점진적 맵 업데이트를 위한 모바일 DBMS의 플래시메모리 페이지 관리 기법

Flash-aware Page Management Policy of the Mobile DBMS for Incremental Map Update

민 경 욱* 최 정 단** 김 주 완***
Kyoung Wook Min Jeong Dan Choi Ju Wan Kim

요 약 최근 모바일 디바이스에서 대용량 데이터 저장/관리를 위해 모바일 DBMS를 사용하려는 추세이며 특히 내비게이션 응용과 같이 대용량 맵 데이터의 저장/관리를 위한 모바일 DBMS의 저장구조 및 질의처리 방법에 대한 연구가 수행되었다. 무작위 데이터 접근(읽기/쓰기/변경) 질의가 대부분인 DBMS의 저장매체로 플래시메모리를 사용할 경우 성능이 저하된다. 그 이유는 플래시메모리는 특성상 순차적인 데이터 기록에는 성능이 좋지만 무작위 데이터 기록에는 성능이 나쁘다. 따라서 플래시메모리를 저장매체로 사용하는 모바일 DBMS의 경우 기존과 다른 저장 및 질의처리 기법이 필요하다. 이에 본 논문에서는 무작위 데이터 업데이트의 성능을 향상시키기 위한 DBMS의 페이지 관리 기법을 연구하였고 이를 점진적 맵 업데이트를 지원하는 내비게이션용 모바일 DBMS에 적용하여 실험하였고 성능을 검증하였다.

키워드 : 모바일 DBMS, 플래시메모리, 모바일 내비게이션, 내비게이션 맵 업데이트

Abstract Recently the mobile DBMS (Database Management System) is popular to store and manage large data in a mobile device. Especially, the research and development about mobile storage structure and querying method for navigation map data in a mobile device have been performed. The performance of the mobile DBMS in which random data accesses are most queries if the NAND flash memory is used as storage media of the DBMS is degraded. The reason is that the performance of flash memory is good in writing sequentially but bad in writing randomly as the features of the NAND flash memory. So, new storage structure and querying policies of the mobile DBMS are needed in the mobile DBMS in which a flash memory is used as storage media. In this paper, we have studied the policy of the database page management to enhance the performance of the frequent random update and applied this policy to the navigation-specialized mobile DBMS which supports incremental map update. And also we have evaluated the performance of this policy by experiments.

Keywords : Mobile DBMS, Flash Memory, Mobile Navigation, Navigation Map Update

1. 서 론

모바일 디바이스에서는 계산능력과 자원이 풍부해짐에 따라 대용량의 데이터를 관리할 수 있게 되었다. 가장 대표적인 서비스는 내비게이션 서비스이며, 내비게이션 맵 데이터의 크기는 최근 8Gbyte까지 이른다. 모바일 디바이스에서 대용량 데이터를 저장/관리하기 위하여 모바일 DBMS를 이용하는 추세이고 다양한 모바일 DBMS 제품이 존재한다.

모바일 디바이스의 저장매체로 NAND 플래시메모리가 대부분 사용된다. 플래시메모리는 지금까지 저장매체로 사용되던 디스크와는 다른 특성들을 가

† 이 논문은 지식경제부 산업전략기술개발 프로그램 - 10035250, 자동발렛파킹을 위한 센서기반 공간인지 및 자동주행 기술개발 과제 - 의 연구비지원에 의해 수행되었습니다.

* 한국전자통신연구원 융합연구부문 선임연구원 kwmin92@etri.re.kr (교신저자)
** 한국전자통신연구원 융합연구부문 책임연구원 jdchoi@etri.re.kr
*** 한국전자통신연구원 융합연구부문 책임연구원 juwan@etri.re.kr

지고 있으며, 특히, 잦은 무작위 업데이트의 경우 성능이 많이 저하된다. 과거, 모바일 디바이스의 계산능력 및 자원이 부족한 때에는 읽기 전용의 응용이 많았지만, 최근 고성능 스마트 폰의 출시에 따라 대용량 데이터를 처리할 수 있는 응용에서의 모바일 DBMS 사용은 필수적이다. 하지만 앞서 설명한 바와 같이 기존 저장매체인 디스크와 동일한 방식을 가지는 모바일 DBMS의 경우 다양한 응용에서 성능 보장을 할 수 없다.

최근 모바일 DBMS 연구와 관련하여 대용량 맵 데이터의 부분 업데이트가 불가능한 점을 해결하고자 모바일 DBMS에 대한 연구가 일부 수행되었다 [8, 11, 12, 13]. 이 연구는 내비게이션에 특화된 DBMS에 대한 연구이며 기존 모바일 DBMS의 한계를 극복하고자 다양한 접근 방법을 시도하였다. 하지만, 이 연구에서도 플래시메모리에 특화된 데이터 처리 기능을 지원하지 못하고 향후 추가 연구관제로 Flash-aware 기능이 필요함을 강조하고 있다.

따라서, 본 논문의 연구는 플래시메모리의 특성을 반영한 모바일 DBMS의 개발에 관한 것이며, 특히, 데이터 무작위 업데이트의 성능을 향상시키기 위한 데이터베이스 페이지 관리방법에 대한 부분이다. 일반적인 DBMS의 구조에서 페이지를 관리하는 부분은 시스템 레이어 중에서 하위 부분인 저장관리기(Storage Manager)이며 기존 연구 결과물인 [8]의 모바일 DBMS에 직접 반영하여 개발하였고 다양한 실험을 수행하였다. 이 모바일 DBMS는 FUNs (Flash-aware Ubiquitous Navigation System)로 명명하였고 이 이름을 이후에는 사용하도록 한다. 그림 1에서는 FUNs 의 구조를 보이고 있으며 앞서 설명한 DBMS의 저장 관리기에 플래시메모리의 특성을 반영한 페이지 관리 기법을 구현 및 적용하였다. 또한 본 연구에서는 실제 내비게이션 맵 데이터를 이용하여 업데이트 실험을 하고 분석하였으며 기존 모바일 DBMS로 가장 많이 사용되고 있는 SQLite와 성능을 비교하였다.

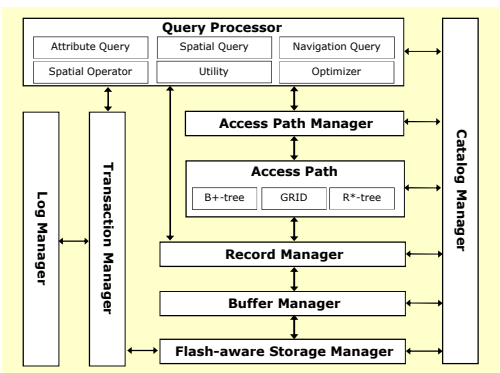


그림 1. 모바일 DBMS(FUNs) 구조

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 관련 연구에 대해서 살펴보고, 3장에서는 본 연구에서 제안하는 플래시메모리 특성을 반영한 페이지 관리기법에 대해서 자세히 기술한다. 그리고 4장에서는 실험을 통하여 본 논문의 기법의 성능에 대해서 분석하여 평가하고 마지막 5장에서는 결론을 맺는다.

2. 관련 연구

지금까지는 모바일 내비게이션에서 부분 맵 데이터만의 변경이 불가능하여 전체 맵 업데이트를 수행해 왔다. 그 이유는 기존 모바일 내비게이션에서의 맵 데이터의 저장 구조는 PSF (Physical Storage Format)를 따르고 있기 때문이었고, 이를 해결하고자 모바일 DBMS를 이용하려는 연구가 수행되었다. 하지만 기존 상용 모바일 DBMS를 이용할 경우 첫 번째로 우리나라 도로를 구성하는 약 100만건에 해당하는 노드/링크 데이터를 테이블에 저장하여 레코드의 논리 ID 질의에 의한 경로검색의 성능을 보장할 수가 없다. 두 번째로는 배경 맵 데이터를 수용하기 위한 공간 데이터 자료형, 공간색인 및 질의처리 기능이 없다. 따라서 이를 해결하고자 [8, 11, 12, 13]에서는 기존 관계형 DBMS의 구조를 따르면서 경로검색 및 공간검색의 성능을 보장할 수 있는 기법들을 제시하였고 다양한 실험을 통해 유용성을 검증하였다.

플래시메모리는 디지털카메라, MP3 플레이어, 스마트 폰과 같은 모바일 기기의 저장매체로 가장 많이 활용되고 있다. 플래시메모리는 일반 하드 디스크와 다른 특징들을 가지고 있다. 우선 하드 디스크는 읽기/쓰기 인터페이스를 제공하지만, 플래시메모리는 읽기/쓰기/소거 인터페이스를 제공한다. 플래시메모리의 구조는 블록과 페이지(또는 섹터)로 구성되어 있으며 대블록 플래시메모리인 경우 하나의 블록에는 64개 섹터로 구성되고 섹터의 크기는 2KB이다. 앞서 살펴본 플래시메모리의 읽기/쓰기/소거 연산에서 읽기/쓰기의 단위는 섹터이고 소거의 단위는 블록이다. 또 다른 특징으로, 플래시메모

리는 디스크와 다르게 덮어쓰기 연산을 하지 못한다. 예를 들어 특정 섹터를 수정하기 위해서는 해당 섹터가 포함된 블록을 소거(erase)하고 쓰기 연산을 수행해야 한다. 이것을 쓰기 전 소거(erase before write) 연산이라고 한다. 따라서 블록 내의 한 섹터를 수정하기 위해서는 그 블록의 내용을 다른 곳에 복사해 두고, 해당 블록을 소거하여 지운 다음, 수정하고자 하는 섹터 및 회피해 두었던 다른 섹터의 내용을 해당 블록으로 복사해야 한다. 이것은 대단히 비효율적인 연산이다. 그리고 이러한 읽기/쓰기/소거 연산의 수행비용은 모두 다르다. 일반적으로 제품 모델마다 다르지만 연산 수행비용은 표 1과 같다. 즉 연산의 비율은 1:2.5:18.7 (80:200:1,500) 이다. 디스크의 경우 읽기와 쓰기의 단위가 동일하고 연산 수행비용이 동일한 점과 비교했을 때, 비용의 차이가 큰 것을 알 수 있다. 마지막 특징으로 디스크는 해당 페이지를 찾기 위해 특정 트랙으로 헤드를 옮기거나 디스크 원판이 회전하여 원하는 섹터를 헤드 아래에 위치시키는 기계적인 시간이 필요하지만 플래시메모리에서는 이런 기계적인 시간이 필요가 없다[3, 5].

표 1. 플래시메모리의 연산속도

	Read	Write	Erase
Samsung SLC NAND 9WAG08U1A 16Gbits	80 μs (2 KB)	200 μs (2 KB)	1.5 ms (128 KB)

지금까지 살펴본 플래시메모리의 특성 중 몇 가지 문제점을 해결하기 위한 소프트웨어 계층이 FTL (Flash Translation Layer)이다. 그림 2에서와 같이 FTL의 가장 큰 역할은 앞서 살펴본 파일시스템의 인터페이스와 플래시메모리 인터페이스의 불일치를 맞춰주는 역할, 즉 플래시메모리 디바이스 자체를 일반 디스크와 같은 장치로 사용할 수 있게 하는 역할을 수행한다. 그리고 하드 디스크와 플래시메모리의 구조가 다르기 때문에 파일시스템에서 요청한 페이지와 물리적인 플래시메모리의 저장 섹터 및 블록을 매핑하고 관리한다[3].

플래시메모리에서 잦은 소거연산은 성능 저하의 원인이 되므로 FTL에서는 이미 기록되어 있는 페이지에 재 기록(overwrite)을 요청한 경우 이를 직접 업데이트(쓰기 전 소거) 하지 않고 임의의 다른 여유 블록의 페이지에 기록하는 방식 즉, 인플레이스(in-place) 업데이트가 아닌 아웃플레이스(out-place) 업데이트 방식을 사용한다. 이와 관련된 기존 알고리즘은 여유영역 기법[9], 복사 블록(Mirror block) 기법[2] 그리고 로그 블록 기법[3]이 많이 사용되고, 최근 [6, 7]에서도 새로운 알고리즘들을 제시하고 있다. 플래시메모리는 순차적(sequential) 쓰기 연산에 대해서는 좋은 성능을 내지만 무작위 쓰기 연산의 경우 성능이 많이 저하된다. 즉, 파일의 복사와 같은 연산은 순차적 업데이트 연산이며 이 경우 성능이 좋지만 DBMS와 같이 잦은 무작위 업데이트 질의 같은 경우, 순차적인 연산에 비해 기존 FTL 알고리즘에서 블록 합병 연산의 발생으로 인해 잦은 소거연산이 수행되어 전체 성능이 저하될 수 있다. DBMS에서는 빠른 레코드 접근을 위하여 색인을 이용하고 데이터의 업데이트에 의해 색인이 함께 업데이트가 되기 때문에 무작위 업데이트 연산이 많이 발생한다. 따라서 무작위 페이지 업데이트 연산을 순차적 페이지로 재 매핑함으로써 성능이 향상 될 수 있다. 그러나 물리적인 페이지 재 매핑 기법[1]은 플래시메모리 장치를 직접 다루어야 하므로 그림 2와 같이 일반적인 제품형태인 FTL이 포함된 디바이스에는 적용할 수 없다.

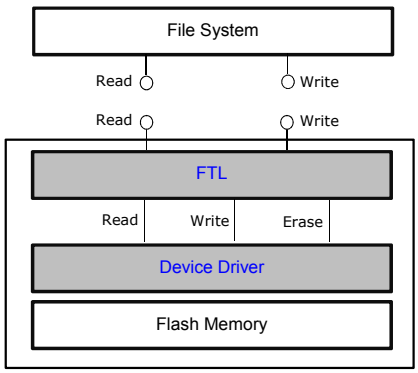


그림 2. 플래시메모리의 구조

3. 플래시메모리에서 업데이트 성능향상을 위한 페이지 관리기법

모바일 디바이스의 저장매체로 플래시메모리가 대부분 사용된다. 플래시메모리의 특징은 앞서 관련 연구에서 살펴본 바와 같이 순차적 기록에 적합한 구조로 되어있다. 지금까지의 모바일 디바이스 응용에서 무작위로 페이지에 쓰기를 하는 응용들은 많지 않았다. 기존 내비게이션에서의 맵 업데이트 방식인 전체 맵 데이터를 업데이트하는 경우 역시 순

차적 기록 방식이다.

순차적 기록방식과 무작위 쓰기 기록방식에 대하여 그림 3에서와 같이 간단한 실험을 하였다. 2KB 크기의 페이지 1,000개를 생성하고 하나의 파일을 생성하여 기록하였다(Task 1). 다음으로 1,000개의 페이지에 순차적으로 다시 한번 기록하였다(Task 2). 그리고 난 뒤, 이번에는 임의의 순서를 1~1,000까지 생성한 뒤 이 순서대로 기록하였다(Task 3). 마지막에는 1,000개의 페이지를 기존 페이지에 기록하지 않고 파일의 끝에 다시 기록하였다(Task 4). 저장매체로는 플래시메모리와 디스크를 이용하였다.

디스크를 저장매체로 사용한 경우 Task 1~4 모두 동일한 성능을 보였다(디스크의 경우 성능의 200배로 표기). 플래시메모리를 저장매체로 사용한 경우 Task 1, 2, 4의 성능은 거의 동일하다. 하지만 3번 Task는 아주 큰 차이를 보인다. 여기서 Task 2의 의미는 순차적 업데이트, Task 3의 의미는 무작위 업데이트를 의미한다. 즉, 무작위 업데이트는 플래시메모리를 저장매체로 이용할 때 심각한 성능 저하를 보인다. 그 이유는 FTL의 업데이트 알고리즘은 잦은 무작위 재 기록 연산은 많은 소거연산과 기록연산을 동반하여 수행되기 때문에 성능이 저하된다. Task 2의 경우 같은 재 기록 연산이 수행되지만 FTL에서 이미 기록되어 있는 페이지에 순차적으로 재 기록 요청이 발생할 경우 해당 페이지를 무효화하고 새로운 페이지를 할당하여 기록하는 업데이트 알고리즘이 동작하고 최악의 경우라 하더라도 소수의 소거연산만을 수반하는 교환(데이터 블록과 로그 블록의 교환) 연산이 수행되기 때문에 그만큼 알고리즘이 효과적으로 동작하기 때문이다. 실험의 결과로 플래시메모리를 저장매체로 하는 경우 DBMS와 같이 잦은 무작위 페이지 기록을 야기하는 응용의 업데이트 성능은 저하 될 수 있음을 알 수 있다.

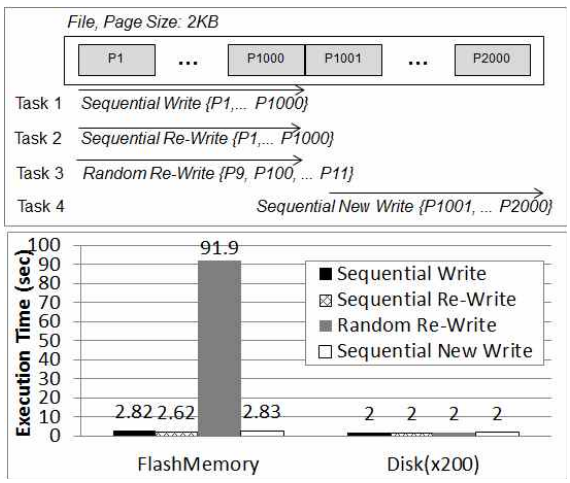


그림 3. 플래시메모리에서의 페이지 업데이트 성능 실험

플래시메모리와 관련된 연구는 플래시메모리 칩에 임베딩되는 FTL에 대한 연구들이 많았다. 이는 일반 디스크 저장매체의 인터페이스(읽기, 쓰기)의 플래시메모리 저장매체의 인터페이스(읽기, 쓰기, 소거)의 불일치 및 단위의 불일치를 맞추기 위한 매핑 알고리즘과 플래시메모리에서의 소거연산의 비용이 아주 크기 때문에 이미 기록되어 있는 페이지에 재 기록 요청을 수행 할 경우 해당 페이지를 무효화하고 새로운 페이지에 기록하는 알고리즘 들이 있다[2, 4, 7, 9]. 앞서 살펴본 간단한 실험의 Task 2번의 경우 플래시메모리 내부 FTL의 물리적 페이지 재 매핑 알고리즘이 동작한 것이다.

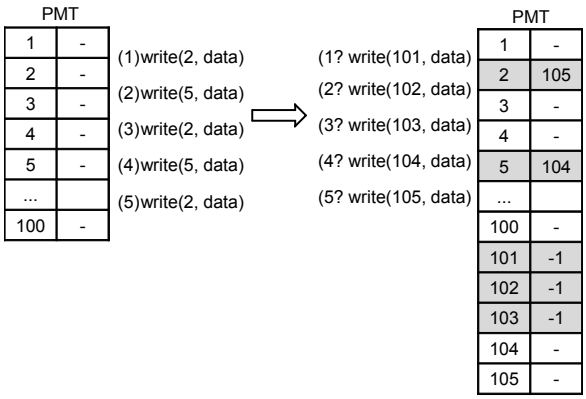


그림 4. 페이지 재 매핑 알고리즘의 동작 예

본 논문에서의 FUNs는 플래시메모리의 FTL을 직접 접근할 수 없다. 즉, 가장 일반적인 형태의 구조가 플래시메모리에 FTL이 포함된 구조이고 일반 응용(DBMS를 포함한)들은 FTL에 접근하지 못하고 디스크 저장매체를 사용하듯이 사용할 수 밖에 없다. 따라서 FUNs 모바일 DBMS 자체에서 페이지 업데이트 성능을 향상 시킬 수 있는 기법이 필요하며 그 방법이 논리적 페이지 재 매핑 방법(LPRM: Logical Page Re-Mapping)이다. 논리적 페이지 재 매핑 알고리즘은 그림 3의 Task 4의 의미와 동일하다. 즉, 초기에 1~1,000번 페이지에 데이터가 기록되어 있고, 임의순서로 이 1,000개를 업

데이트 할 때, 1~1,000번까지의 페이지를 무효화 시키고 1,001~2,000까지 순차적으로 기록하는 것이다. 간단한 알고리즘이지만 실험의 결과와 같이 그 효과는 크다. 이 기법의 동작은 그림 4에서와 같다.

(1)~(5)까지의 임의 순서 기록 요청에 대하여 새로운 페이지에 순차적으로 기록하는 (1') ~ (5')로 바꾸어 처리하고 페이지 매핑 테이블(PMT: Page Mapping Table)을 업데이트하는 순서로 동작한다. 일반적인 저장관리기의 페이지 쓰기 및 페이지 읽기에 페이지 재 매핑 알고리즘을 적용한 의사 코드는 아래와 같다.

```
Algorithm: LPRM_WritePage(pageId, page, dbFile, PMT, ReMT)
Input: pageId - logical page id, page - page body, dbFile - database file, PMT - page mapping hash table, ReMT - remapping hash table
begin
  newPageId ← Ø;
  remapPageId ← PMT.getMappingPage(pageId);
  If remapPageId = Ø then
    ReMT.add(pageId);
    PMT.add(pageId, pageId);
    newPageId = pageId
  Else
    If remapPageId ≠ pageId then
      ReMT.getRemapPage(remapPageId).valid = false;
    newPageId ← ReMT.getNewRemapPage();
    ReMT.add(newPageId);
    PMT.getMappingPage(pageId).value = newPageId;

  writeFile(dbFile, newPageId, page);
end
```

```
Algorithm: LPRM_ReadPage(pageId, dbFile, PMT, ReMT)
Input: pageId - logical page id, dbFile - database file, PMT - page mapping hash table, ReMT - remapping hash table
Output: page - page body
begin
  page ← Ø;
  realPageId ← PMT.getMappingPage(pageId);
  readFile(dbFile, realPageId, page);
  return page;
end
```

이와 같은 논리적 페이지 재 매핑 알고리즘은 그림 5의 FUNs의 모바일 DBMS의 구조 중 가장 하위인 저장 관리기에 적용하였다.

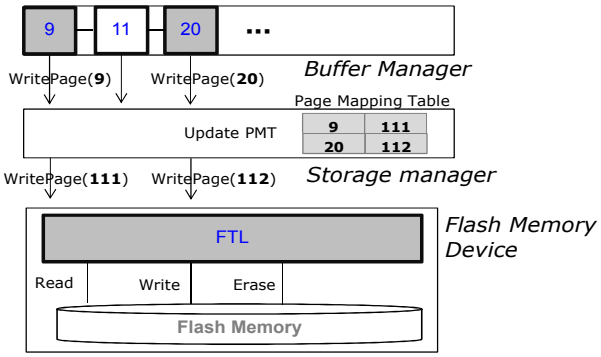


그림 5. LPRM 기법이 적용된 FUNs의 저장관리기

저장 관리기의 역할은 상위 버퍼 관리기에서의 페이지 할당, 페이지 읽기, 페이지 쓰기 및 페이지 할당 해제의 요청을 처리한다. 만약 버퍼 관리기가 페이지 교체 전략에 의해서 해당 버퍼 페이지를 기록해야 할 경우 저장관리기에 이를 요청하고 저장관리기는 페이지를 재 매핑 하여 새로운 페이지에 기록하고 페이지 매핑 테이블을 업데이트한다. 논리적 페이지를 재 매핑함으로써 FTL의 아웃플레이스 업데이트 알고리즘(Mirror, Spare, Log)에서 동일 페이지의 기록에 의한 복사, 여유, 로그 블록의 합병연산 등을 줄여서 플래시메모리의 업데이트 성능을 향상시킬 수 있다. 또한 내비게이션 응용프로그램 종료 시에 재 매핑되었던 페이지를 원위치(Re-Location) 시킬 수도 있다.

FUNs 데이터베이스는 다수의 테이블스페이스로 구성되고, 하나의 테이블 스페이스가 하나의 파일에 대응된다. 그림 6에서와 같이 하나의 테이블스페이스에 데이터 페이지가 존재하고 LPRM을 위한 영역이 존재한다. PMT은 지금까지 할당된 데이터페이지에 대한 매핑 정보가 저장되어 있다. 초기에는 ID와 매핑 정보가 동일하다. 만약 데이터페이지가 업데이트 되면 새로운 페이지가 LPRM 영역에 추가되고 ReMT (Remapping Table)에 이 페이지의 ID를 추가한 후 PMT는 이 테이블의 레코드를 포인팅하게 된다. 동일한 데이터페이지가 여러 번 업데이트 될 경우 무효(Invalid) 페이지가 증가하게 되고 ReMT에서는 이를 관리한다. 페이지 원위치시

에는 PMT 테이블을 스캔하여 매핑 정보를 이용하여 해당 페이지를 데이터 페이지에 원위치시킨다.

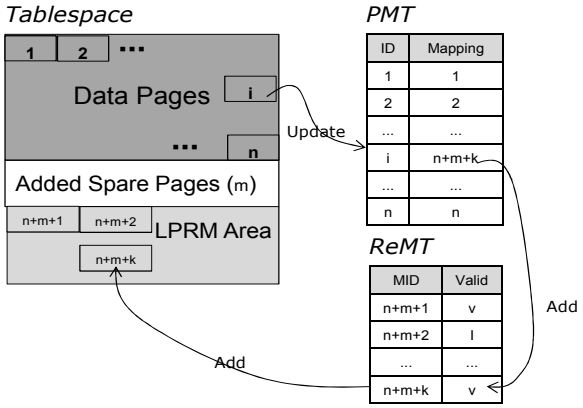


그림 6. LPRM을 위한 매핑 테이블

4. 실험

4.1 가상 데이터를 이용한 업데이트 실험 및 평가

이번 절에서는 플래시메모리를 저장매체로 할 경우 업데이트 성능을 향상시키기 위한 논리적 페이지 재 매핑 및 페이지 원위치 기법에 대한 실험을 수행하였다. 실험 환경 및 방법은 표 2와 같다.

표 2. 실험환경

항목	설명
모바일단말	LG KC1, CPU 800 Mhz, 128MB DRAM
저장매체	NAND 플래시메모리 2GB
DB 크기	페이지 크기: 1024 byte 버퍼 크기: 1,000 / 500 / 100페이지
레코드크기	약 100byte
레코드추가	순차적 10,000개 추가
업데이트 Task	①순차적 레코드 10,000개 업데이트 ②임의 순서 레코드 10,000개 업데이트 ③논리적 페이지 재매핑 임의 순서 레코드 10,000개 업데이트 ④페이지 원위치(ReLocation) 업데이트
실험결과	업데이트 시간 및 페이지 개수

DBMS를 이용한 페이지 업데이트의 실험은 3장의 그림 3에서 수행한 실험과 많은 차이가 있다. 3장에서 수행한 실험은 응용 프로그램에서 파일을 생성하고 파일에 일정 크기에 데이터를 기록 및 재기록을 수행한 것이다. 하지만 실제 DBMS는 버퍼 관리기를 통해서 데이터가 관리되고 버퍼 교체 전략에 의해서 필요한 데이터 페이지가 파일에 기록되는 등 복잡한 로직에 의해서 수행된다. 업데이트 실험은 총 4가지 종류로 나눈다. 순차적 레코드 업데이트와 임의 순서 레코드 업데이트 그리고 논리적 페이지 재 매핑 기법 적용 업데이트 및 페이지 원위치 업데이트이다. 실험 순서는 다음과 같다.

1) FUNs 데이터베이스 생성, 테이블 생성
2) 레코드 10,000개 생성 후 순차적 일괄 삽입
3) Task ①, ②, ③ 및 ④ 수행
4) FUNs 데이터베이스 종료

실험 결과는 그림 7과 같다. (a)는 업데이트 Task 의한 페이지 기록 회수를 나타낸다. 실제 10,000개의 레코드는 총 1,250개의 페이지에 순차적으로 기록되어 있다. 업데이트 트랜잭션은 해당 레코드가 포함된 페이지를 읽어 버퍼에 적재하고 이 페이지의 레코드를 변경하고 저장매체에 해당 페이지를 기록하는 순으로 동작한다. 이는 순차적 레코드 업데이트이기 때문에 버퍼에 적재된 해당 레코드가 위치하는 페이지에 존재하는 모든 레코드는 단 한 번의 읽기와 쓰기에 의해 모든 레코드를 업데이트 할 수 있다. 즉, 최적의 경우이기 때문에 1,250개의 페이지만을 읽고 변경하고 기록하면 된다. 임의 순서 업데이트의 경우 무작위 순서로 레코드를 읽기 위해 해당 레코드가 포함된 페이지를 읽어 버퍼에 적재하고 해당 레코드를 변경한다. 이 때 최악의 경우에는 하나의 레코드를 변경하기 위해 하나의 페이지를 읽고 이를 저장매체에 기록하는 경우이다. 이는 버퍼의 크기에 의존적이다. 즉, 평균적으로 10,000개의 레코드가 1,250개의 페이지에 저장되어 있기 때문에 하나의 페이지에 8개의 레코드가 저장되어 있고 버퍼의 크기가 적을 경우에는 동일 페이지를 8번 읽고 기록하는 경우가 발생할 수 있다. 따라서 버퍼의 크기가 크면 클수록 페이지 읽기와 쓰기의 회수가 줄어들 수 있다. LPRM은 버퍼 관리기에서 해당 페이지를 저장 관리기에 기록을 요청할 때마다 새로운 페이지에 순차적으로 기록하기 때문에 페이지 기록 회수는 임의순서 업데이트와 동일하다. 마지막으로 페이지 원위치는 페이지 매핑 테이블을 스캔하여 변경된 페이지만을 찾아서 원위치시키기 때문에 1,250개의 페이지만을 읽고 기록한다.

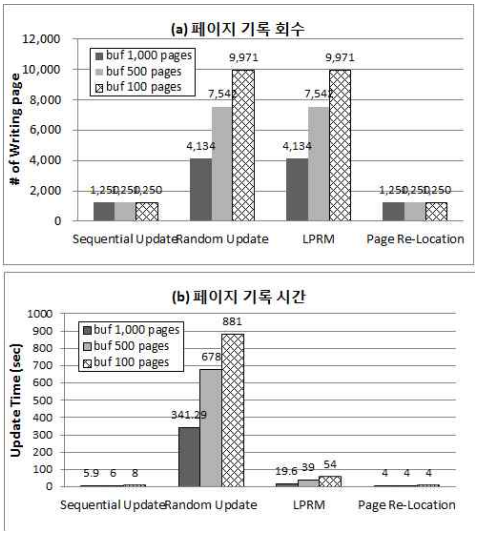


그림 7. 페이지 업데이트 성능 실험 결과

(b)는 (a)의 각각에 대응하는 업데이트 시간이다. 순차적 업데이트는 플래시메모리의 특성으로 인해 높은 성능을 보인다. 순차적 업데이트와 페이지 원위치는 동일한 개수의 페이지를 기록하는데 약간의 차이가 있다. 순차적 업데이트는 레코드를 업데이트하는 것이기 때문에 DBMS 질의처리를 통해 업데이트가 되는 것이고 페이지 원위치는 DBMS의 저장 관리기의 페이지 매핑 테이블만을 읽어서 해당 페이지를 업데이트하는 것이기 때문에 페이지 원위치 업데이트가 조금 더 빠르게 수행된다. 무작위 업데이트는 플래시메모리의 특성상 FTL에 의해서 관리되는 블록의 합병 및 교환이 잦아져서 결국 블록의 소거, 섹터의 읽기/쓰기가 증가해서 심각한 성능 저하를 보이고 있다. 이를 해결하기 위한 LPRM의 경우 페이지 기록 회수에 비례하여 상대적으로 아주 빠른 처리 속도를 보이고 있다. 결론적으로 무작위 업데이트가 빈번한 DBMS에서 LPRM과 페이지 원위치를 적용한 성능이 무작위 업데이트의 성능에 비해 아주 좋은 성능을 나타냄을 알 수 있다.

위의 각 4가지 업데이트 방법에 의한 업데이트 페이지의 순서(패턴)은 <그림 8>과 같다. (a)의 순차적 업데이트는 그래프와 같이 총 1,250개의 페이지를 순차적으로 업데이트한다. 여기서 데이터베이스 전체 페이지는 약 1,500개 정도이며 여기에는 색인에 의한 페이지 및 메타 페이지 등이 포함되어 있다. (b)는 무작위 업데이트 패턴을 나타내고 있다. 총 1,250개의 페이지를 총 9,598 번 무작위로 기록하는 패턴이다. (c)는 LPRM 업데이트 패턴이다. 페이지 기록 회수는 (b)와 동일하지만 순차적으로 기록한다. 마지막 (d)는 페이지 매핑 테이블을 스캔하여 매핑된 페이지를 읽어서 순차적으로 기록하는 순서를 나타내고 있다. 그림 8은 버퍼 크기가 100 페이지인 경우이지만 버퍼 크기에 상관없이 동일한 패턴을 보인다.

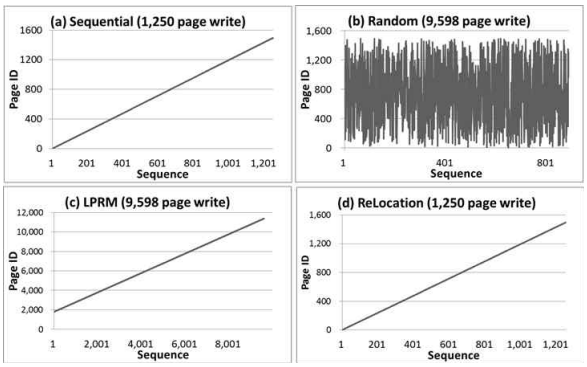


그림 8. 페이지 업데이트 패턴(버퍼 크기 100)

지금까지의 실험은 가상 데이터를 이용하여 FUNs DBMS 자체 실험이었다. 추가 실험으로 SQLite를 이용하여 레코드를 업데이트하고 이를 FUNs와 비교하였다. 두 시스템 모두 동일한 환경(버퍼 2.5MB)에서 동일한 가상 데이터 셋(레코드 크기 300B, 레코드 업데이트 개수 10,000)을 이용하였다. 실험 결과는 그림 9와 같다. SQLite_Sequential과 FUNs_Sequential, SQLite_Random과 FUNs_Random은 거의 유사한 성능을 보인다. 하지만 무작위 업데이트에 대해서 FUNs에 논리 페이지 재 매핑과 페이지 원위치를 적용 시킨 경우 약 22초 소요되며 이는 SQLite의 약 366초에 비해 성능이 월등히 뛰어남을 알 수 있다.

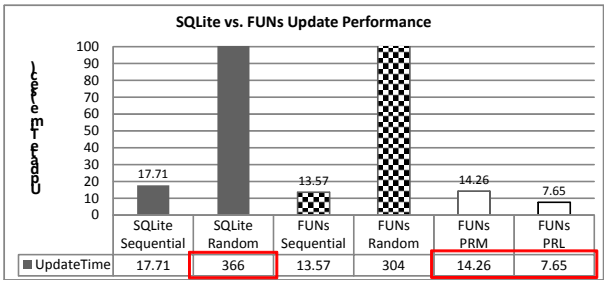
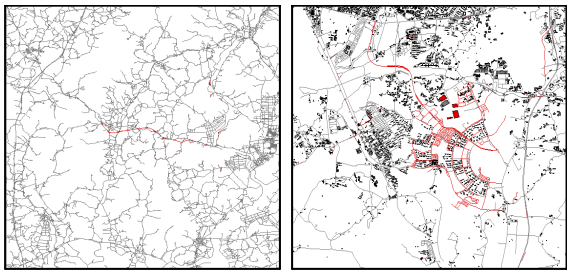


그림 9. SQLite와 FUNs의 무작위 업데이트 성능 비교 실험결과

4.2 점진적 내비게이션 맵 업데이트 실험 및 평가

이번 절에서는 실제 맵 데이터 업데이트를 실험하였다. 업데이트 데이터 셋은 실제 광주 지역 및 동탄시 업데이트 데이터를 이용하였다. 그림 10은 광주 및 동탄 지역의 도로네트워크 업데이트 데이터를 나타내고 있다.



(a) 광주 지역 (b) 동탄 지역

그림 10. 부분 맵 업데이트 데이터 셋

그림 10에서 붉은 색으로 표시 된 부분이 업데이트 객체이며 상세 내역은 표 3과 같다.

표 3. 업데이트 데이터 셋

지역	업데이트 객체(크기 및 개수)
광주	Network 40 Kb (Del #: 49, Add #: 270, Chg #: 90)
동탄	Network 100 Kb (Del #: 106, Add #: 1,103, Chg #: 124)

각 지역의 업데이트 맵 데이터에 의한 페이지 업데이트 순서를 살펴 보았다. FUNs의 페이지 크기는 16KB, 버퍼 크기는 250 페이지이다. 그림 11의 (a-1), (a-2), (a-3)은 광주지역 업데이트 데이터에 대한 보통 업데이트, 논리 페이지 재 매핑 적용 업데이트, 페이지 원위치 업데이트에 대한 업데이트 패턴을 나타내고 있고 (b-1), (b-2), (b-3)는 동탄 지역에 대한 업데이트 패턴을 나타내고 있다.

보통 업데이트는 일반적으로 데이터베이스 업데이트가 무작위 업데이트이기 때문에 무작위 업데이트를 나타낸다. (a-1)과 (b-1)의 그래프와 같이 업데이트 페이지가 무작위로 업데이트 되며 페이지 개수는 각각 131개와 157개이다. 논리 페이지 재 매핑을 적용한 경우 무작위 업데이트 페이지를 순차적 업데이트 페이지로 재 매핑을 하기 때문에 (a-2), (b-2)와 같이 순차적으로 페이지가 업데이트 됨을 알 수 있다. 이 때 무작위 업데이트 페이지와 개수는 동일하다. 마지막으로 페이지 원위치 업데이트는 응용 프로그램, 즉, 내비게이션이 종료될 때 지금까지 페이지 재 매핑 되었던 페이지를 찾아서 원위치 시키기 위한 업데이트를 의미한다. 페이지 매핑 테이블을 차례로 스캔하면서 재 매핑된 페이지를 다시 읽어와서 원래 위치에 업데이트를 한다. (a-3), (b-3)과 같이 순차적으로 스캔하면서 원위치 시키기 때문에 부분적으로 순차적으로 업데이트가 됨을 알 수 있으며 페이지 업데이트 개수는 각각 123개, 138개이다. 이 의미는 정확히 업데이트 되는 페이지 개수는 123개와 138개이지만 (a-1), (b-1)에서와 같이 중복해서 페이지가 업데이트되기 때문에 페이지 업데이트 회수는 이보다 일반적으로 크다.

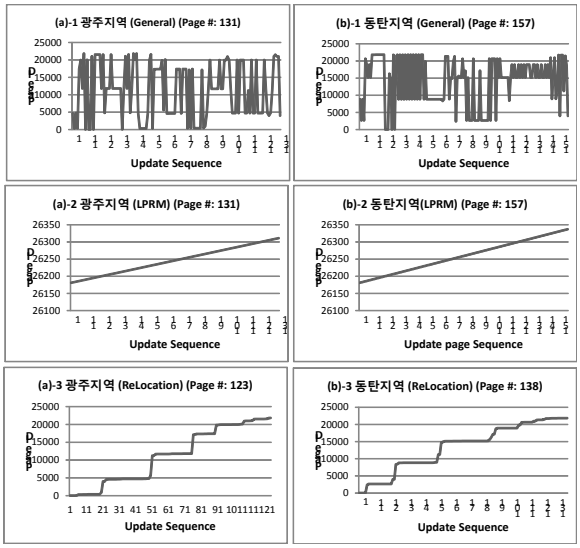


그림 11. 광주 및 동탄 맵 업데이트 데이터의 페이지 업데이트 패턴

이들에 대한 각각의 업데이트 성능은 그림 12와 같다. (a)에서와 같이 보통 업데이트의 경우 업데이트 시간이 8.74초이며 논리 페이지 재 매핑을 적용한 경우는 0.8초이다. 그리고, 재 매핑된 페이지를 원위치 시키는데 소요되는 시간은 4.1 초다. 논리 페이지 재 매핑과 페이지 원위치 방법을 적용한 경우의 I/O 시간이 보통 업데이트의 성능보다 뛰어남을 알 수 있다. 동탄 지역의 경우에도 논리 페이지 재 매핑과 페이지 원위치의 I/O 시간이 약 4.8초이고 보통 업데이트의 시간이 약 11초며 보다 성능이 좋음을 알 수 있다.

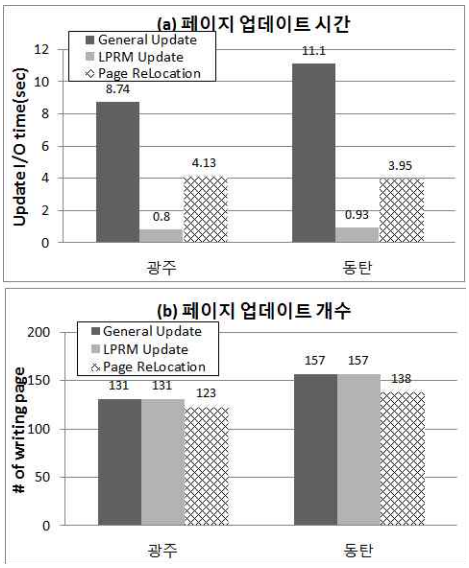


그림 12. 광주 및 동탄 맵 업데이트 성능 실험 결과

5. 결론 및 향후 연구과제

본 논문에서는 플래시메모리에서의 데이터 무작위 기록 성능향상을 위한 페이지 관리방법에 대해서 연구하였다. 특히, 내비게이션에 특화된 모바일 DBMS인 FUNs의 저장관리기에 알고리즘을 적용하여 가상 및 실제 데이터를 이용하여 다양한 실험을 수행하였고 결과를 분석하였다. 또한 모바일 디바이스에서 가장 많이 사용되는 SQLite와도 성능을 비교하였다. 그 결과 모바일 DBMS에 적용된 논리적 페이지 재 매핑과 페이지 원위치 기법에 의해서 페이지 업데이트의 성능을 저하를 방지 할 수 있었고 SQLite 보다 뛰어난 성능을 보였다. 본 연구에 이어 추가적으로는 DBMS의 버퍼관리기 및 질의처리기 등에 플래시메모리의 특징을 반영한 다양한 기법들과 공간데이터 처리 시에 공간데이터의 특징을 반영하여 보다 성능을 향상시킬 수 있는 기법[10]들에 대한 추가 연구가 필요할 것으로 생각된다.

참 고 문 헌

[1] H.J.Choi, S.Lim, and K.H.Park, 2009, “JFTL: a flash translation layer based on a journal re-mapping for flash memory,” In ACM Transaction on Storage, Vol. 4, No. 4, January.

[2] Petro Estakhri and erhanu Iman, 1999, “Moving Sequenctial Sectors within a Block of Information in a Flash Memory Mass Storage Architecture,” United States Patent 5,930,815.

[3] E. Gal and S. Toledo, 2005, “Algorithms and Data Structures for Flash Memories,” ACM Computing Surveys, vol. 37, no. 2, pp. 138-163.

[4] J.S. Kim et al., 2002, “A Space-Efficient Flash Translation Layer for Compact Flash Systems,” IEEE Trans. Consum. Electron., vol. 48, no. 2, pp. 365-375.

[5] S.W. Lee et al., 2007, “Research Issues in Next Generations DBMS for Mobile Platforms,” Mobile HCI, ACM, Sept.

[6] S.W. Lee and B.K. Moon, 2007, “Design of Flash-Based DBMS: An In-Page Logging Approach,” SIGMOD pp. 55-66.

[7] S. Lee et al., 2008, “Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems,” SIGOPS Oper. Syst. Rev., vol. 62, no. 6, pp. 36-42.

[8] K. W. Min, K. W. An, I. J. Jang, and S. I. Jin, 2011, “A System Framework for Map Air Update Navigation Service,“ ETRI Journal, vol. 33, no. 4, pp. 476-486.

[9] Takayuki Shinohara, 1999, “Flash Memory Card with Block Memory Adress Arrangement,” United States Patent 5,905,993.

[10] 김정준, 2009, “플래시 메모리 기반의 효율적인 공간 인덱스 기법,” 한국공간정보시스템학회 논문지, 제11권, 제2호, pp. 133-142.

[11] 민경욱, 2008, “실시간 맵 업데이트를 위한 모바일 공간 DBMS 개발,” 한국GIS학회, 한국공간정보시스템학회 공동춘계학술대회 논문집, pp. 37-40.

[12] 민경욱, 2008, “부분 맵 업데이트 지원 내비게이션을 위한 모바일 공간 DBMS 개발 및 성능 평가,” 정보처리학회논문지, 제15-D권, 제5호, pp. 609-620.

[13] 박지웅, 2005, “공간 모바일 장치를 위한 내장형 공간 MMDBMS의 설계 및 구현,” 한국공간정보시스템학회 논문지, 제7권, 제1호, pp. 25-37.

논문접수 : 2012.04.17
수 정 일 : 1차 2012.09.28 / 2차 2012.10.26
심사완료 : 2012.10.30

민 경 욱

1996년 부산대학교 전자계산학과(학사)
1998년 부산대학교 전자계산학과(석사)
2012년 충남대학교 컴퓨터공학과(박사)
2001년~현재 한국전자통신연구원 선임연구원
관심분야는 GIS, LBS, 자동차-IT

최 정 단

1993년 중앙대학교 컴퓨터공학과(학사)
1995년 중앙대학교 컴퓨터공학과(석사)
2005년 충남대학교 컴퓨터과학과(박사)
1995년~현재 한국전자통신연구원 책임연구원
관심분야는 자동차-IT, 컴퓨터그래픽스, 영상처리

김 주 완

1993년 부산대학교 컴퓨터공학과(학사)
1995년 부산대학교 컴퓨터공학과(석사)
2004년 충남대학교 컴퓨터과학과(박사)
1995년~현재 한국전자통신연구원 책임연구원
관심분야는 GIS, LBS, IT융합