectors가 있고 여유 공간이 있다면, 쓰기 연산은 logRec을 logSectors에 기록한다. page가 logSectors를 가지고 있지 않다면, 새로운 logSector를 할당하고 logRec을 logSectors에 기록한다. logSector가 logBuffer에 없으면, 쓰기 연산은 논리 페이지 그룹 (lpg)을 선택하고 logRec을 로그 페이지에 기록한다.

Algorithm for write operation

```
Input : logRec

page = find logRec.lpid page in dataBuffer
If (page.logSectors exists and free space in logSectors)
      write logRec to logSectors;
      return;
If (free logSector in logBuffer)
      allocate a new logSector to page.logSectors ;
      write logRec to the logSectors;
      return;
lpg = select logicalPageGroup which has the most logSectors;
if (lpg.logPages exists and free space in lpg.logPages)
      write logRecs of lpg to lpg.logPages;
      return;
else if (lpg.logpages exists but no free space in lpg.logPages)
      merge(lpg.logPages);
      return;

If (number of lpg.logSectors/number of logBuffer's logSectors >
      threshold)
      logPages = allocate more than one log pages;
else
      logPages = allocate one log page;

update mapTable;
write logRecs    of lpg to logPages;
return;
```

그림 5. 쓰기 연산에 대한 의사 코드

Fig. 5. Pseudo code of write operation

lpg에 로그 페이지가 있다면, 쓰기 연산은 lpg의 logRec을 로그 페이지에 기록한다. 그렇지 않으면, 새로운 로그 페이지가 lpg에 할당되고 매핑 테이블 (mapTable)이 변경된다. 로그 페이지를 할당할때는 logBuffer의 logSector 개수에 대한 lpg의 logSector 개수의 비율을 계산하여 어떤 기준을 초과할 때 두 개

이상의 로그 페이지를 할당한다.

그림 6은 병합 연산의 알고리즘을 의사코드로 보여주고 있다. 병합연산의 입력은 논리 페이지 그룹의 로그 페이지들 (logPages)이다. 병합 연산은 logPages에 있는 logRec 들과 변경된 페이지들을 읽고 결합을 수행하여 최신 버전의 데이터 페이지 (newPage)들을 만들어 낸다. newPage는 새로운 물리적 페이지에 쓰이고, 매핑 테이블이 변경된다. 동시에 병합연산은 변경된 페이지들을 “invalid" 로 표시한다. 블록의 모든 페이지들이 "invalid"로 표시되면 가비지 컬렉터가 삭제를 수행한다.

```
Algorithm for merge operation
Input : logPages

mergePages = read logPages and analyze logRecs in
             logPages to find pages to be merged;

for ( each page in mergePages )
    newPage = combine page with associated logRecs
              in logPages;
    allocate a new physical page and write newPage to it
    ;mark page as invalid page;

update mapTable
return;
```

그림 6. 머지 연산의 의사코드
Fig. 6. Pseudo code of merge operation

Ⅲ. 성능 평가

이 논문에서는 제안하는 플래시 고려 스토리지 관리 기법과 IPL을 시뮬레이션을 통해 비교하였다. 조사한바에 따르면, IPL이 가장 최신의 DBMS를 위한 플래시 고려 스토리지 관리기법을 제안하고 있다. 두 방법을 평가하기 위해서 리눅스 플랫폼상에서 C 언어로 시뮬레이터를 구현하였다. 구현한 시뮬레이터는 IPL과 제안하는 방법의 플래시 고려 스토리지 입출력 모듈로서 동작한다. 실험에 사용될 데이터를 만들어 내기 위해서 로그 레코드 발생기를 C 언어로 구현하였다. 생성한 로그 레코드는 모두 500,000 였으며, 로그레코드의 크기는 1 byte에서 2KByte 로 하였다. 생성한 로그레코드의 80% 는 전체 논리 페이지의 20%에서 발생하도록 하였다.

NAND 플래시 메모리를 위한 파라미터 값은 Samsung K9W8G08U1M [6] 에 따라서 설정하였다. 플래시 메모리의 크기는 1GByte 로 하였고, 각 블록은 2Kbyte의 64개 페이지들을 포함하고 있다. 그리고, 각 페이지는 4개의 512Byte 섹터들을 포함하고 있다. 도한, 각 논리 페이지 그룹은 16개의 논리 페이지들을 포함한다.

그림 7 과 8은 IPL과 제안하는 스토리지 관리기법을 섹터 쓰기 횟수와 블록 삭제 횟수 측면에서 비교한 것이다. 그림에서처럼 제안하는 방법이 IPL에 비해 월등히 높은 성능을 보인다. IPL은 매 병합연산시 블록 삭제가 발생하지만, 제안하는 방법이 병합연산시 변경이 발생한 페이지들만을 선택적으로 로그 레코드와 병합하므로 삭제 연산의 횟수가 대폭 줄어들게 된다. 또한, 메인 메모리의 로그섹터의 운용도 IPL과 다르게 하나의 메인메모리 데이터 페이지 버퍼에 다수의 섹터를 할당할수 있도록 하여 섹터를 쓰는 횟수를 줄일 수 있었다. 하지만, 제안하는 방식의 경우 실험이 끝난후에도 플래시 메모리에 “invalid" 표시가 되어 있는 페이지가 많이 남아 있는 것을 볼 수 있었다. 따라서, 보다 많은 로그 레코드를 가지고 실험을 하게 되면 제시된 결과보다 블록 삭제 횟수나 섹터 쓰기 횟수가 다소 증가할 것으로 예상된다.

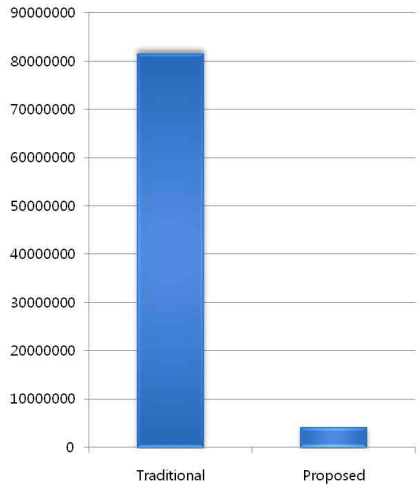

그림 7. 섹터 쓰기 횟수
Fig. 7. Number of sector writes

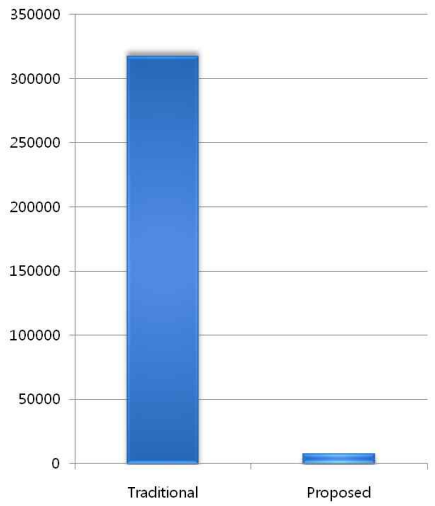

그림 8. 블록 삭제 횟수
Fig. 8. Number of block erases

이 시뮬레이션에서는 페이지 수준의 매핑테이블 관리로 인한 비용은 고려하지 않았다. IPL은 페이지 수준의 매핑 기법을 필요로 하지 않지만, 제안하는 방법에서는 변경된 페이지만 선택적으로 병합하기 위해서는 페이지 수준의 매핑 기법이 필요하다. 페이지 수준, 블록 수준, 혼합 매핑 기법에 대해 많은 기법들이 제안된 바 있다. 최신의 FTL인 DFTL[4] 에 따르면 페이지 수준의 매핑이 가장 유연하고 기타 다른 매핑 기법에 비해 우수하다고 주장하고 있다. 이에 근거해서 페이지 수준의 매핑에 비용이 소요되지만 전체적인 성능에 크게 영향을 미치지 않음을 예측할 수 있다.

Ⅳ. 관련 연구

조사한 바에 따르면, IPL [2]이 가장 최신의 잘 알려진 플래시 고려 DBMS를 제안하고 있다. 이 기법에서는 데이터베이스 페이지와 연관된 모든 로그 레코드들이 데이터베이스 페이지가 속한 블록에 저장된다. 데이터베이스 페이지의 최신 버전은 해당 페이지와 같은 블록에 있는 로그 레코드를 읽어서 만들어 낸다. 데이터베이스 페이지와 이에 대한 로그 레코드들을 같은 블록에 위치시키기 위해서는 각 블록마다 고정된 개수의 페이지가 로그 영역으로 할당된다. 또한, 512 Byte 크기의 메인 메모리 로그 섹터가 버퍼 풀의 각 데이터베이스 페이지에 할당되어 해당 데이터베이스 페이지에 대한 로그 레코드들을 저장한다. 메인 메모리의 로그 섹터가 가득 차거나 데이터베이스 페이지가 교체대상으로 선정이 될 때는 로그 섹터만 플래시 메모리에 기록된다. 변경된 데이터베이스 페이지 자신은 플래시 메모리에 기록되지 않는다.

[6]은 SSD의 하드웨어나 펌웨어를 수정하지 않는 방법을 제안하고 있다. StableBuffer라 불리는 곳에 보관해 놓은 쓰기 요구들중에서 좋은 쓰기 패턴의 쓰기 스트림을 분리해내고, 좋은 쓰기 패턴 (순차 쓰기, 분할된 순차 쓰기, 정렬된 순차 쓰기, 클러스터 쓰기 패턴 등)을 DBMS 응용들을 위한 SSD의 쓰기 성능을 높이는데 활용한다.

Ⅴ. 결 론

이 논문에서는 DBMS를 위한 새로운 플래시 고려 스토리지 관리 기법을 제안하였다. 제안하는 스토리지 관리 기법은 데이터 페이지의 변경 정도에 따라 할당되는 로그 페이지의 수를 동적으로 결정한다. 또한, 변경된 데이터 페이지에 대해서만 병합연산을 수행하여 전체적인 쓰기 연산의 수를 줄인다. 시뮬레이션을 통해서, 쓰기 연산의 회수 관점에서 제안하는 스토리지 관리기법과 IPL의 성능을 비교하였으며, 제안하는 기법이 보다 더 우수함을 보였다. 향후에는 성능평가를 보다 실제적인 환경에서 수행할 예정이다. 이 논문에서는 로그 레코드를 가상으로 생성하여 사용하였는데, 실제 DBMS에서 생성하는 로그 레코드를 이용한 실험을 수행한다. 또한, 페이지 수준의 매핑 기법이 제안하는 방법에 어떠한 영향을 미치는가에 대해서도 추가 연구에서 분석한다.

감사의 글

이 논문은 2008년도 충주대학교 해외파견연구교수지원사업을 통해 수행한 연구임

참 고 문 헌

[1] A. Birrell, M. Isard, C. Thacker and T. Wobber, “A Design for High-Performance Flash Disks,” *ACM SIGOPS Operating Systems Review archive, vol. 41, no. 2, pp. 88-93,* April 2007.

[2] S. Lee and B. Moon, “Design of Flash-based DBMS: an In-page Logging Approach,” *SIGMOD, pp. 55-56,* 2007.

[3] D. Myers, “On the Use of NAND Flash Memory in High-Performance Relational Databases,” *Department of Electrical Engineering and Computer Science, MIT, Master Thesis*

[4] A. Gupta, Y. Kim and B. Urgaonkar, “DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings,” *ASPLOS, pp. 229-240*, 2009

[5] Y. Li, “Optimize write performance for DBMS on Solid State Drive,” *HKBU Computer Science Department Post Graduate Symposium,* 2009.

[6] K9W8G081M/K9K4G08U0M Flash Memory Datasheet, Samsung Electronics. 512M x 8Bit / 1G x 8Bit NAND Flash Memory.

길 기 정 (丁道令)

2007년 3월 ~ 현재 : 충주대학교 컴퓨터공학과 학사과정
관심분야 : 데이터베이스, 스토리지 시스템, 모바일 프로그램 등

최 길 성 (丁道令)

1988년 2월 : 대전산업대학교 전자계산학과(이학사)
1992년 2월 : 수원대학교 전자계산학과(이학석사)
1999년 2월 : 충북대학교 정보통신공학과(공학박사)
1999년 ~ 현재 : 동아방송예술대학 방송통신과 교수
관심분야 :데이터베이스, 색인구조, 스토리지 시스템 등

송 석 일 (丁道令)

1998년 2월 : 충북대학교 정보통신공학과(공학사)
2000년 2월 : 충북대학교 정보통신공학과(공학석사)
2003년 2월 : 충북대학교 정보통신공학과(공학박사)
2003년 7월 ~ 현재 : 충주대학교 컴퓨터공학과 부교수
관심분야 : 데이터베이스, 센서 네트워크, 스토리지 시스템등Storage&Server Virtualization