논문 2008-45CI-3-10

플래시 기반 임베디드 DBMS의 전력기반 질의 최적화를 위한 비용 모델

(Cost Models of Energy-based Query Optimization for Flash-aware Embedded DBMS)

김 도 윤*, 박 상 원**

(Do Yun Kim and Sangwon Park)

요 약

임베디드 시스템에서 데이터베이스의 사용이 증가하고 있으며 이의 임베디드 시스템의 저장 장치로 낸드 플래시 메모리가 널리 사용되고 있다. 기존 데이터베이스 시스템의 질의 처리기의 최적화기는 저장 시스템을 디스크로 가정하고 설계되어 있다. 플래시 메모리는 디스크와는 달리 덮어 쓰기 연산을 하기 위해서는 기존 블록을 소거한 후 쓰기 연산을 해야하는 부담이 있다. 그러므로 기존 디스크 기반의 질의 최적화기는 임베디드 시스템에 적합하지 않다. 특히 임베디드 시스템은 전력 소모량을 최소화해야 하나 플래시에서의 빈번한 쓰기 연산은 추가적인 소거 연산으로 인한 전력 소비를 증진시킨다. 본 논문은 임베디드 데이터베이스에서 전력 기반 비용 모델을 새롭게 제시하고, 디스크 기반 비용 모델과 비교하여 제시한 비용 모델과의 차이를 보인다.

Abstract

The DBMS are widely used in embedded systems. The flash memory is used as a storage device of a embedded system. The optimizer of existing database system assumes that the storage device is disk. There is overhead to overwrite on flash memory unlike disk. The block of flash memory should be erased before write. Due to this reason, query optimization model based on disk does not adequate for flash-aware database. Especially embedded system should minimize the consumption of energy, but consumes more energy because of excessive erase operations. This paper proposes new energy based cost model of embedded database and shows the comparison between disk based cost model and energy based cost model.

Keywords : flash memory, DBMS, query optimization, cost model

Ⅰ. 서 론

최근 디지털 카메라, MP3 플레이어, 핸드폰 등의 사용량이 급증해 짐에 따라 이동성이 중요한 요소로 차지하는 기기들이 많이 등장하였다. 이에 따라 소형화, 대용량화, 저 전력화, 비휘발성, 고속화 그리고 충격에 강한 메모리가 필요하게 되었다. 이러한 요구사항을 잘 충족하는 메모리는 많은 응용에서 디스크를 대체할 것으로 예상된다.

플래시 메모리는 특정 블록에 쓰기 연산을 하기 전에 해당 블록을 미리 소거(erase-before-write)해야 하는 제약이 있다. 이에 따라 플래시 메모리에 저장된 데이터를 갱신하려면 다른 블록에 원래의 데이터를 복사하여 회피해 두고 변경된 데이터를 기록한 후 변경되지 않는 기존 데이터를 옮기는 작업이 필요하다. 또한 각 블록은 최대 10만번 정도의 소거 연산을 수행할 수 있기 때문에 각 블록에 대한 소거 횟수를 평준화하는 기

* 정회원, 한국외국어대학교 컴퓨터및정보통신공학과
(Dep. of Computer Science & Information Communications Engineering Hankuk University of Foreign Studies)
** 평생회원, 한국외국어대학교 컴퓨터및정보통신공학부
(Computer Science & Information Communications Engineering Division Hankuk University of Foreign Studies)
※ 본 논문은 2008년 한국외국어대학교 교내 학술연구지원비의 지원을 받아 연구되었음.
접수일자: 2008년4월26일, 수정완료일: 2008년5월6일

표 1. 플래시 메모리의 예(Intel 28F640J3A)
Table 1. Example of flash memory.

Read access time	100~150ns
Writing time using buffer Erasing and writing a block	218μs /32bytes 0.8 sec/block
Erasing block time	1.0 sec/block
Erase limit	100,000 times
Power consumption	Standby power : 0~120μA Operating power : 5~70mA

법이 필요하다. 이러한 단점으로 인해 플래시 메모리 시스템의 성능이 저하된다. 표 1은 플래시 메모리의 예를 보여주고 있다.

이와 같은 플래시 메모리의 단점을 해결하기 위하여 플래시 변환 계층(Flash Translation Layer; FTL)이 필요하다. 최신의 멀티미디어 응용 제품에서는 여러 가지 계층화된 형태의 플래시 메모리 기반 소프트웨어가 필요하다. 플래시 메모리 기반 소프트웨어의 핵심인 플래시 변환 계층과 파일시스템 뿐만 아니라 대량의 멀티미디어 콘텐츠를 효율적으로 처리하기 위한 임베디드 DBMS 처리 기술이 모두 요구된다. 앞으로 많은 이동형 기기에 임베디드 데이터베이스가 사용될 것이다. 이러한 임베디드 시스템은 저전력, 고성능, 소형화되어야 하며 이를 위해 대부분의 임베디드 시스템의 저장장치는 플래시 메모리가 사용될 것이다.

이처럼 플래시 메모리는 휴대용 기기의 저장 매체로 각광받고 있으며 플래시 메모리의 용량이 커지면서 기존의 하드 디스크를 대체하고 있다. 배터리로 구동되는 PDA나 휴대 전화 그리고 휴대용 미디어 재생기와 같은 기기의 경우 기기가 사용할 수 있는 전기 에너지 자원이 배터리에 한정되어 있기 때문에 전력 소모를 줄이는 것은 매우 중요한 요소라고 할 수 있다[1]. 또한 하드 디스크는 읽기와 쓰기 연산의 비용이 동일하지만 플래시 메모리는 읽기에 비해서 쓰기의 비용이 훨씬 많이 든다. 뿐만 아니라 하드 디스크는 덮어쓰기(overwrite) 연산이 가능하지만 플래시 메모리는 덮어쓰기를 할 수 없다. 이러한 요소들은 데이터베이스의 전력 소비에 대한 비용 모델이 변경되어야 함을 의미한다. 본 논문에서는 플래시 메모리에 적합한 전력 소비에 대한 새로운 질의 비용 모델을 제시하고 이를 이용한 질의 최적화 방법을 제시한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 플래시 메모리와 FTL[2]에 대해 살펴보고, Ⅲ장에서는 플래시 메모리의 비용 계산에 대해서 설명한다. Ⅳ장에서는 플래시 메모리에서의 조인 비용 모델을 제시하고 Ⅴ장은 질의 처리 시뮬레이션 시스템에 대해 설명한다. Ⅵ장에서 실험을 통한 결과를 분석하고, Ⅶ장에서 결론과 향후 연구에 대해 살펴본다.

Ⅱ. 관련 연구

1. 플래시 메모리 시스템

최근 플래시 메모리는 휴대용 장비에 필수적으로 탑재되는 메모리가 되었다. 플래시 메모리는 기존의 하드 디스크에 비해 휴대성에 맞는 특징을 가지고 있다. 플래시 메모리는 데이터에 대한 신뢰성, 갱신의 용이성, 낮은 전력 소모, 전원 차단시의 안정성이 큰 특징이다. 플래시 메모리는 읽기, 쓰기와 소거 연산의 수행 시간이 다르며 읽기와 쓰기는 2K 바이트의 페이지(page) 단위로 수행되고, 소거의 경우 128K 바이트의 소거 유닛(erase unit) 단위로 수행된다. 또한 플래시 메모리는 쓰기를 수행하기 위해서 소거 연산이 선행되어야 한다. 또한 각 소거 유닛은 약 10만번 정도 소거가 가능하고, 이보다 많은 소거 연산이 발생하게 되면 데이터의 무결성을 보장하지 못한다. 때문에 소거 횟수 평준화(wear-leveling) 기법이 필요하다[3]. 플래시 메모리는 이러한 단점을 보완해주는 소프트웨어 모듈이 필요하며, 이를 플래시 변환 계층(FTL)이라 한다.

그림 1은 플래시 메모리 시스템에 대한 구조도이다. FTL은 플래시 메모리와 파일 시스템 사이에 존재하je 파일 시스템이 플래시 메모리를 논리적 수준에서 디스크와 같은 블록 디바이스로 바라볼 수 있게 해준다[4]. FTL 알고리즘은 파일 시스템으로부터 전달받은 섹터 번호와 길이를 가지고 효율적인 사상(mapping)을 통해 전체적으로 읽기, 쓰기, 소거연산을 줄이고 블록 소거 연산 회수를 평준화 시킨다.

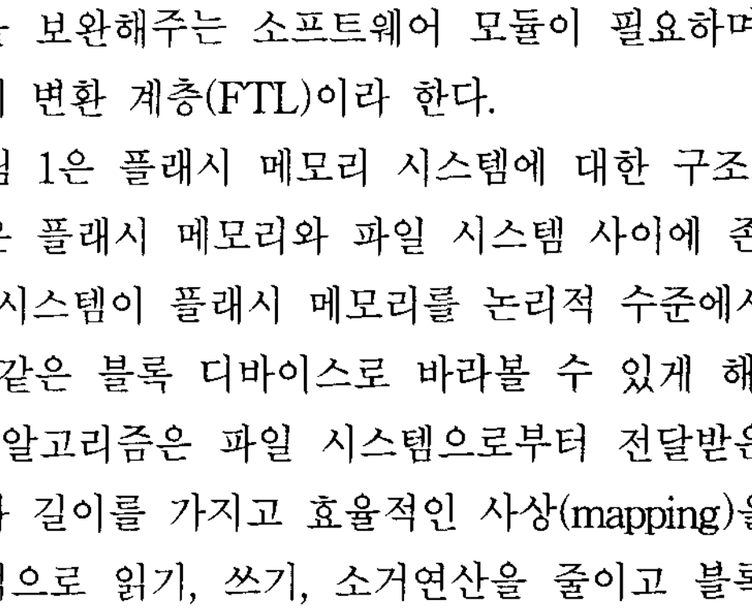

그림 1. 플래시 메모리 시스템 구조
Fig. 1. System architecture of flash memory.

2. 플래시 변환 계층

FTL 알고리즘에서 사용되는 사상 방법은 섹터 사상(sector mapping), 블록 사상(block mapping), 하이브리드 사상(hybrid mapping)으로 나누어진다[5]. 그림 2에서와 같이 섹터 사상은 물리적인 섹터 번호와 논리적인 섹터 번호를 사상하는 것이고, 그림 3에서와 같이 블록 사상은 물리적인 블록 번호와 논리적인 블록 번호를 사상하는 것이다. 섹터 사상을 하면 한 번의 사상 테이블의 검색을 통해 실제 플래시 메모리의 물리적인 섹터 번호를 알 수 있어 매우 빠르게 동작 할 수 있지만 플래시 메모리의 용량이 증가하는 만큼 매우 많은 양의 램을 사용하게 된다. 블록 사상은 플래시 메모리의 블록 수만큼의 사상이 필요하여 섹터 사상보다 상대적으로 적은 양의 램을 사용하지만 블록 내에서의 덮어 쓰기 연산을 효율적으로 수행할 수 없다. 하이브리드 사상은 두 가지 사상을 동시에 사용하는 것으로 전체 블록에 대해서는 블록 사상을 하여 램의 사용량을 줄이고 블록 내에서는 섹터 사상을 하여 두 가지 사상의 장점을 모두 취할 수 있다.

플래시 메모리에서 한 개의 블록 내에 있는 섹터들에 대해 기록하는 방법을 두 가지로 나뉘는데 한 블록에 저장된 섹터들이 정해진 특정 위치에 고정되는 것을 고정 섹터 방식(in-place)이라 부르며, 섹터들이 블록 내의 임의의 위치에 존재하는 것을 변동 섹터 방식(out-of-place)라 부른다.

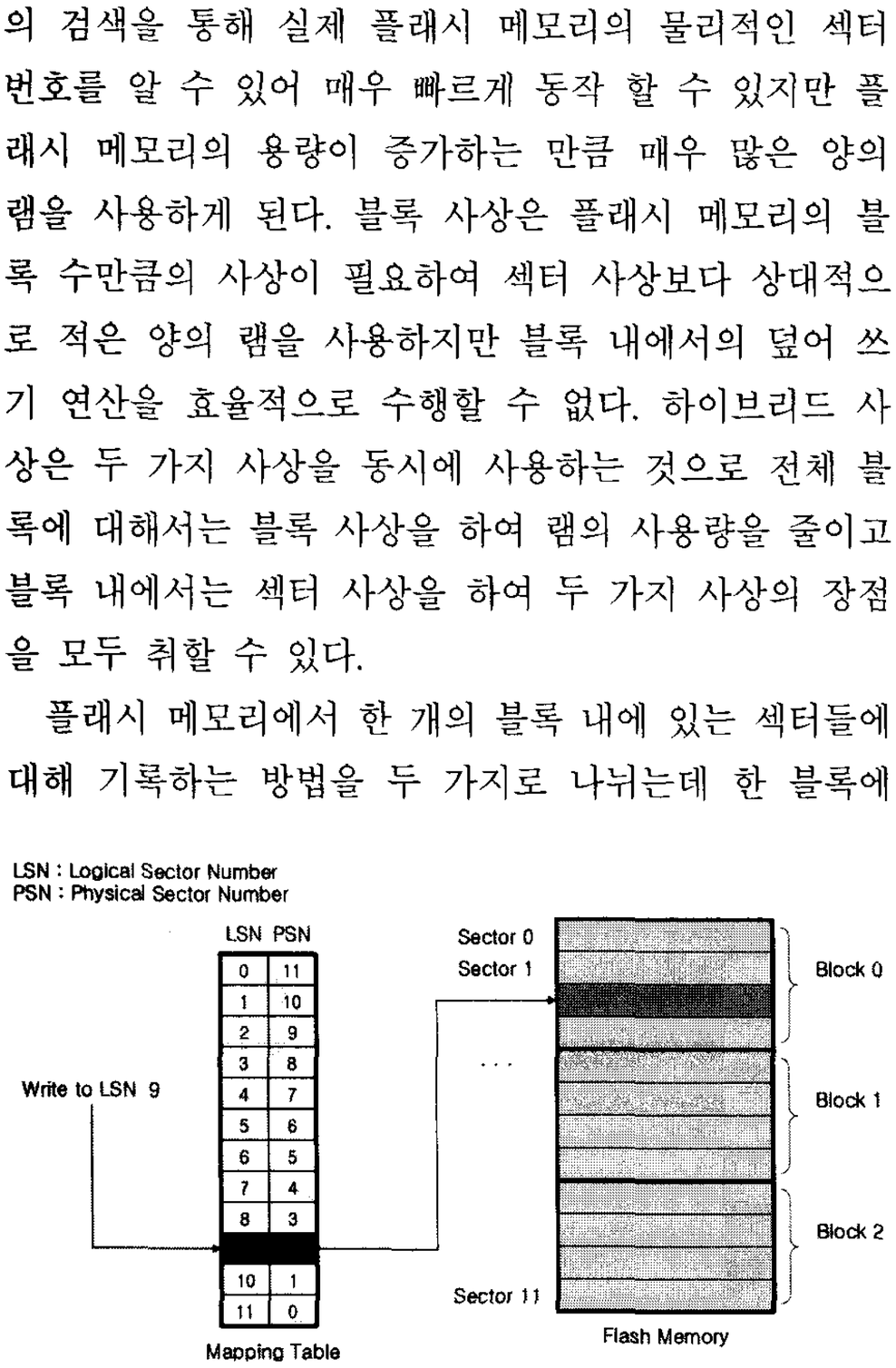

그림 2. 섹터 사상
Fig. 2. Sector mapping.

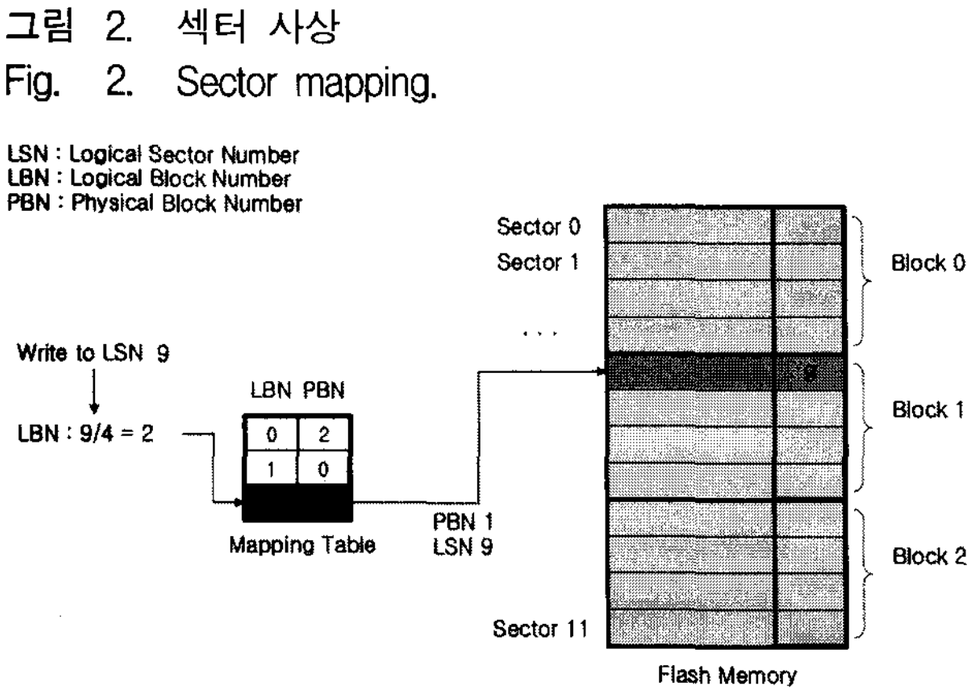

그림 3. 블록 사상
Fig. 3. Block mapping.

E_r : 플래시 메모리 한 개의 페이지를 읽을 때 드는 에너지
E_w : 플래시 메모리 한 개의 페이지를 쓸 때 드는 에너지
E_e : 플래시 메모리 한 개의 페이지를 소거할 때 드는 에너지

FTL은 매핑 방법이나 물리적인 위치에 쓰는 방식에 따라 표 2와 같은 방법들로 분류할 수 있다. 표 2에서는 FTL 중 복사블록 기법[6], 여유공간 기법[7]과 로그블록 기법[8]의 특징을 나타내고 있다.

표 2. FTL 알고리즘의 특성
Table 2. Characteristics of FTL algorithms.

	복사블록 기법	여유공간 기법	로그블록 기법
데이터 블록	고정섹터방식	고정섹터방식	고정섹터방식
사상 방법	블록 사상	블록 사상	혼합 사상
덮어 쓰기	고정섹터방식 변동섹터방식	변동섹터방식	로그 영역의 변동섹터방식
특징	각 블록 당 복사블록	블록내의 여유 공간	로그 블록

Ⅲ. I/O에 대한 추가 비용 비율

하드 디스크에서는 데이터베이스의 I/O 하나당 디스크에 하나의 I/O가 발생한다. 하지만, 플래시 메모리에서는 하나의 데이터베이스 페이지 I/O 요청에 추가적인 연산이 발생될 수 있다. 그러므로 플래시 메모리에서 질의 처리를 위한 실제 비용을 계산하는 방법이 필요하다[9].

$$k = \frac{S_P}{S_{fp}\, m} \tag{1}$$

한 개의 데이터베이스 페이지의 I/O에 대한 플래시 메모리에 요청되는 플래시 메모리 페이지 개수 k를 나타낸 것이다. S_P는 데이터베이스의 한 페이지 크기(KB)이고, S_{fp}는 플래시 메모리의 한 페이지의 크기(KB)이다. m은 플래시 메모리에서 칩 인터리빙 (chip-interleaving)의 수를 나타낸다. 일반적으로 S_{fp}는 2KB이다.

칩 인터리빙은 그림 4와 같이 여러 개의 플래시 메모

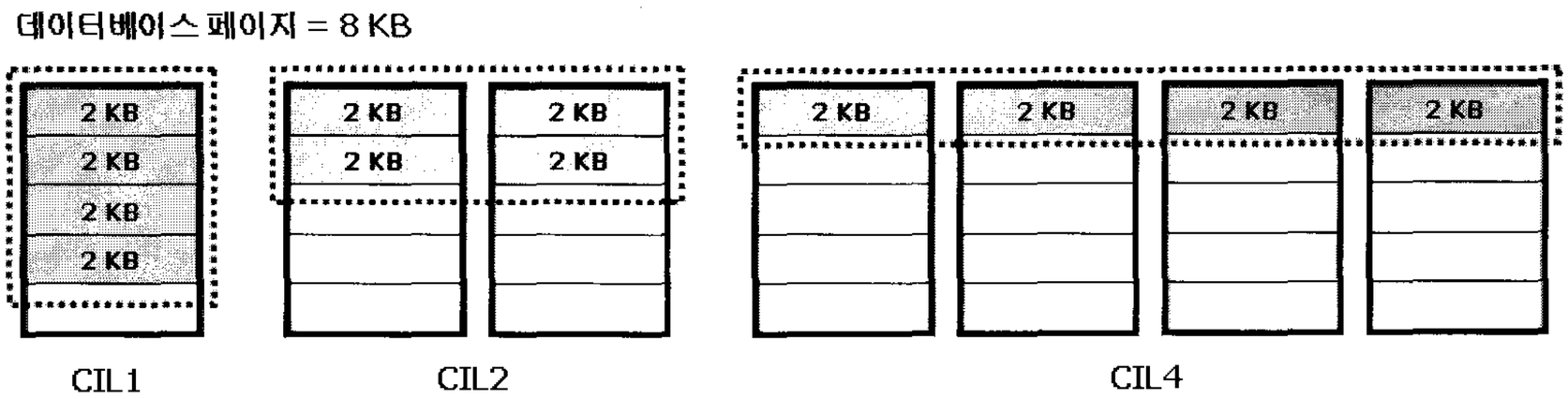

그림 4. 칩 인터리빙 개념도
Fig. 4. Concepts of chip interleaving.

리를 병렬로 연결하여 읽기와 쓰기에 대한 대역폭을 늘리는 방법이다. 예를 들어 칩 인터리빙이 1인 경우 한 번의 읽기와 쓰기에 대한 연산을 칩에 전달하고 칩 인터리빙이 2이면 칩 2개로 인터리빙한 것이다. 따라서 칩 인터리빙이 4인 경우 칩 4개로 동시에 명령어를 보내어 8KB 데이터를 한 번의 연산 시간으로 읽기 혹은 쓰기가 가능하다. 예를 들어 일반적인 데이터베이스의 한 페이지(S_P)가 8 KB이고 플래시 메모리에서 칩 인터리빙(m)이 1이면 k는 4이다. 즉 8 KB의 한 블록을 쓰기 요청 하면 플래시 메모리에서는 4개의 페이지를 쓴다. 다음은 읽기와 쓰기 연산에 대한 추가적인 비용의 비, λ와 μ를 구하는 수식을 보여준다.

$$p_r = k E_r \tag{2}$$

$$P_r = \sum_{i=1}^{n} p_{r_i} = k\, n_{dr} E_r \tag{3}$$

$$\begin{aligned} P_r^{'} &= n_{rr}E_r + n_{rw}E_r + n_{re}E_e \\ &= n_{rr}E_r \\ &\because n_{rw}, n_{re} = 0 \end{aligned} \tag{4}$$

수식 2는 데이터베이스의 한 페이지를 읽을 때 플래시 메모리에서 소요되는 이상적인 비용(p_r)을 나타낸다. 수식 3은 데이터베이스에서 n_{dr}번의 읽기 연산이 일어날 때 최적의 비용(P_r)를 의미한다. 하지만 플래시 메모리는 FTL에 따라서 추가적인 읽기 비용(n_{rr})까지 고려해야 하기 때문에 플래시 메모리에서 실제 소요된 비용은 수식 4와 같다. 읽기는 추가적인 쓰기(n_{rw})나 소거(n_{re})가 일어나지 않기 때문에 수식 4와 같은 식이 도출된다.

$$\lambda = \frac{P_r^{'}}{P_r} = \frac{n_{rr}E_r}{n_{dr}p_r} = \frac{\frac{n_{rr}}{n_{dr}}E_r}{p_r} \tag{5}$$

위 수식들을 이용해서 수식 5의 최적 비용에 대한 실제 비용의 비(λ)를 구할 수 있다. 즉, λ는 데이터베이스의 n_{dw}번의 페이지 읽기 연산을 처리하는데 발생한 비용($P_r^{'}$)을 최적 예상 비용(P_r)으로 나누어 그 비율을 나타낸 것이다. 예를 들어 데이터베이스 100개의 페이지에 대한 읽기 요청했을 때 플래시 메모리에서 500페이지의 읽기가 발생하였다면 λ는 ((500/100)×80)/(4×80) = 1.25이다.

$$p_w = kE_w \tag{6}$$

$$P_w = \sum_{i=1}^{n} p_{w_i} = k n_{dw} E_w \tag{7}$$

$$P_w^{'} = n_{wr}E_r + n_{ww}E_w + n_{we}E_e \tag{8}$$

$$\begin{aligned} \mu = \frac{P_w^{'}}{P_w} &= \frac{n_{wr}E_r + n_{ww}E_w + n_{we}E_e}{n_{dw}p_w} \\ &= \frac{\frac{n_{wr}}{n_{dw}}E_r + \frac{n_{ww}}{n_{dw}}E_w + \frac{n_{we}}{n_{dw}}E_e}{p_w} \end{aligned} \tag{9}$$

읽기와 마찬가지로 쓰기에 대한 추가적인 비용의 비(μ)를 구할 수 있다. 수식 8에서 플래시 메모리에 실제 소요된 비용에는 FTL에 따라 추가적인 읽기(n_{wr}), 쓰기(n_{ww})와 소거(n_{we})가 포함된다. 예를 들어 데이터베이스 100 페이지에 대한 쓰기 요청을 했을 때 플래시 메모리에서 200, 750, 50 페이지의 읽기, 쓰기, 소거가 발생했다고 가정하면 μ는 수식 9에 의해 ((200/100)×80+(750/100)× 200+(50/100)×1500)/(4×200)=4.7875이다.

이때 λ와 μ는 p_r과 p_w의 영향을 받는다. p_r과 p_w값은 칩 인터리빙에 따라 달라지는 값으로, 하드웨어 성능을

표 3. 플래시 메모리에서의 λ와 μ (CIL=1)
Table 3. λ and μ of flash memory.

FTL	λ	μ
복사블록 기법	1.66	17.86
여유공간 기법	34.21	23.7
로그블록 기법	1.01	10.29

고려한 값이 된다. 즉, λ와 μ는 데이터베이스의 한 페이지를 I/O하는데 플래시 메모리에서 드는 비용을 비율로 나타낸 값이다. 표 3은 칩 인터리빙이 1일 때 각 FTL에서의 λ와 μ를 나타낸 것이다. 이는 TPC-A 벤치마크를 MySQL[10]에서 수행하여 얻어낸 트레이스 A를 5장의 FTL 시뮬레이터에서 수행하여 구한 각 FTL에 대한 평균적인 λ와 μ이다.

Ⅳ. 플래시 메모리에서 전력기반 조인 비용 모델

디스크 기반 비용 모델[11]은 I/O 횟수를 토대로 정의하지만, 임베디드 DBMS에서의 플래시 전력기반 비용 모델은 읽기, 쓰기와 소거에 따른 전력 소모에 따라서 정의해야 한다. 질의 최적화를 위한 가장 큰 비용은 조인 연산이다. 따라서 조인 연산의 플래시 전력기반 비용은 새롭게 정의할 필요가 있다. 이를 위한 변수는 다음과 같다.

E_{disk}	:	디스크 기반 비용
E_{flash}	:	플래시 전력기반 비용
E_{CPU}	:	질의 처리하는데 필요한 CPU 에너지
B	:	데이터베이스 페이지의 크기(bytes)
M	:	버퍼의 크기(데이터베이스 페이지의 수)
b_r, b_s	:	r, s 릴레이션의 데이터베이스 페이지 수
n_r, n_s	:	r, s 릴레이션의 데이터베이스 레코드 수

이 때 디스크 기반 비용 모델에 따라서 플래시 전력기반 비용 모델을 정의하기 위해 플래시 메모리에서 데이터베이스 페이지를 I/O하는 비용을 계산할 필요가 있다.

E_{rB}	: 플래시 메모리로부터 데이터베이스 페이지 크기의 데이터를 읽을 때 드는 에너지
E_{wB}	: 플래시 메모리로 데이터베이스 페이지 크기의 데이터를 쓸 때 드는 에너지

위와 같이 정의할 때 하나의 데이터베이스 페이지를 플래시 메모리에 읽고 쓰는 비용은 수식 10과 같다. 디스크의 경우 두 비용이 같지만 플래시 메모리에서는 추가적인 비용이 발생하기 때문이다.

$$E_{rB} = kp_r = k\lambda E_r$$
$$E_{wB} = kp_w = k\mu E_w \quad (10)$$

위의 변수를 이용하여 대표적인 네 가지 조인방법인 블록 중첩 반복 조인, 색인된 중첩 반복 조인, 합병 조인과 해시 조인의 플래시 전력기반 비용 모델을 제시한다[12].

1. 블록 중첩 반복 조인

$$E_{disk}(bnlj) = b_r b_s + b_r + E_{CPU}$$
$$(\text{if } M > b_r \text{ or } M > b_s,$$
$$then\ E_{disk}(bnlj) = b_r + b_s + E_{CPU}) \quad (11)$$

$$E_{flash}(bnlj) = E_{rB}(b_r b_s + b_r) + E_{CPU}$$
$$(\text{if } M > b_r \text{ or } M > b_s,$$
$$then\ E_{flash}(bnlj) = E_{rB}(b_r + b_s) + E_{CPU}) \quad (12)$$

블록 중첩 반복 조인(bnlj, block nested-loop join)은 내부 릴레이션의 모든 블록이 외부 릴레이션의 모든 블록과 쌍을 이루도록 중첩 반복하여 두 릴레이션을 읽고 조인을 한다. 수식 11은 블록 중첩 반복 조인의 디스크 기반 비용 모델이고 수식 12는 플래시 전력기반 비용 모델이다. 블록 중첩 반복 조인은 $b_r b_s + b_r$번의 읽기 연산이 필요하다. 따라서 플래시 전력기반 비용은 $b_r b_s + b_r$번의 데이터베이스 페이지 당 몇 개의 플래시 메모리 페이지를 읽는지를 계산하여 정의한다. 따라서 수식 12처럼 디스크 기반 비용 모델에 E_{rB}를 곱해서 구할 수 있다.

2. 색인된 중첩 반복 조인

$$E_{disk}(inlj) = b_r + n_r d_s + E_{CPU},$$
$$d_s = \lceil \log_{f_B} n_s \rceil \quad (13)$$

$$E_{flash}(inlj) = E_{rB}(b_r + n_r d_s) + E_{CPU} \quad (14)$$

한쪽 릴레이션에 조인하는 속성에 대해 색인이 존재하는 경우 색인된 중첩 반복 조인(inlj, indexed nested-loop join)을 수행할 수 있다. 색인된 중첩 반복 조인은 s 릴레이션에 색인이 존재한다고 가정한다. 수식 13은 외부 릴레이션인 r 릴레이션을 읽고(b_r), r 릴레이션의 하나의 레코드에 해당하는 조인 결과를 찾기 위해 색인을 검색하는 디스크 기반 비용이다. 조인의 결과를 찾는 비용은 s 릴레이션에 구축된 B+트리의 깊이(d_s)만큼 찾으므로 $n_r \times d_s$이다. 이때 f_B는 B+트리의 한 노드에 저장할 수 있는 엔트리의 크기이다. 플래시 전력기반 비용인 수식 14의 비용은 수식 13에 E_{rB}를 곱한 것이다.

3. 합병 조인(mj, merge join)

$$E_{disk}(mj) = b_r + b_s + E_{sort-disk} + E_{CPU}$$

$$E_{sort-disk} = 2b_r(\lceil \log_{M-1}\frac{b_r}{M} \rceil + 1) + 2b_s(\lceil \log_{M-1}\frac{b_s}{M} \rceil + 1) + E'_{CPU} \quad (15)$$

$$E_{flash}(mj) = E_{rB}(b_r + b_s) + E_{sort} + E_{CPU}$$

$$E_{sort-flash} = b_r(E_{rB} + E_{wB})(\lceil \log_{M-1}\frac{b_r}{M} + 1 \rceil) + b_s(E_{rB} + E_{wB})(\lceil \log_{M-1}\frac{b_s}{M} + 1 \rceil) + E'_{CPU} \quad (16)$$

수식 15의 합병 조인에 대한 디스크 기반 비용 모델은 r과 s 릴레이션을 정렬하는 비용($E_{sort-disk}$)과 정렬된 결과를 조인하기위해서 읽는 비용으로 이루어진다. $E_{sort-disk}$는 런을 만들기 위해 릴레이션을 읽고 쓰는 비용과 $\lceil \log_{M-1} b_r/M \rceil$ 의 단계만큼 합병 정렬(merge sort)하는데 소요되는 읽고 쓰는 비용을 더해서 구한다. 따라서 합병 정렬에 대한 플래시 전력기반 비용($E_{sort-flash}$)은 우선 런을 생성하는데 드는 비용을 읽기와 쓰기에 대한 비용으로 분리하여 구한다. 따라서 $E_{sort-flash}$는 런을 생성하는 비용과 합병 정렬에서 각 단계에 대한 읽기와 쓰기에 대한 비용을 더해서 구할 수 있다. 최종 비용은 플래시 메모리에서 두 릴레이션을 조인하기 위해서 정렬된 결과들을 읽는 비용을 포함하여 수식 16과 같이 정의한다.

4. 해시 조인(hj, hash join)

$$E_{disk}(hj) = b_r + b_s + E_{part-disk} + E_{CPU}$$

$$E_{part-disk} = 2(b_r + b_s)(\lceil \log_{M-1} b_s \rceil - 1) + E'_{CPU} \quad (17)$$

$$E_{flash}(hj) = E_{rB}(b_r + b_s) + E_{part-flash} + E_{CPU}$$

$$E_{part-flash} = (E_{rB} + E_{wB})(b_r + b_s)(\lceil \log_{M-1} b_s \rceil - 1) + E'_{CPU} \quad (18)$$

수식 17은 합병 조인에서처럼 해시를 구축하는 비용($E_{part-disk}$)과 해싱(hashing)된 결과를 조인하기 위해서 읽는 비용을 합하여 구한다. 릴레이션의 크기가 커서 재귀 분할(recursive partitioning)이 필요한 경우 분할의 각 단계에서 분할 영역의 크기를 M-1의 비율로 줄이게 된다. 따라서 분할에 필요한 단계의 수는 $\lceil \log_{M-1} b_s \rceil - 1$이 된다. $E_{part-disk}$은 재귀 분할의 각 단계에서 릴레이션의 모든 데이터베이스 페이지를 읽고 쓰는 비용을 말한다. 분할 단계에서 비용($E_{part-disk}$)은 읽기와 쓰기에 대해서 E_{rB}와 E_{wB}를 각각 곱해서 구할 수 있다. 그리고 수식 18의 해시 조인을 위한 플래시 기반 전체 비용은 해싱된 결과를 읽는 비용을 더해서 구한다.

V. 질의 처리 시뮬레이션 시스템

본 논문에서 질의 처리 비용을 측정하기 위해 질의 처리 시뮬레이터, FTL 시뮬레이터, 플래시 임베디드 보드로 이루진 질의 처리 시뮬레이션 시스템을 구현하였다. 그림 5는 질의 처리 시뮬레이터와 실험에 필요한 모듈에 대한 전체 시스템 구조를 나타낸다.

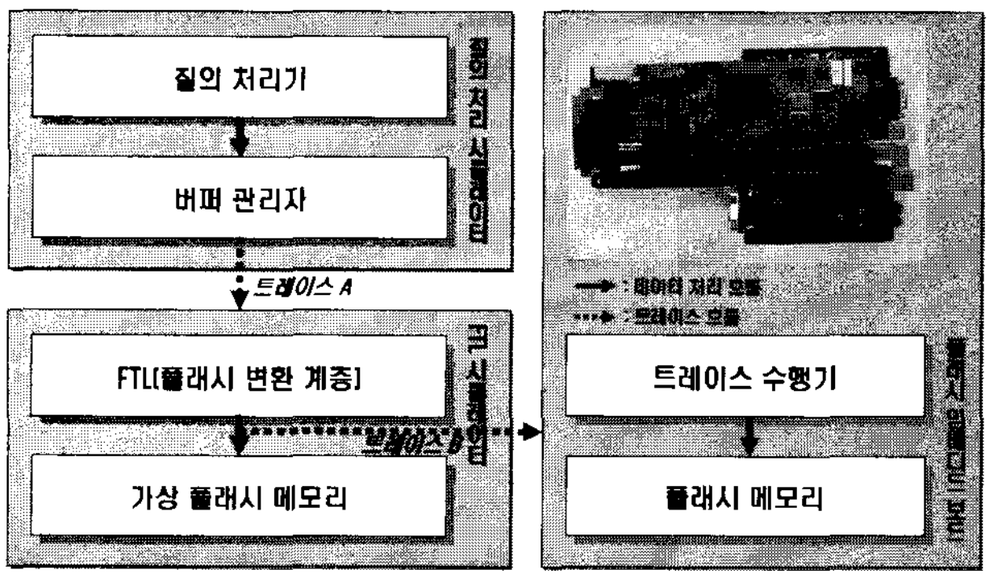

그림 5. 질의 처리 시뮬레이션 시스템 구조
Fig. 5. Architecture of query processing simulator.

1. 질의 처리 시뮬레이터

질의 처리 시뮬레이터는 데이터베이스가 수행하는 질의 처리를 가상으로 수행하는 모듈을 말한다. 여기서 처리하는 질의는 질의에 대한 비용이 가장 비싼 조인 질의로 제한한다. 그림 5에서 질의 처리 시뮬레이터는 질의 처리기와 버퍼 관리자로 구성된다. 조인 질의에 대한 질의 처리기는 버퍼 관리자를 통해 I/O를 요청한다. 여기서 사용하는 조인 질의는 블록 중첩 반복 조인, 색인된 중첩 반복 조인, 합병 조인과 해시 조인이다. 버퍼 관리자는 LRU로 동작하며 데이터베이스 페이지에 대한 읽기와 쓰기를 수행한다. 버퍼 관리자가 데이터베이스의 페이지를 읽고 쓰는 연산을 추출한 트레이스 A는 가상 플래시 메모리에 동작할 수 있도록 하는 FTL 시뮬레이터 모듈 및 플래시 임베디드 보드 모듈의 입력으로 사용된다.

2. FTL 시뮬레이터

FTL 시뮬레이터는 FTL과 가상 플래시 메모리 모듈로 구성된다. FTL 시뮬레이터는 질의 처리 시뮬레이터에서 발생된 트레이스 A를 입력으로 받아서 가상 플래시 메모리에 연산을 수행한다. FTL이 필요한 이유는 앞에서 설명한 것과 같이 질의 처리 시뮬레이터는 일반적인 DBMS와 같이 디스크를 저장장치로 간주하기 때문에 플래시 메모리에 동작할 수 있도록 하기 위해서이다. 이는 디스크는 읽기와 쓰기 연산만 존재하는 반면 플래시 메모리는 소거 연산이 필요하다는 특성이 있기 때문이다. 그리고 가상 플래시 메모리는 실제 플래시 메모리와 유사하게 동작하도록 구현하였다. FTL 시뮬레이터 결과의 정확성을 검증하기 위하여 FTL에서 가상 플래시 메모리로 보내는 연산을 추출한 트레이스 B를 플래시 임베디드 보드에서 수행하여 비교하였다. 이때 트레이스 B는 플래시 임베디드 보드의 입력이 되며 플래시 메모리가 동작하는 트레이스이다.

3. 플래시 임베디드 보드

플래시 임베디드 보드는 EDB9315A 보드와 도터 보드(daughter board), 플래시 메모리, DAQ 장비 등으로 구성된다. 플래시 임베디드 보드는 임베디드용 리눅스 커널(linux 2.6.11-1)을 설치하고, 이 커널에 플래시 메모리를 접근하여 읽기, 쓰기, 소거연산을 수행할 수 있는 시스템 콜을 추가하였다. FTL 시뮬레이터의 트레이스 B를 읽어서 임베디드 보드의 시스템 콜을 호출하는 응용프로그램을 작성하고, 이 프로그램에서 플래시 메모리에서 조인에 의해 발생된 읽기, 쓰기, 소거 연산의 전력을 측정할 수 있다. 이렇게 측정된 전력은 사용자의 입력 조건이 같은 경우 여러 조인 알고리즘 중 가장 소요시간이 짧은 알고리즘과 비교 분석하였다.

4. 전력 측정 장비

플래시 메모리에서 소요되는 시간을 측정하기 위한 계측 시스템은 DAQ(Data AcQuisition) 장비를 사용하였다. 이 장비의 시스템 구성은 그림 6과 같다. 랩뷰(LabVIEW)는 내셔널 인스트루먼트 사에서 사용자에게 프로그램 개발 환경을 제공한다. 또한 DAQ 장비는 동일 회사에서 제공하는 데이터 수집 장치이며 플래시 임베디드 보드와 연결하여 시간과 전력을 측정할 수 있다. 본 논문에서는 이와 같은 계측 시스템을 사용하여 여러 조인 알고리즘에 따른 플래시 메모리에서의 전력을 측정하였다.

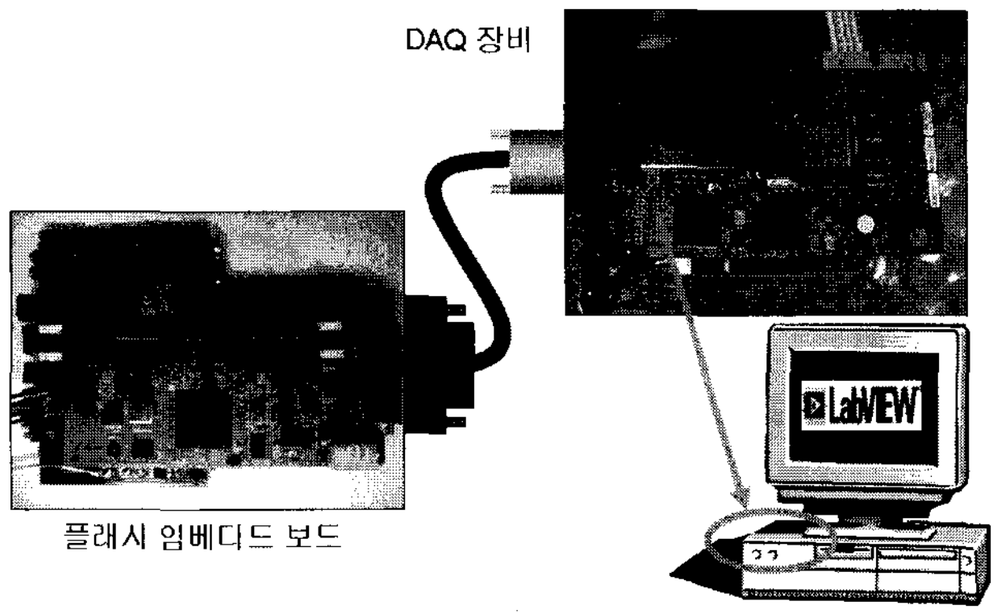

그림 6. 계측시스템 구조도
Fig. 6. Architecture of data acquisition system.

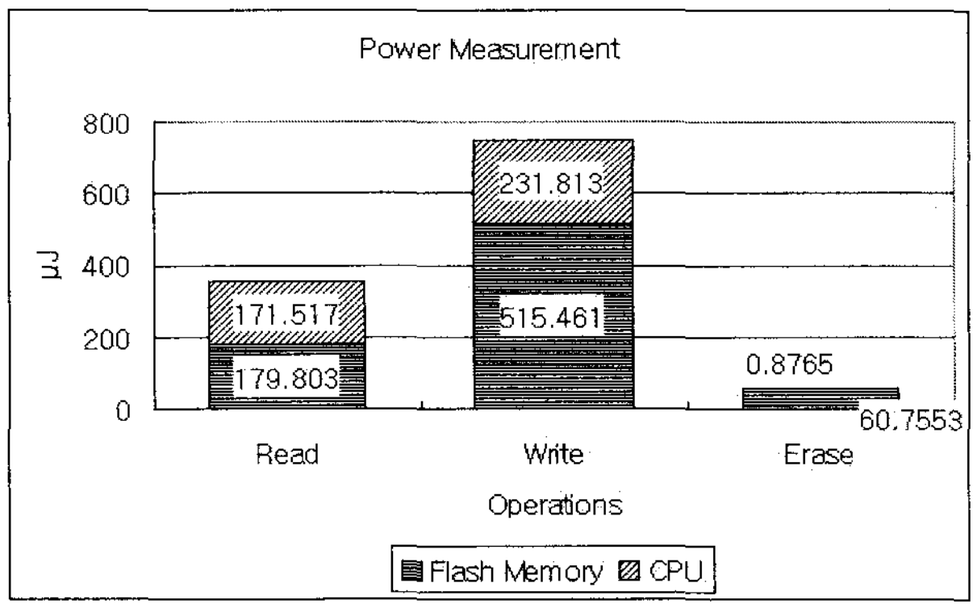

그림 7. DAQ 장비로 측정한 1개의 소거 유닛을 쓰기, 읽기, 소거한 전력
Fig. 7. Energy of read, write and erase operation on one unit measured by DAQ.

표 4. 실험 데이터
Table 4. Experimental pararmeter.

변수	데이터	단위
b_r	40	8KB
b_s	5, 20, 80, 320	8KB
M	20	8KB
S_r	256	바이트
조인알고리즘	bnlj , inlj, mj, hj	
FTL알고리즘	복사블록 기법, 여유공간 기법, 로그블록기법	
플래시 메모리 크기	r, s 릴레이션을 합한 크기의 1.25배	

그림 7은 플래시 임베디드 보드에서 플래시 메모리의 1개의 소거 유닛에 해당하는 크기의 데이터를 쓰고, 읽고, 소거했을 때의 전력을 DAQ 장비를 이용하여 측정한 결과이다. 플래시 메모리에 해당하는 부분이 플래시 메모리에서 소요된 전력이고 CPU에 해당하는 부분이 벌크램으로 플래시 메모리 결과를 받아서 운영체제로 보내는 부분이다.

VI. 실 험

1. 실험 환경

본 논문에서는 Ⅳ장의 비용 모델을 검증하기 위해서 디스크와 플래시 전력기반 비용 모델을 표 3의 실험 데이터를 토대로 비용을 측정 및 실험한다. 휴대용 장비는 제한된 메모리 자원을 가지게 되므로 적은 릴레이션 크기와 버퍼 크기를 가질 것이라고 가정한다. 실험 변수는 표 4와 같이 b_r은 40의 고정 크기를 가지고 b_s는 5에서 4배씩 증가시키며 320까지 범위를 한정한다. b_s의 변화에 따른 플래시 전력기반 비용을 예측하고 전력을 측정한다. 본 논문의 실험에서는 FTL 알고리즘별 조인을 수행한 비용을 비교한다.

플래시 전력기반 비용을 적용하기 위해 λ와 μ를 정해야 하는데 이는 TPC-A 벤치마크를 MySQL에서 동작시켜 얻어 낸 트레이스 A를 실험 시스템의 FTL 시뮬레이터에 수행하여 각 FTL에 대한 평균적인 λ와 μ를 구하였다. TPC-A 벤치마크는 TPC에서 제안된 표준 debitCredit 벤치마크이다. debitCredit 벤치마크는 데이터베이스를 사용하는 OLTP(On-Line Transaction Processing) 시스템의 성능 측정 및 평가를 위한 시험 방법 중의 하나로써 컴퓨터 업계에서 표준으로 인정받는 벤치마크이다. 실제 은행 업무를 작업 부하(workload)로 하는 debitCredit 트랜잭션을 사용하며 1초당 처리 가능한 트랜잭션 수(TPS, transaction per second)를 시스템의 처리능력으로 나타낸다[13]. TPC-A 벤치마크는 터미널과의 통신을 포함한 점대점(end-to-end) 측정을 가진 은행 텔러 어플리케이션을 모델링하는 단순한 OLTP 어플리케이션을 가정한다 [14]. 전력 기반 비용 모델을 토대로 예측한 결과를 검증하기 위해 FTL 시뮬레이터에서 가상 플래시 메모리로 보내지는 읽기, 쓰기, 소거의 횟수를 측정하여 결과를 구한다. 또한 FTL 시뮬레이터에서 추출한 트레이스 B를 플래시 임베디드 보드에서 수행하여 DAQ장비를 이용 각 조인의 소요시간을 측정한다.

2. 비용 모델을 기반으로 예측한 결과

그림 8은 디스크 기반 비용 모델에 따른 예측 결과이다. x축은 b_s이고 y축은 디스크 I/O 횟수를 나타낸다. 그림 8에서 b_s가 320블록인 경우 해시 조인이 작은 비용을 가지며, 320블록을 초과할 경우 색인된 중첩 반복 조인이 작은 비용을 가진다.

이런 디스크 기반 비용 모델을 플래시 기반 모델로 평가할 경우 그 결과는 그림 9의 예측된다. 이것은 표 2의 TPC-A 벤치마킹을 통해 얻은 λ와 μ를 통해 계산된 결과로서 각 FTL 알고리즘에 따른 조인 알고리즘별로 전력 기반 비용을 나타낸 것이다. 그림 8의 디스크 기반 비용 모델에 따른 결과와 그림 9의 플래시 전력기반 비용 모델을 통한 결과에는 차이가 있다. 복사 블록 기법과 로그 블록의 경우 s 릴레이션이 320일 때 색인된 중첩 반복 조인이 좋은 성능을 보이며, 블록 중첩 반복 조인이 해시 조인보다 좋은 결과를 보인다. λ와 μ의 값이 큰 여유공간 기법은 디스크와 비슷한 결과의 그래프 형태를 보인다. 플래시 전력기반 비용 모델의 경우 여유공간 기법을 제외하고 b_s가 80 블록일 때부터 색인된 중첩 반복 조인이 작은 비용을 보인다. 하지만 여유공간 기법의 경우 λ가 매우 크기 때문에 색인된 중첩 반복 조인이 해시 조인보다 좋지 못한 결과를 보인다. 조인 연산의 플래시 전력기반 비용 모델은 쓰기의 추가적인 비용의 비인 μ보다 읽기의 추가적인 비용의 비인 λ에 더 많은 영향을 받는다. 그러므로

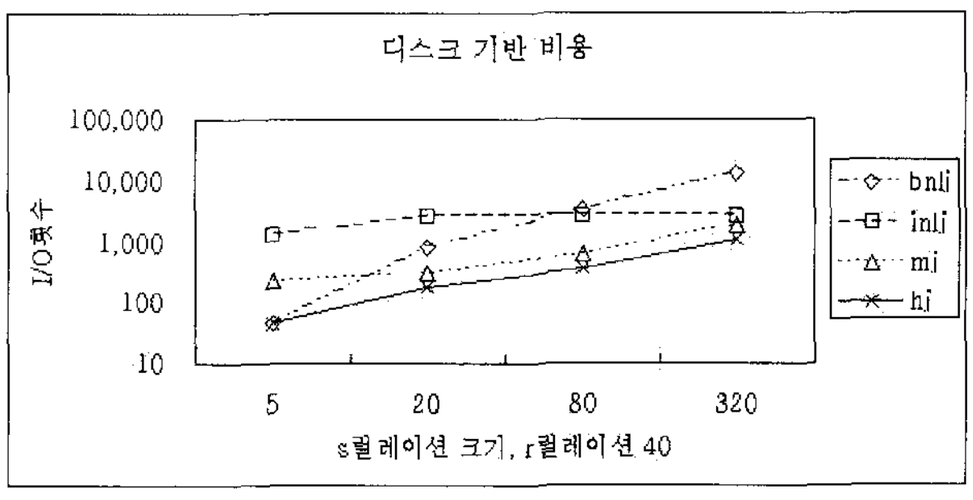

그림 8. 디스크 기반 비용
Fig. 8. Costs based on disk model.

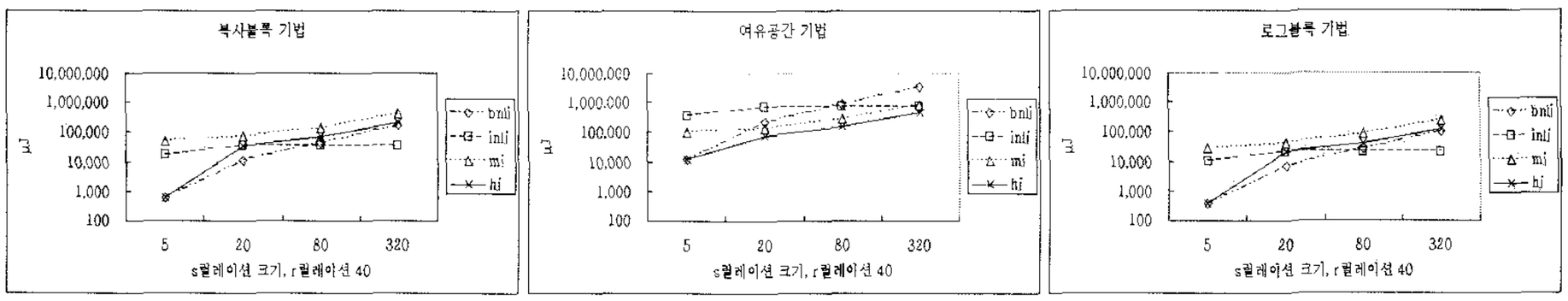

그림 9. 비용 모델을 통한 결과
Fig. 9. Experimental results by cost models.

조인에 한하여 FTL의 성능을 향상 시키는 데 μ를 줄이는 것도 중요하지만 λ를 줄이는 것이 소요시간을 줄이는데 효과적이다.

3. FTL 시뮬레이터에서의 결과

FTL 시뮬레이터를 이용해 결과를 얻기 위하여 질의처리 시뮬레이터에서 발생한 I/O에 대한 트레이스 A를 추출했다. FTL 시뮬레이터는 트레이스 A를 입력받아 FTL을 통해서 가상 플래시 메모리로 요청한 읽기, 쓰기, 소거의 횟수를 바탕으로 소모된 전력을 계산하였다.

표 5는 FTL 시뮬레이션에서 발생한 읽기, 쓰기와 소거의 횟수를 기반으로 하여 λ와 μ의 평균으로 계산한 결과이다. 표 4의 λ와 μ를 토대로 플래시 전력기반 비용 모델을 적용한 결과는 그림 10과 같다.

플래시 메모리의 크기가 r, s 릴레이션을 합한 크기의 1.25배이기 때문에 덮어쓰기가 발생하여 λ와 μ에 차이를 보이게 된다. 특히 여유공간 기법은 여유공간을 먼저 읽는 특성 때문에 다른 FTL 알고리즘보다 λ의 값이 크다. 표 4의 λ와 μ를 토대로 FTL 시뮬레이터에서 수행한 결과는 그림 10과 같다. FTL 시뮬레이터에서의 결과는 그림 9의 비용 모델을 통한 결과와 유사하다. 하지만 색인된 중첩 반복 조인은 b_s가 M보다 훨씬 작아 B+-트리와 s 릴레이션이 모두 버퍼에 올라오기 때문에 계산된 값과 다른 결과를 보인다. 그림 9에서 여유공간 기법을 제외하고 블록 중첩 반복 조인이 합병 조인보다 좋은 성능을 보였지만 그림 10에서 보면 μ 값이 작아졌기 때문에 합병 조인이 더 좋은 성능을 보였다.

표 5. FTL 시뮬레이션 결과에서의 λ와 μ (CIL=1)

Table 5. λ and μ obtained by FTL simulations.

FTL	λ	μ
복사블록 기법	2.29	3.05
여유공간 기법	29.11	5.23
로그블록 기법	1.00	3.47

4. 플래시 임베디드 보드에서 결과

FTL 시뮬레이터에서의 결과를 검증하기 위해 플래시 임베디드 보드에서 실험을 하였다. 플래시 임베디드 보드에서의 실험은 FTL 시뮬레이터에서 가상 플래시 메모리에서 수행되는 읽기, 쓰기, 소거연산에 대한 트레이스 B를 추출하여 플래시 메모리에 수행하고 DAQ장비로 소요시간을 측정한다. 플래시 임베디드 보드의 트레이스 수행기(trace executor)는 트레이스 B를 플래시 메모리에 동작시키고, DAQ장비는 트레이스 B에 의해 플래시 메모리에서 발생한 읽기, 쓰기, 소거에 대한 전력을 측정한다. 이를 통해 플래시 임베디드 보드를 통해 얻은 결과는 그림 11과 같다.

그림 11에서 플래시 임베디드 보드에서의 결과는 그림 10에서 FTL 시뮬레이터를 통한 결과와 큰 차이점은 없다. 하지만 FTL 시뮬레이터를 통한 결과보다 전체적으로 적은 전력 소모를 보인다. 그 이유는 플래시 임베디드 보드에서 한 번의 연산을 수행한 전력 소모량과 여러 번 수행한 전력 소모량이 정확히 비례하지 않기

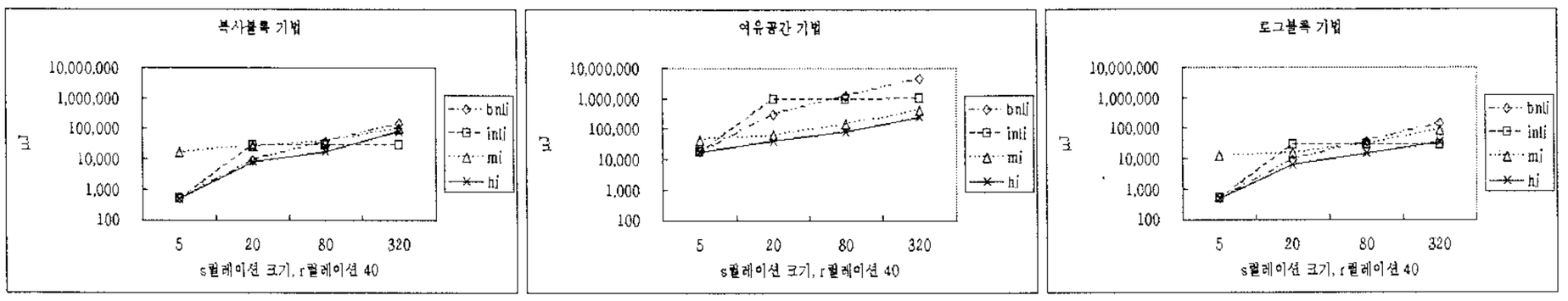

그림 10. FTL 시뮬레이터에서의 결과

Fig. 10. Experimental results on flash simulation.

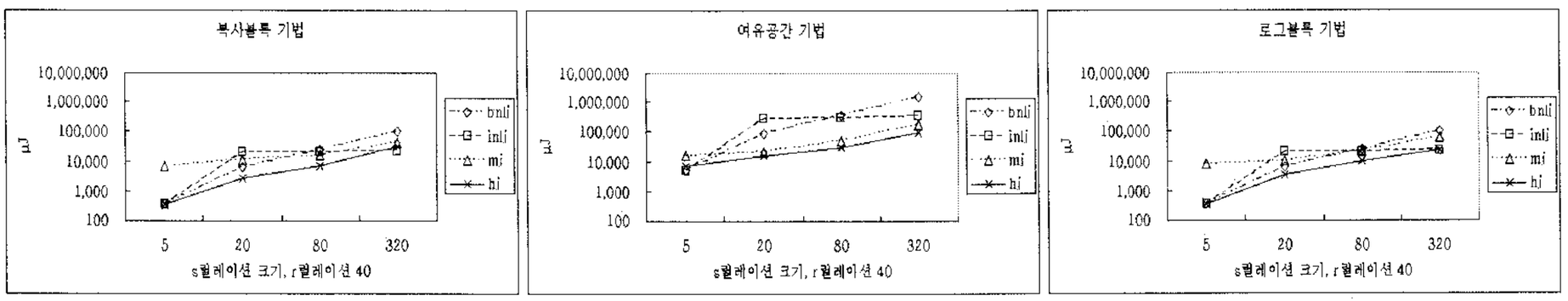

그림 11. 플래시 임베디드 보드에서의 결과

Fig. 11. Experimental results on flash embedded board.

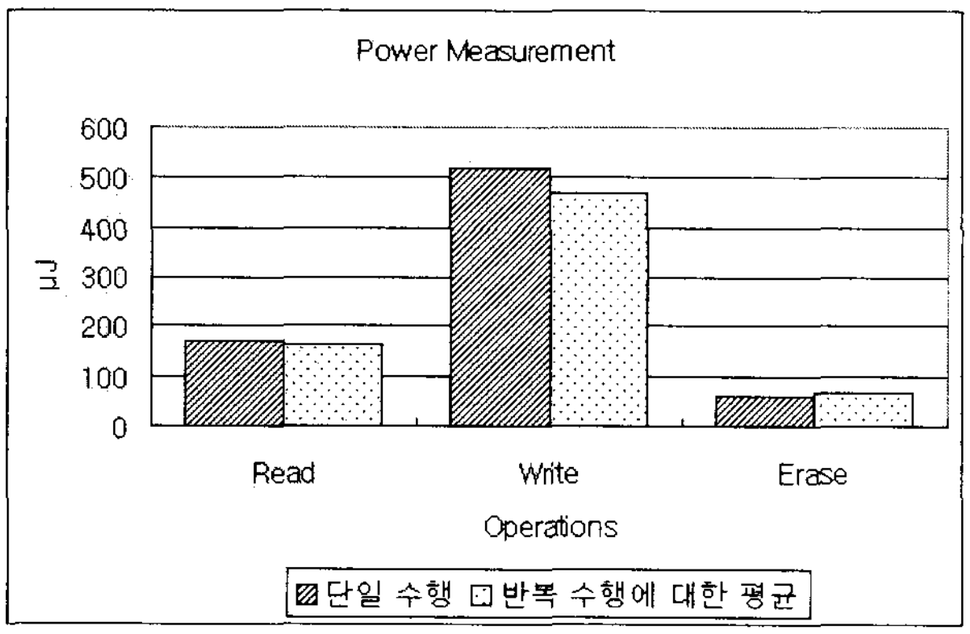

그림 12. 단일 수행과 반복 수행에 대한 평균값 비교
Fig. 12. Comparison average costs between unit operation and iterative operations.

때문이다.

그림 12는 플래시 임베디드 보드에서 한 번 소거 유닛에 대한 읽기, 쓰기, 소거 연산에 대한 전력과 10번 수행한 전력을 더한 것의 평균을 비교한 결과이다. 읽기와 쓰기의 경우 한 번 수행했을 때의 전력보다 10번 수행하여 평균을 낸 전력이 줄어들었으며 소거의 경우 늘어나는 결과를 보인다. 이것은 버스 인터리빙으로 인하여 플래시 콘트롤러에 명령어를 전달하는 것과 버퍼에서 플래시로 전달하는 복사 연산이 동시에 수행되기 때문이다. 이것은 플래시 콘트롤러의 명령어 전송 시간만큼의 이득이 발생한다. 따라서 FTL 시뮬레이터에서의 결과와 플래시 임베디드 보드에서의 결과에서 차이를 보이게 된다. 전체적인 조인의 수행결과는 비용모델의 예측된 결과와 FTL 시뮬레이터의 결과, 플래시 임베디드 보드에서의 결과가 모두 유사하다. 따라서 플래시 전력기반 비용 모델을 사용하면 플래시 메모리의 특성과 FTL의 특성을 모두 고려하여 질의를 최적화할 수 있는 방법이 된다.

Ⅶ. 결론 및 향후과제

최근 이동성이 중요한 요소로 차지하는 기기들이 등장하면서 플래시 메모리가 각광을 받고 있다. 보조기억장치로 디스크보다 많은 장점을 가지고 있다. 따라서 DBMS도 플래시 메모리를 저장 장치로 사용할 것이라고 예상된다. 하지만 휴대용 기기들은 제한된 자원을 가지며 전력에 의존적이다. 따라서 디스크와 다른 특성을 가진 플래시 메모리에서의 전력기반 비용 모델이 요하며 이를 고려한 질의 최적화가 필요하다.

플래시 메모리에서 전력기반 비용 모델은 플래시 메모리의 읽기, 쓰기, 소거의 전력 소모량을 기반으로 비용을 계산한다. FTL과 같은 플래시 메모리를 위한 시스템 소프트웨어에서는 I/O에 대한 추가 비용비인 λ와 μ를 고려하여 비용을 예상해야 한다. 본 논문에서는 TPC-A 벤치마크를 MySQL에서 수행하여 각 FTL에 따른 λ와 μ를 구하였으며, 이를 플래시 메모리에서 전력기반 비용 모델에 적용하여 릴레이션의 크기 변화에 따른 각 조인의 비용을 예측하였다.

질의 처리 시뮬레이션 시스템을 이용하여 전력 기반 비용 모델을 검증하고 평가하는데 사용하였다. DBMS의 조인 질의가 수행되는 것과 유사한 질의처리 시뮬레이션을 구현하여 I/O에 대한 트레이스 A를 추출하였다. 트레이스 A를 FTL 시뮬레이터를 이용하여 각 FTL 알고리즘에 따른 λ와 μ를 구하고 이를 기반으로 전력 소모량을 계산하여 본 논문에서 제시한 전력 기반 비용 모델을 통한 예측 결과와 비교, 분석하였다. 또한 FTL 시뮬레이터를 이용하여 얻은 트레이스 B를 플래시 임베디드 보드에서 수행하여 검증하였다. 실험을 통하여 전력 기반 비용 모델은 질의를 최적화하는데 유용하며 FTL 알고리즘에 따른 λ와 μ의 값이 중요한 요소로 작용할 것이며 λ와 μ를 잘 결정해야할 것이다.

앞으로 본 논문에서 다른 조인 질의뿐 아니라 다른 질의에 대해서도 비용 모델을 세우고 질의 최적화를 할 필요가 있다. 또한 각 FTL 알고리즘에 따른 λ와 μ를 제공받지 못하면 DBMS는 λ와 μ를 계산하여 자신의 플래시 메모리에 맞는 비용을 예상해야 한다. 따라서 이 λ와 μ는 점진적으로 수정해 나아가며 적당한 값으로 맞추어 나가는 방법에 대해서도 연구할 예정이다.

참 고 문 헌

[1] Prajakta Kalekar, Query Optimization in Resource Constrained Environment, M. Tech Project First Stage Report, Submitted in partial fulfillment of the requirements for the degree of Master of Technology

[2] 박원주, 박성환, 박상원, 윈도우즈 기반 플래시 메모리의 플래시 변환 계층 알고리즘 성능 분석, 한국정보과학회, 정보과학회논문지 : 컴퓨팅의 실제, Vol. 13, 2007. 11

[3] 김도윤, 박상원, KM-평준화: NAND 플래시 메모리를 위한 레벨 기반 소거 횟수 평준화 기법, 한국정보과학회, 한국정보과학회 학술발표논문집 한국정보과학회 2007 한국컴퓨터종합학술대회 논문집 (B), Vol. 34, 2007. 6, pp. 321~326

[4] E. Gal and S. Toledo. "Algorithms and data structures for flash memories.", ACM Computing Surveys, 37(2), 2005.
[5] Bum-soo Kim, Gui-young Lee, Method of driving remapping in flash memory and flash memory architecture suitable therefor, 2002. US Patent, no. 6,381,176 B1.
[6] Amir Ban. Flash file system optimized for page-mode flash technologies, 1999. US Patent, no. 5,937,425.
[7] Takayuki Shinohara. Flash memory card with block memory address arrangement, 1999. US Patent, no. 5,905,993.
[8] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for compact flash systems. IEEE Transactions on Consumer Electronics, 48(2), 2002.
[9] 박성환, 장주연, 서영주, 박원주, 박상원, 플래시 변환 계층에 대한 TPC-C 벤치마크를 통한 성능분석, 한국정보과학회, 한국정보과학회 학술발표논문집 한국정보과학회 2007 한국컴퓨터종합학술대회 논문집(A), Vol. 34, 2007. 6.
[10] MySQL AB. www.MYSQLkorea.co.kr, 1995~2007.
[11] Silberschartz, Korth, Sudarshan, Database System Concepts, McGraw-Hill, 2000.
[12] Goetz Graefe, Query Evaluation Techniques for Large Databases, ACM Computing Serveys, Volume 25, Number 2, 1993. 6.
[13] 김장수, 김태환, 윤재철, Database 시스템의 성능평가를 위한 TPC-A Benchmark의 구현, 한국정보과학회 가을 학술발표논문집 Vol.17 No.2, 1990,
[14] Transaction Processing Performance Council (TPC), TPC BenchmarkTM A - Standard Specification - Revision 2.0, 7 June 1994.

저 자 소 개

김 도 윤(정회원)
2006년 한국외국어대학교 정보통신공학과 학사.
2008년 한국외국어대학교 정보통신공학과 석사.
2008년~현재 대신증권.
<주관심분야 : 데이터베이스, 플래시메모리, 임베디드시스템

박 상 원(종신회원)
1994년 서울대학교 컴퓨터공학과 학사.
1997년 서울대학교 컴퓨터공학과 석사.
2002년 서울대학교 컴퓨터공학과 박사.
2002년~2003년 세종사이버대학교 디지털콘텐츠학과 전임강사.
2003년~현재 한국외국어대학교 정보통신공학과 부교수.
<주관심분야 : 데이터베이스, 플래시메모리, XML>