낸드 플래시 메모리 상에서 효율적인 MR-트리 동작을 위한 지연 연산 기법

(Delay Operation Techniques for Efficient MR-Tree on Nand Flash Memory)

이 현 승 † 송 하 윤 ††
(Hyun Seung Lee) (Ha Yoon Song)

김 경 창 ††
(Kyung-Chang Kim)

요 약 플래시 메모리 중 저장장치로 사용되는 낸드 플래시 메모리는 유비쿼터스 및 모바일 환경에 적합한 특성으로 다양한 분야의 저장장치로 이용되고 있으며 효율적인 활용을 위한 많은 연구가 진행되고 있다. 모바일 환경에서 이용할 수 있는 멀티미디어 데이타베이스 시스템을 위한 인덱스로써 공간 데이타 액세스가 가능한 R-트리의 검색 성능을 향상 시킨 MR-트리는 메인 메모리 데이타베이스 시스템에서 캐쉬 미스를 줄이고 중간 노드의 이용률을 높임으로써 연산 성능을 높일 수 있는 특성을 가진다. 본 논문에서는 검색 성능이 좋은 MR-트리를 활용하여 낸드 플래시 메모리 기반에서 효율적인 동작을 위한 지연 연산 기법을 제안하였다. MR-트리의 노드 크기를 낸드 플래시 메모리의 쓰기 연산 단위에 맞추고 인덱스 수정 연산 시 노드 크기만큼 지연 연산하여 쓰기 연산으로 인한 플래시 메모리에서의 추가적인 비용을 줄이고 연산 횟수를 줄여 인덱스 성능을 향상 시켰다.

키워드 : 데이타베이스, 낸드플래시메모리, MR-트리, 데이타베이스 인덱스 성능

Abstract Embedded systems usually utilize Flash Memories with very nice characteristics of non-volatility, low access time, low power and so on. For the multimedia database systems, R-tree is an indexing tree with nice characteristics for multimedia access. MR-tree, which is an upgraded version of R-tree, has shown better performance in searching, inserting and deleting operations than R-tree. Flash memory has sectors and blocks as a unit of read, write and delete operations. Especially, the delete is done on a unit of 512 byte blocks with very large operation time and it is also known that read and write operations on a unit of block matches caching nature of MT-tree. Our research optimizes MR-tree operations in a unit of Flash memory blocks. Such an adjusting leads in better indexing performance in database accesses. With MR-tree on a 512B block units we achieved fast search time of database indexing with low height of MR-tree as well as faster update time of database indexing with the best fit of flash memory blocks. Thus MR-tree with optimized operations shows good characteristics to be a database index schemes on any systems with flash memory.

Key words : Database, Nand Flash Memory, MR-Tree, Database Indexing Performance

1. 서 론

낸드 플래시 메모리는 비휘발성, 휴대성, 저전력 소모 등 유비쿼터스 및 모바일 컴퓨팅 환경에 적합한 특성을 가진다. 따라서 이동 컴퓨팅 장치의 저장매체로의 활용이 증가하고 있으며 그 적용 분야도 다양하다. 또한 플래시 메모리의 용량도 기술 개발에 따라 점점 대용량화 되고 있어 개인용 컴퓨터 및 노트북에서도 활용이 가능해지고 있다. 이에 따라 저장 데이타에 대한 효율적인 접근 및 관리를 위한 인덱싱 기법의 적용이 요구된다.

플래시 메모리는 읽기 속도에 비해 쓰기 연산 속도가 느리며 덮어쓰기 연산이 지원되지 않아 소거 연산이 있는 특징이 있다. 읽기, 쓰기 연산은 페이지 단위, 소거연산은 블록 단위로 이루어져 반복된 쓰기 연산은 성능 저하뿐만 아니라 소거횟수가 제한적인 플래시 메모리의 수명에도 지장을 줄 수 있다. 따라서 플래시 메모리 기반의 저장 데이타를 관리하기 위해서는 데이타 접근 및 수정 시 수반되는 쓰기 연산으로 인한 성능저하를 줄일 수 있는 방법이 요구된다.

MR-트리[1]는 공간 데이타 접근이 가능한 R-트리의

· 이 논문은 2007학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음
· 이 논문은 제34회 추계학술대회에서 '낸드 플래시 메모리 상에서 효율적인 MR-트리 동작을 위한 지연 연산 기법'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 홍익대학교 컴퓨터공학과
lhseung@gmail.com
†† 정 회 원 : 홍익대학교 컴퓨터공학과 교수
song@cs.hongik.ac.kr
kckim@wow.cs.hongik.ac.kr

논문접수 : 2007년 12월 6일
심사완료 : 2008년 9월 1일

변형된 형태의 공간 인덱스 구조이다. 메인 메모리 데이타베이스 시스템에서는 CPU 속도와 메모리 속도 차이가 점점 벌어지는 현실에서 캐쉬 미스를 줄이느냐가 중요한 문제가 되는데 이러한 점을 고려하여 MR-트리는 R-트리에 비해 중간 노드 엔트리들의 이용률을 높이고 트리의 높이를 줄여 캐쉬 동작에 민감한 장점을 가진다.

플래시 메모리에 인덱싱 기법을 적용하면 대용량의 데이타에 대한 효율적으로 접근 및 관리가 가능하지만 트리 인덱스의 특성상 삽입(insert), 삭제(delete), 노드 분할(split) 등의 수정이 자주 일어나게 되고, 한 노드 엔트리의 수정으로도 많은 쓰기 연산과 소거 연산을 유발하게 됨으로써 성능 저하를 가져올 수 있는 문제점을 갖고 있다. 이 문제를 해결하기 위해 제안된 BFTL[2]과 플래시 메모리 시스템 상에서의 효율적인 R-트리 구현[3]은 버퍼링 레이어를 통해 쓰기 연산 횟수를 줄임으로써 문제를 개선하였으나 검색 연산이 추가적으로 발생하는 단점을 지닌다.

본 논문에서는 검색성능이 향상된 MR-트리의 플래시 메모리 상에서 효율적인 동작을 위해 MR-트리의 노드 크기 별 성능을 비교하고 지연 연산 기법을 제안하여 쓰기 연산 횟수를 줄이고 검색 성능을 향상시켰다.

본 논문의 구성은 2장에서 플래시 메모리의 특성 및 플래시 메모리와 인덱싱 기법 관련 연구들을 살펴보고 3장에서 본 연구에 활용한 MR-트리에 대해 설명한다. 4장에서는 제안하는 지연 연산을 설명하며 5장에서 구현 및 실험 결과를 분석하고 6장에서 결론을 맺는다.

2. 관련 연구

2.1 플래시 메모리의 특성

플래시 메모리는 섹터(Sector) 단위로 읽기, 쓰기 연산이 수행되지만 소거 연산은 블록(Block) 단위로 이루어지며 덮어쓰기가 지원되지 않으므로 쓰기 연산을 위해서는 썼던 페이지는 소거 연산을 거쳐 해당 블록을 지워야만 페이지에 데이타를 쓸 수 있는 제약이 있다. 또한 블록의 소거 횟수는 10만~100만 번 사이로 제한적이다.

플래시 메모리가 동작 후 한 번도 쓰여 지지 않은 페이지를 프리 페이지(Free page)라 하고 올바르지 못한 데이타를 가지고 있는 기존의 페이지를 데드 페이지(Dead page)라 하며, 기존 페이지의 데이타가 복사된 페이지는 라이브 페이지(Live page)라 한다. 플래시 메모리에 수정이 잦을수록 데드 페이지가 쌓이고 프리 페이지가 줄어들게 되므로 데드 페이지를 프리 페이지로 바꿔주는 가비지 컬렉션(Garbage collection) 작업이 필요한데 이는 특정 블록을 선택하여 지움으로써 프리 페이지를 만드는 과정을 뜻한다.

또한 플래시 메모리는 소거 연산 횟수가 제한되어 있어 소거 연산이 일정 부분에만 몰리지 않고 골고루 사용되도록 하는 웨어레벨링(wear leveling)도 필요하다. 가비지 컬렉션과 웨어레벨링은 쓰기 연산과 밀접하게 연관되며 쓰기 연산 자체의 비용 외에 추가적인 비용이 들어가게 된다.

2.2 플래시 메모리와 인덱스

대용량의 데이타의 효과적인 접근 및 수정을 위해 이용되는 인덱스는 데이타의 삽입 시 순차 쓰기 연산보다 임의 쓰기 연산인 경우가 많은 점이 플래시 메모리 상에서 성능 저하를 일으키는 원인이 된다.

트리 인덱스의 경우 하나의 노드 엔트리에 수정이 일어나면 해당 엔트리의 데이타와 데이타를 가리키는 주소 및 키 값, 그리고 부모 노드나 자식 노드에 대한 포인터 정보 등 노드에 대한 관련 정보의 수정이 요구되고 이러한 특성을 지닌 인덱스를 그대로 플래시 메모리에 적용할 경우 많은 페이지의 수정이 수반되게 된다. 이는 데드 페이지를 증가와 동시에 프리 페이지 소모를 증가시킴으로써 소거 연산 또한 자주 필요하게 되고 결과적으로 소거 횟수가 제한되어 있는 플래시 메모리의 수명을 단축시키게 된다. 그러므로 플래시 메모리 상에서 효율적인 인덱스 사용을 위해서는 플래시 특성을 고려한 임의 쓰기 연산의 처리 기법이 요구된다.

BFTL[2]과 플래시 메모리 시스템 상에서의 효율적인 R-트리 구현[3]기법은 각각 B-트리와 R-트리를 플래시 메모리 상에서 이용할 때에의 연산 처리 기법을 제안한 것이다. 인덱스 엔트리의 삽입, 삭제, 갱신 등 어떠한 수정이 일어날 때마다 색인단위(Index unit)를 생성하여 수정된 정보를 예약 버퍼(Reservation buffer)에 유지하다가 버퍼가 다 차게 되면 플래시 메모리에 기록하는 방식이다. 플래시 메모리 한 개의 섹터에 여러 개의 색인 단위를 모아서 기록하여 한 노드의 내용이 여러 섹터에 나누어 기록되므로 하나의 노드 데이타가 들어간 각 섹터에 대한 정보는 노드변환테이블(Node translation table)에서 관리한다. 노드변환테이블이 커지면 압축을 수행하는데 자주 발생될 수 있는 압축과 노드 검색 시 노드변환테이블을 거쳐야 하는 추가적인 검색 비용이 단점이 될 수 있다.

BOF[4]는 이러한 BFTL의 단점을 보완하여 한 노드에 대한 색인단위는 한 섹터에만 저장하도록 하는 방법이다. 이 기법은 노드 변환 테이블을 쓰지 않음으로 검색 성능을 높이고 압축으로 인한 비용을 줄였다.

플래시 상의 멀티미디어 데이타베이스를 위한 공간 인덱스를 활용 연구가 아직 미진하며 플래시 메모리의 빠른 검색 속도와 휴대성 등 여러 장점들을 이용할 수 있는 인덱스 구조와 인덱스 수정 단위인 노드 크기의

적합성이 고려된 효율적인 연산 처리 기법 연구가 요구된다.

이에 본 논문은 R-트리의 변형된 모델인 MR-트리의 플래시 메모리 상에서 효율적인 동작을 위한 지연 연산 기법과 노드 크기별 성능을 알아본다.

3장에서는 본 논문에서 활용한 MR-트리의 특성에 대해 알아보고 플래시 메모리에의 적용을 위한 고려 사항을 알아본다.

3. MR-트리

3.1 MR-트리의 특성

MR-트리[1]는 R-트리보다 캐쉬 동작에 민감하여 메인 메모리 데이타베이스 환경에 적합하도록 변형한 형태로써 R-트리가 높이 균형을 위해 자주 발생시키는 높이 증가를 줄여준다.

R-트리는 모든 리프 노드들의 높이가 같은 균형 트리의 형태를 지니지만, MR-트리는 서브 트리들 사이의 높이가 불균형 할 수 있으며 그 차이는 최대 1 이하이다. 그리고 분할 발생 시 삽입 경로 상에 빈 엔트리를 지닌 노드가 존재할 경우에만 분할을 상위 노드로 전달하며 삭제의 경우에도 선택적으로 재삽입 여부를 판단한다.

MR-트리는 그림 1과 같이 중간 노드(Internal node), 리프 노드(Leaf node), 그리고 반-리프 노드(Half-leaf node)로 구성된다. 반-리프 노드는 리프 노드들과 데이타 객체에 대한 엔트리들을 모두 지니는 노드이다. MR-트리 구조에는 노드의 높이(Height of node)를 나타내는 필드와 데이타 엔트리의 수(Number of data entries)를 나타내는 필드가 추가적으로 존재하며 각각 MR-트리의 불균형 상태 검색과 노드 안에 저장하고 있는 데이타 엔트리 저장에 사용된다.

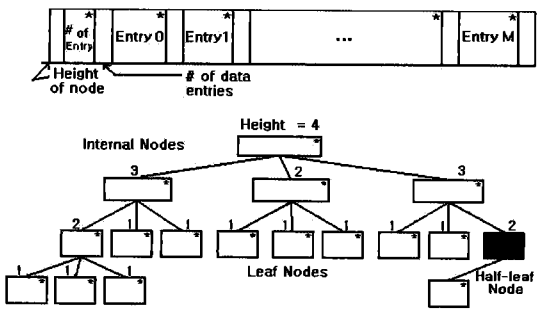

그림 1 MR-트리의 구조

3.2 MR-트리와 노드 크기

MR-트리는 리프가 아닌 중간 노드 엔트리들을 100%에 가깝게 사용하여 트리의 높이와 중간 노드 엔트리의 MBR(Minimum Bounding Rectangle)을 줄여준다.

트리의 노드 크기에 따라 생성되는 트리 형태가 달라짐으로써 검색, 삽입, 삭제 연산 속도에 영향을 준다. 노드 크기 별 검색 실험을 위해 R-트리와 MR-트리의 MBR의 크기를 16바이트로 하고 검색 시간을 비교해보면 노드의 크기가 커지면서 검색 시간은 빠르게 최소상태에 도달하며, 다시 조금씩 증가하는 것을 확인할 수 있다[1]. 검색 사각형의 크기가 작을수록 검색 성능이 좋고 검색 시간이 최소 상태에 도달하는 노드 크기는 256 바이트에서 512 바이트에 해당되며 이보다 더 작으면 트리의 높이 증가로 인해 검색 시간이 늘어나고 노드 크기가 더 커지면 조금씩 검색 시간이 증가한다. 이는 노드 크기가 커져도 접근해야 하는 노드의 수는 크게 줄어들지 않고, 캐쉬 미스의 수는 점점 증가하기 때문이다.

표 1 삽입 연산 100만개에 대한 노드 크기 별 특성

노드크기(byte)	128	256	512	1024
노드수	21153	3615	664	52
트리전체크기(Mbyte)	212.5	399.8	812.9	1532.7
트리높이(Height)	11	8	5	4
트리분할 수(Split)	241020	118494	57195	28092

따라서 MR-트리에서 좋은 검색 성능과 플래시 메모리의 검색, 삽입 연산의 단위를 적용하기 위해서는 섹터 크기에 해당하는 512 바이트를 노드 크기로 이용하는 것이 적합하다.

표 1은 100만개의 공간 데이타를 랜덤하게 생성하여 연속적으로 삽입 연산을 실행했을 때 생성되는 MR-트리의 특성을 나타낸다. MR-트리가 생성될 때 사각형(Rectangle) 데이타 정보가 노드를 구성하는데 한 노드에는 각 노드 크기별로 6개에서 51개 까지 여러 개의 사각형이 들어갈 수 있다. 노드의 크기가 커질수록 트리의 노드 수와 높이, 분할 횟수는 적어지지만 트리 전체 크기는 커지는 것을 알 수 있다. 이는 한 노드에 들어가는 데이타의 양이 늘어나면서 분할 횟수를 줄이고 트리의 높이도 낮아져 검색 성능을 향상 시킨다. 그러나 노드 크기가 커질수록 삽입이나 분할 등으로 노드가 생성될 때마다 큰 노드가 추가됨으로써 트리 전체의 크기를 늘리게 된다.

트리의 높이와 넓이는 검색 속도에 영향을 주므로 트리의 크기와 삽입, 삭제 연산에 성능 저하가 일어나지 않는 최적의 높이, 노드 크기의 선택이 요구된다.

4. MR-트리와 플래시 메모리

3장에서 MR-트리의 특성을 살펴보면서 플래시 메모

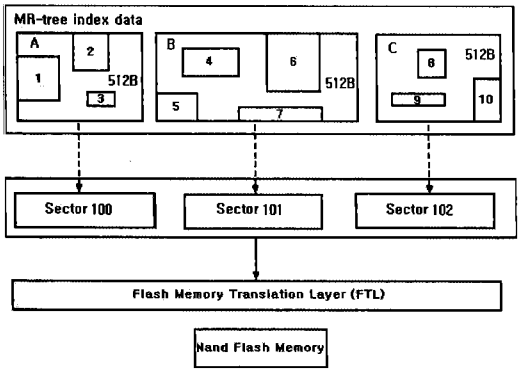

그림 2 플래시 메모리 상에서 지연연산 MR-트리 구조

리의 기본 접근 단위(Access unit)가 섹터 크기이므로 이를 고려한 최상의 검색 성능을 보일 수 있는 노드 크기는 512 바이트일 때인 것을 알 수 있었다.

이에 MR-트리를 플래시 메모리에 적용함에 있어 본 논문에서는 인덱스 수정 연산에 대한 지연 연산 기법을 제안하였다. 이 기법은 섹터 크기보다 작은 노드 엔트리에 대한 삽입, 삭제, 검색 연산 등 수정 동작의 연산 결과를 그대로 적용하지 않고 일정 크기만큼 지연 수행하는 것으로 플래시 메모리에 대한 접근 횟수 자체를 줄이면서 플래시 메모리의 연산 단위를 고려하는 방식이다. 지연 연산하는 일정 크기를 섹터 크기에 맞추게 되면 데드 페이지 증가와 프리 페이지 소모를 줄이면서 전체 연산 속도를 향상 시킬 수 있어 소거 연산 횟수가 제한된 플래시 메모리의 수명에도 긍정적인 영향을 줄 수 있다.

또한 MR-트리의 검색 성능은 R-트리에 비해 향상되어있으며 노드 크기가 256~512 바이트일 때 가장 좋은 검색 속도를 보이므로 플래시 메모리의 연산 단위와 동일한 노드 크기로 트리를 최적화 하면 플래시 상에 지연 연산을 적용할 때 한 노드에 대한 정보를 그대로 쓸 수 있어 한 노드에 대한 정보를 쉽게 관리할 수 있다.

5. 실험 결과

본 논문에서는 플래시 메모리 상에서의 MR-트리 이용할 때에 임의 크기만큼의 지연 연산 기법을 시행했을 때의 성능을 기법을 적용하지 않았을 때와 비교하였다.

트리 인덱스를 구현하고 플래시 메모리는 삼성 Nand Flash SLC-small block의 제품 데이타[6]를 이용하여 시뮬레이션 하였다.

MR-트리를 구현하고 연산 누적 시간을 측정하기 위해 2차원 사각형 데이타를 가지고 연산을 6만 번 연속 실행 하였을 때 누적 연산 시간 추이를 살펴보았다. 시간은 ms 단위이다. 플래시 메모리 상에서 MR-트리를 구현하기 위해 MR-트리 구현과 함께 플래시 접근 시간(Access time)을 이용하였다.

그림 3은 각 노드 크기별로 6만 번의 삽입 연산을 수행 했을 때의 연산 누적 시간을 나타낸다. 그래프의 X축은 삽입 연산 종류를 나타내고 Y축은 연산으로 걸리는 시간의 누적 값을 나타낸다. 범례의 숫자는 노드 크기를, 지연 연산을 실행한 경우를 delay로 표시하였다. 삽입 시 들어가는 사각 형 데이타의 크기는 데이타 위치 정보 및 노드 포인터를 포함하여 40 바이트이며 연속적인 삽입 시 플래시 메모리에 그대로 접근할 때와 임의 크기만큼 지연 연산하여 플래시 메모리에 접근할 때의 시간 추이를 그래프로 나타내었다. 삽입 연산 시 노드 크기별 차이는 미미하지만 지연 연산 수행의 누적 시간이 현저히 줄어들며, 지연 연산 중에서 512 바이트의 크기에 따른 지연 연산한 것이 가장 좋은 성능을 보임을 확인할 수 있다.

그림 4는 각 노드 크기별 6만 번의 검색 연산을 수행했을 때의 연산 누적 시간을 나타낸다. 이 그래프 역시 X축은 삽입 연산 종류를, Y축은 연산 누적 시간을 나타낸다.

검색 속도는 삽입 연산에 비해 노드 크기별 차이가 나타나는데 256 바이트와 512 바이트일 때가 128 바이

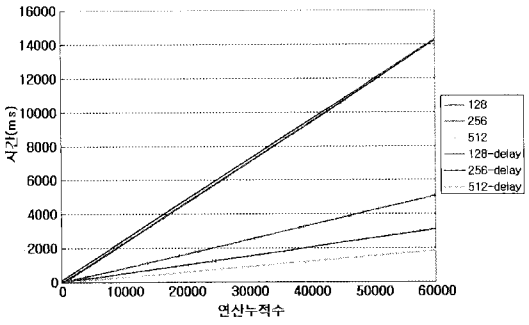

그림 3 노드 크기 별 삽입 연산 및 지연 연산 누적시간

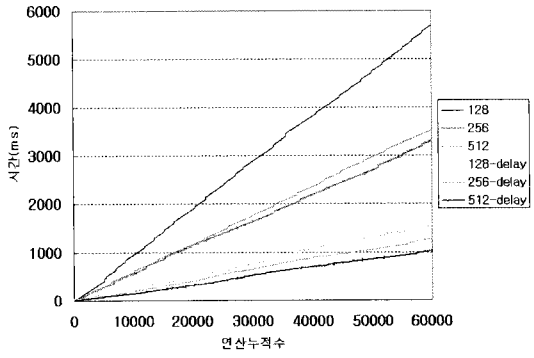

그림 4 노드 크기 별 검색 연산 및 지연 연산 누적시간

표 2 프로파일별 플래시 액세스 횟수

프로파일	지연연산 여부	액세스 횟수			감소율(%)
		검색	삽입	삭제	
Read Oriented	X	17252	3012	72325	
	O	7956	878	30215	46
Write Oriented	X	3426	23589	73695	
	O	854	6678	30215	25
Delete Oriented	X	6251	5697	422325	
	O	1852	1678	172215	30

트일 때에 비해 검색 속도가 좋은 것을 확인할 수 있다. 이는 [1]에서도 확인할 수 있으며, 2차원 사각형 데이타의 검색 연산에 대한 플래시 접근을 임의 노드 크기에 맞춰 지연 연산했을 시 각 노드별 누적 시간이 지연 전보다 줄어든 것을 확인할 수 있다.

그림 4의 결과에서 보듯 노드 크기 중 플래시 메모리의 섹터 크기인 512 바이트에 맞추어 지연 연산 수행 시 전체 연산 누적 시간이 가장 많이 줄어드는 성능 향상을 확인할 수 있다. 또한 MR-트리의 노드 크기를 섹터크기에 맞추게 되면 검색성능 향상과 더불어 한 노드 그대로 플래시 메모리에 적용할 수 있어 쓰기, 소거 연산에 효율적이며 노드 정보 관리에 적합하다.

표 2는 512MB의 플래시 메모리 상에서의 지연 연산 여부에 따른 액세스 횟수 변화를 연산 프로파일 별로 비교한 것으로 총 1 만 번의 연산 중 검색, 삽입, 삭제 위주의 프로파일을 작성하여 검색위주의 프로파일의 경우 검색 연산이 8000번, 삽입, 삭제가 각각 1000번씩 실행하는 방식으로 구성하였다. 연산 시 일어나는 플래시 액세스 횟수를 분석하고 지연 연산을 실행했을 때와 비교한 결과 25~46퍼센트의 감소율을 나타냈다. 지연 연산 실행으로 액세스 횟수를 줄여줌으로써 플래시 상의 쓰기와 소거 연산 횟수를 감소시킴을 알 수 있다.

6. 결론 및 향후 연구

본 논문에서는 플래시 메모리 상에서의 효율적인 데이타 접근을 위한 인덱스 기법과 관련하여 공간 인덱스 자료구조로써 검색 성능을 향상시킨 MR-트리의 활용을 제안하고 각 노드 크기별 MR-트리의 특징을 살펴보고, 플래시 메모리에 적용하기 위하여 지연 연산 기법을 제안하였다. 플래시 메모리의 읽기, 쓰기 단위인 섹터의 크기와 같은 512 바이트의 노드 크기를 MR-트리 지연 연산에 적용하였을 때 가장 좋은 성능을 보였다. 삽입, 검색 등의 연산 시 일어나는 데이타를 플래시 메모리에 적용할 때 섹터 크기만큼 모아서 지연 연산하는 것이 쓰기 연산으로 일어나는 추가 비용을 줄이는 성능 향상을 확인할 수 있었다.

섹터 크기인 512 바이트와 같은 크기의 노드 크기로 MR-트리를 최적화 하게 되면 MR-트리의 검색 연산에도 좋은 성능을 보이며 또한 플래시 메모리에 노드별로 쓰기가 가능해 노드 관리에 적합하고 쓰기 연산 및 소거 연산에서 효율적이다. 이는 플래시 메모리의 쓰기와 소거 연산을 최소화함으로써 추가 비용을 줄여 단점을 최소화하고 수명소모를 늦출 수 있다.

향후 연구로 플래시 메모리에 쓰기 연산이 집중될수록 성능 저하 및 플래시 메모리의 수명 저하를 가져올 수 있음을 확인하는 작업과, MR-트리의 노드 크기를 섹터 크기에 맞추고 요구 되는 페이지를 고려하여[7] 플래시 메모리의 웨어레벨링의 성능 향상시킬 수 있는 MR-트리 최적화를 연구할 계획이다.

참 고 문 헌

[1] Kyung-Chang Kim and Suk-Woo Yun, MR-Tree : A cache-conscious main memory spatial index structure for mobile GIS, Web and wireless geographic information systems, The 4th international workshop (W2GIS 2004), pp. 167-180, 2004.

[2] Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo, An efficient B-tree layer for flash memory storage systems, The 9th international conference on Real-Time and Embedded Computing systems and Applications (RTCSA), 2003.

[3] Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo, An efficient R-tree implementation over flash-memory storage systems, Proceeding of the 11th ACM international symposium on Advances in geographic information systems, 2003.

[4] 남정현, 박동주, 플래시 메모리 상에서 B-트리 설계 및 구현, 정보과학회논문지, 제34권, 제2호, pp. 109-118, 2007.

[5] Chanik Park, Jeong-Uk Kang, Seon-Yeong Park, and Jin-Soo Kim, Energy aware demand paging on Nand Flash-based embedded storages, Proceedings of IEEE/ACM ISLPED, 2004.

[6] Eran Gal and Sivan Toledo. Algorithms and data structures for flash memories, ACM Computing Surveys(CSUR), pp. 138-163, 2005.

[7] SAMSUNG NAND flash SLC-small block, http://www.samsung.com/global/business/semiconductor/productInfo.do?fmlyid=158&partnum=K9F1208R0C&&ppmi=1157, 2007.