플래시 변환 계층에 대한 TPC-C 벤치마크를 통한 성능분석

(Performance Analysis of Flash Translation Layer using TPC-C Benchmark)

박 성 환 † 장 주 연 †
(Sunghwan Park) (Juyeon Jang)
서 영 주 †† 박 원 주 †††
(Youngju Suh) (Wonjoo Park)
박 상 원 ††††
(Sangwon Park)

요 약 최근 플래시 메모리는 소형 정보기기의 주된 저장매체로서 그 사용이 급격히 증가하고 있다. 또한, 플래시 메모리의 용량이 점점 증가함에 따라 더욱 많은 정보를 저장하고 관리하려는 시도가 이루어지고 있다. 따라서 효율적으로 정보를 관리하는 시스템인 데이타베이스가 플래시 메모리에서도 필요로 하게 되었다. 그러나 데이타베이스는 임의적인 디스크 I/O를 발생시키는 특징이 있어 현재의 플래시 메모리 시스템에서 그 성능이 매우 좋지 않다. 본 논문에서는 이러한 문제점을 고찰하고자 기존의 FTL 알고리즘들이 플래시 기반의 데이타베이스 시스템에서 어떠한 성능을 보이는지 실험하였다. 그 결과 실험에 사용한 FTL 알고리즘 모두가 매우 좋지 않은 성능을 보였다. 특히 지금까지 상당히 좋은 FTL 알고리즘으로 평가되었던 것들이 데이타베이스 응용에서는 나쁜 성능을 보였다. 또한 현재 플래시 메모리의 성능을 개선하고자 사용되는 칩 인터리빙 기술 또한 모든 FTL 알고리즘에서 좋은 성능을 내지 못하였다. 본 논문에서는 실험 결과를 바탕으로 데이타베이스 시스템을 잘 지원하는 새로운 FTL 알고리즘이 필요한 이유와 그 방향을 제시한다.

키워드 : 플래시 메모리, 플래시 변환 계층, TPC-C 벤치마크, 데이타베이스

Abstract The flash memory is widely used as a main storage of embedded devices. It is adopted as a storage of database as growing the capacity of the flash memory. We run TPC-C benchmark on various FTL algorithms. But, the database shows poor performance on flash memory because the characteristic of I/O requests is full random. In this paper, we show the performance of all existing FTL algorithms is very poor. Especially, the FTL algorithm known as good at small mobile equipment shows worst performance. In addition, the chip-interleaving which is a technique to improve the performance of the flash memory doesn't work well. In this paper, we inform you the reason that we need a new FTL algorithm and the direction for the database in the future.

Key words : Flash Memory, Flash Translation Layer, TPC-C benchmark, Database

1. 서 론

최근 디지털 카메라, 휴대폰, PDA 등 이동성을 중요시하는 여러 제품이 등장하였다. 이러한 휴대장치들은 낮은 소비전력, 휴대성, 안정성과 비휘발성을 지닌 저장매체가 필요하게 되었다. 플래시 메모리는 이러한 특성들을 만족함으로써 소형 정보기기의 주된 저장매체로서 사용되고 있다. 특히 플래시 메모리의 용량이 대폭 늘어남에 따라 머지않아 디스크를 대체하는 대표적인 저장매체가 될 것이다.

플래시 메모리는 데이타베이스 시스템을 통하여 보다 정보를 효율적으로 관리해야 할 필요가 있다. 그러나 플래시 메모리는 아직 데이타베이스를 충분히 지원하지 못하고 있다. 그 이유는 플래시 메모리가 물리적인 특성이 있어서 몇 가지 제약이 있기 때문이다. 우선 플래시 메모리는 각 블록에 대한 소거 횟수가 제한되어 있다. 그리고 특정 블록에 쓰기 연산을 하려면 반드시 그 블록이 먼저 소거되어 있어야 한다. 이러한 플래시 메모리의 단점들을 보완하기 위해 FTL(flash translation layer) 소프트웨어가 플래시 메모리 시스템에 탑재되어 있다[1].

· 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업[2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발]과 건설교통부 첨단도시기술개발사업 - 지능형국토정보기술혁신사업과제의 연구비지원(07국토정보C05)에 의해 수행되었음.
· 이 논문은 2007 한국컴퓨터종합학술대회에서 '플래시 변환 계층에 대한 TPC-C 벤치마크를 통한 성능분석'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 한국외국어대학교 컴퓨터및정보통신공학과
shpark@dislab.hufs.ac.kr
jyjang@dislab.hufs.ac.kr
†† 학생회원 : 한국외국어대학교 정보통신공학과
yjsuh@dislab.hufs.ac.kr
††† 학생회원 : 한국외국어대학교 컴퓨터및정보통신공학과
wjpark@dislab.hufs.ac.kr
†††† 종신회원 : 한국외국어대학교 정보통신공학과 교수
swpark@hufs.ac.kr

논문접수 : 2007년 9월 28일
심사완료 : 2008년 1월 4일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제2호(2008.4)

대표적인 FTL 알고리즘[2]으로 M-System사의 FMAX, 미쓰비시사의 여유 영역 기법과 로그 블록 기법(log block scheme)이 있다.

본 논문은 TPC-C 벤치마크를 수행하였을 때의 로그[3]를 이용하여 이미 제안된 세 가지 FTL 알고리즘들이 플래시 메모리를 저장매체로 사용하는 데이타베이스 시스템에서 어떠한 성능을 보이는지 평가하였다.

실험 결과 대표적인 세 가지 FTL 알고리즘 모두 매우 좋지 않은 성능을 보였다. 이것은 현재의 FTL 소프트웨어가 데이타베이스를 효율적으로 지원하지 못한다는 것을 의미한다. 그러므로 데이타베이스를 잘 지원하는 새로운 FTL 소프트웨어가 제안되어야 한다. 또한, 현재의 데이타베이스 시스템은 플래시 메모리 환경에 적합하도록 별도의 내부적인 수정이 필요하다. 본 논문은 세 가지 대표적인 FTL 알고리즘에 대한 시뮬레이션 결과를 분석하고, 새로운 FTL 알고리즘이 필요한 이유와 그 방향을 제시한다.

2. 관련 연구

2.1 플래시 메모리

플래시 메모리의 여러 장점에도 불구하고 플래시 메모리를 효율적으로 활용하려면 몇 가지 단점들을 극복해야 한다. 플래시 메모리는 하드디스크와는 다르게 표 1에서처럼 읽기와 쓰기 연산의 수행 속도가 다르며 덮어쓰기 연산이 불가능하여 블록을 소거한 후 쓰기 연산을 수행해야 하고, 소거 연산은 블록 단위로만 수행할 수 있다. 이로 인하여 쓰기 연산 시 별도의 소거 연산을 동반하여 디스크보다 오랜 수행시간이 걸릴 수 있다. 또한, 플래시 메모리는 10만 번 이상 소거 연산을 수행할 때, 그 블록에 대한 데이타의 일관성을 보장하지 못하는 단점도 있다[4].

본 논문에서 사용한 대블록 플래시 메모리는 4개의 섹터가 하나의 페이지를 구성하고 있으며, 한 번에 한 개의 페이지를 읽고 쓸 수 있다. 그러므로 최대 4개의 섹터에 대하여 동시에 읽고 쓰는 연산을 수행할 수 있다. 대블록 플래시 메모리의 한 블록은 64개의 페이지를 가지고 있으며, 한 블록의 크기는 약 128K 바이트이다.

표 1 접근 속도 : 하드 디스크 vs NAND 플래시

매체	접근 시간		
	읽기	쓰기	소거
하드 디스크	12.7 ms (2 KB)	13.7 ms (2 KB)	N/A
NAND 플래시	80 μs (2 KB)	200 μs (2 KB)	1.5 ms (128 KB)

디스크: 시게이트 바라쿠다 7200.7 ST380011A
NAND 플래시: 삼성 K9WAG08U1A 16 Gbits SLC NAND

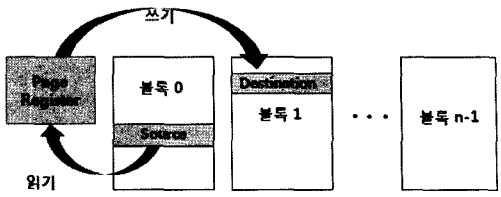

그림 1 내부 복사

대블록 플래시 메모리도 역시 블록 단위로 소거 연산을 수행한다.

최근의 플래시 메모리는 그 성능을 개선하기 위해 내부 복사(copy-back)와 칩 인터리빙(chip-interleaving) 기술을 사용한다. 내부 복사란 데이타의 복사를 칩 내부에서 해결하여 데이타의 전송 오버헤드를 줄이는 기술이다. 내부 복사 기술을 사용함으로써 복사 연산의 시간이 쓰기 연산의 시간과 같아진다. 플래시 메모리의 내부 복사 과정은 그림 1과 같다.

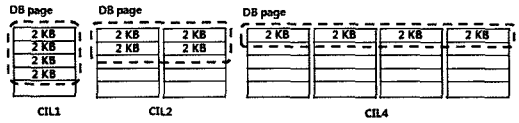

그림 2 칩 인터리빙

칩 인터리빙은 그림 2와 같이 여러 개의 플래시 칩을 병렬로 연결하여 읽기와 쓰기의 대역폭을 늘리는 기술이다. 예를 들어 칩 인터리빙의 값이 4인 경우에는 칩 4개로 인터리빙을 한 것으로서 8K 바이트 데이타를 한 번의 연산으로 읽기와 쓰기가 가능하다. 본 논문에서는 칩 인터리빙을 효율적으로 사용하기 위하여 가상 페이지와 가상 블록을 제안하였다. 가상 페이지(virtual page)는 플래시 메모리의 한 번에 읽고 쓸 수 있는 단위이며, 가상 블록(virtual block)은 플래시 메모리의 소거 단위이다. 예를 들어, 칩 인터리빙의 값이 4일 경우 가상 페이지는 4개의 페이지가 되며, 가상 블록은 4개의 블록이다. 칩 인터리빙을 적용한 플래시 메모리의 접근속도는 표 2와 같다. 칩 인터리빙 외에 버스 인터리빙을 사용할 수 있으나, 본 논문에서는 칩마다 하나의 버스를 연결한

표 2 NAND 플래시 메모리의 칩 인터리빙 시 접근 속도

	읽기	쓰기	소거
CIL(1)	320 μs (8 KB)	800 μs (8 KB)	1.5 ms (128 KB)
CIL(2)	160 μs (8 KB)	400 μs (8 KB)	1.5 ms (256 KB)
CIL(4)	80 μs (8 KB)	200 μs (8 KB)	1.5 ms (512 KB)

구조로 가정한다. 즉, 칩 인터리빙 시 칩별로 버스 인터리빙도 한다고 가정한다.

2.2 FTL

FTL 소프트웨어는 플래시 메모리를 블록 디바이스로 보여준다. FTL은 효율적인 매핑을 제공함으로써 연산 속도를 증진시키고 플래시 메모리의 수명을 늘려준다. 본 논문에서는 M-System사에서 발표한 FMAX, 미쓰비시사에서 발표한 여유 영역 기법과 로그 블록 기법을 사용하였다. 각 알고리즘의 특징은 표 3과 같다[5].

표 3 알고리즘의 특징 비교

	FMAX	여유 영역 기법	로그 블록 기법
데이타	고정 섹터 방식	고정 섹터 방식	고정 섹터 방식
매핑	블록 매핑	블록 매핑	혼합 매핑
덮어 쓰기	고정 섹터 방식 → 변동 섹터 방식	변동 섹터 방식	변동 섹터 방식
특징	각 블록당 복사블록을 사용	블록내의 여유 영역을 사용	한 블록에 로그 블록 하나를 할당

3. 실험 환경

본 논문에서는 [3]에서 사용한 로그를 이용하여 실험하였다. 사용한 로그는 TPC-C 벤치마크를 오라클(Oracle) 데이타베이스에 수행하여 생성한 재수행 로그(redo log)이다. 재수행 로그에는 하드디스크의 페이지 번호가 포함된 쓰기 연산 기록이 포함되어 있다. 본 논문에서는 실제로 플래시 메모리에 영향을 미치는 쓰기 연산들만을 고려하여 실험하였다. 표 4는 [3]에서 로그를 추출하기 위해 실험을 수행한 환경을 정리한 것이다. 데이타베이스를 설정할 때 플래시 메모리는 Raw 장비로 설정하고, 로깅 옵션을 주지 않았다. 이는 캐싱이나 로깅 시에 플래시 메모리의 I/O에 대한 간섭이 발생하는 것을 피하기 위해서이다. 그러므로 대부분의 I/O는 데이타 페이지나 인덱스 노드에 집중될 것이다. 플래시 메모리는 최근에 가장 많이 사용되는 대블록 플래시 메모리를 사용하였다.

본 논문에서는 [3]에서 사용한 로그를 FMAX, 여유 영역 기법과 로그 블록 기법에 적용하여 실험하였다.

각 알고리즘에 대해 플래시 메모리에서 발생한 총 읽기, 쓰기, 소거, 내부 복사 그리고 합병 연산의 횟수를 측정하였다. 각각의 연산 횟수를 통해 총 수행 시간, 데이타베이스 페이지 쓰기 연산에 대한 성능 비와 평균적인 응답 시간을 계산하였다. 이를 통해 각 알고리즘의 성능을 평가할 수 있다.

표 4 실험 환경 구성표

CPU/RAM	2.0 GHz Intel Pentium/1 Gbytes RAM
DB크기	1 Gbytes
DB버퍼크기	20 Mbytes
DB페이지크기	8 Kbytes
사용자 수	100 사용자

4. 성능 분석

본 절은 TPC-C 벤치마크의 I/O 트레이스(trace)를 이용하여 각 FTL에서의 성능을 시뮬레이션하고 비교, 분석한 경과에 관한 내용이다.

4.1 총 수행 시간

데이타베이스 I/O의 일정한 양을 수행하는데 걸린 총 수행 시간을 계산하였다. 이는 해당 알고리즘이 주어진 I/O를 얼마나 빠르게 처리할 수 있는지의 기준이 될 수 있다.

$$T_{total} = n_r C_r + n_w C_w + n_e C_e \quad (1)$$

식 (1)은 총 수행 시간을 계산하는 식이다. 이는 각 FTL 알고리즘이 데이타베이스의 I/O를 처리하는데 발생한 읽기, 쓰기와 소거 연산의 횟수에 각각의 비용을 곱해서 모두 더한 값이다. n_r와 n_w은 플래시 페이지 읽기와 쓰기 횟수, n_e은 플래시 블록 소거 횟수, C_r과 C_w은 플래시 페이지 읽기와 쓰기 비용, C_e은 플래시 블록 소거 비용을 나타낸다.

표 5 플래시 메모리에서의 총 수행 시간(초)

	CIL 1	CIL 2	CIL 4
FMAX	1480	822	483
여유 영역 기법	1899	1439	1560
로그 블록 기법	13856	14366	15928

표 5는 플래시 메모리의 크기가 2G바이트일 때 각 FTL 알고리즘에 대해서 칩 인터리빙에 따른 실험 결과이다. CIL은 칩 인터리빙을 뜻한다. 이 표에서 로그 블록 기법의 총 수행 시간이 가장 느린 것을 알 수 있다. 데이타베이스에서의 I/O 패턴은 임의적이다. 그러므로 로그 블록 기법은 로그 블록의 페이지를 많이 활용하지 못하고 계속해서 합병 연산이 발생하게 되어 수행 시간이 가장 느리다. FMAX와 여유 영역 기법은 복사블록이나 여유 영역에 변동 섹터 방식으로 데이타를 기록하기 때문에 복사블록이나 여유 영역의 활용도가 증대하고 블록의 활용도(utilization)가 증가하여 더 적은 합병 연산이 유발된다. 그러나 두 배의 메모리가 필요하기 때문에 성능이 좋다고 볼 수 없다. 또한 여유 영역과 복사블록의 전체 크기가 같음에도 불구하고 여유 영역 기법보다 FMAX의 성능이 가장 좋은 것을 알 수 있다. 이는 특정 페이지에 대한 쓰기 연산이 더 많이 발생하면 그 블록에 대한 여유 공간이 여유 영역 기법 방법보다는 더 크고 결국엔 합병 연산이 줄어들기 때문이다.

FMAX의 경우 칩 인터리빙이 작을 때보다 클 때가 총 수행시간이 줄어드는 것을 볼 수 있다. 하지만 로그

표 6 로그 블록 기법의 수행 결과

CIL	Read	Write	Erase	Copy-back	Log Util.
1	13,190,631	3,549,376	1,544,703	48,871,312	6.66%
2	25,240,632	1,774,688	1,525,665	48,519,404	4.27%
4	48,909,061	887,344	1,502,727	47,922,889	0.55%

블록 기법은 총 수행시간이 늘어난 것을 알 수 있다. 그 이유는 로그 블록 기법은 칩 인터리빙이 커짐에 따라 로그 블록의 활용도가 급격히 낮아지기 때문이다. 로그 블록 활용도란 할당된 총 로그 블록의 페이지 중에 사용된 페이지의 비율을 나타낸다.

표 6은 로그 블록 기법의 읽기, 쓰기, 소거 그리고 내부 복사의 연산 횟수와 로그 블록의 활용도를 나타낸 표이다. 로그 블록 기법은 칩 인터리빙이 커질수록 쓰기 연산의 횟수는 줄어들지만 읽기 연산의 횟수가 증가하고, 로그 블록의 활용도가 적어지는 것을 알 수 있다. 칩 인터리빙이 증가할 때마다 로그 블록 활용도가 떨어지는 것은 가상 블록의 크기는 커졌지만 데이타베이스의 I/O는 임의 쓰기 연산이라서 합병이 발생할 확률은 변화가 없기 때문이다.

표 7은 FMAX에 대해서 칩 인터리빙에 따른 실험 결과이고 칩 인터리빙이 증가함에 따라 총 수행 시간이 더 좋아진 것을 알 수 있다. 이는 칩 인터리빙이 클 때 읽기 연산의 수가 줄어들고 복사 블록의 크기가 커져 특정 블록에 쓰기 연산이 많은 경우 합병 연산이 줄고 이에 따라 내부 복사 연산의 수가 줄어들기 때문이다.

표 8은 여유 영역 기법에 대해서 칩 인터리빙에 따른 실험 결과이다. 표 5에서 본 것처럼 FMAX는 칩 인터리빙이 커짐에 따라 쓰기 연산의 수가 2배의 비율로 줄어들기 때문에 총 수행 시간은 점점 좋아진다. 하지만 표 8과 같이 여유 영역 기법에서는 칩 인터리빙이 2와 4인 경우를 비교해보면, 칩 인터리빙이 4일 때의 총 수행 시간이 더 느린 결과를 보이고 있다. 그 이유는 칩인터리빙이 커지면 탐색해야 하는 섹터의 개수가 증가하는데, 그 비율이 소거 연산과 내부 복사 연산의 감소비율보다 더 크기 때문이다. 즉, 칩 인터리빙이 커짐으로 인해 소거 연산의 횟수는 줄어들기 때문에 총 수행 시간이 줄어들어야 하지만 읽기 연산이 큰 폭으로 증가했기 때문에 총 수행 시간이 늘어난 결과를 보인다.

표 7 FMAX 총 수행 시간 결과

CIL	Read	Write	W_Spare	Erase	Copy-back	time(s)
1	1,801,106	3,549,376	49,680	102,090	2,316,480	1480
2	1,771,437	1,774,688	25,748	52,420	1,211,156	822
4	1,745,054	887,344	13,132	26,601	618,497	483

표 8 여유 영역 기법 총 수행 시간 결과

CIL	Read	Write	Erase	Copy-back	time(s)
1	4,779,344	3,549,376	96,712	3,309,036	1899.10
2	8,006,230	1,774,688	51,057	1,837,578	1439.54
4	14,403,668	887,344	26,232	959,039	1560.92

플래시 메모리의 크기가 2G바이트보다 큰 경우에도 총 수행 시간은 2G바이트인 경우의 총 수행 시간과 비슷했다. 이로서 플래시 메모리의 크기가 커져도 성능에는 영향을 미치지 않는 것을 알 수 있다.

4.2 데이타베이스 페이지 쓰기 연산에 대한 성능 비

하드디스크는 데이타를 고정 섹터 방식으로 기록하기 때문에 데이타베이스의 쓰기 연산 하나당 실제로 디스크에 쓰기 연산 하나가 발생한다. 하지만 플래시 메모리에서는 데이타베이스의 쓰기 연산 하나당 읽기, 쓰기와 소거 연산이 발생할 수 있다. 그러므로 쓰기 연산이 하나 발생했을 때 플래시 메모리에서 얼마나 부가적인 연산이 발생하는지가 중요하다. 본 논문에서는 데이타 페이지 하나를 저장하는데 필요한 비용에 대한 실제 비용의 비(α)를 계산하여 각 FTL 알고리즘의 성능을 평가하였다.

$$k = \frac{S_p}{0.5m} \tag{2}$$

데이타베이스의 한 페이지를 기록할 때 실제 플래시 메모리에 요청되는 페이지 개수 k는 식 (2)와 같다. S_p는 데이타베이스의 한 페이지 크기(K 바이트), 0.5는 플래시 메모리의 한 섹터의 크기(K 바이트), m은 플래시 메모리가 한 번에 읽기와 쓰기 연산을 하는 섹터의 수를 나타낸다. 예를 들어 데이타베이스에서 한 페이지의 크기가 8K 바이트이면 S_p는 8이다. 대블록 플래시 메모리에서는 4개의 섹터를 동시에 기록이 가능하므로 칩 인터리빙이 1이라면 m값은 4가 된다. 그러므로 k값은 4가 된다. 이것은 데이타베이스 한 페이지를 플래시 메모리에 저장하려면 4K바이트의 쓰기 연산이 소요됨을 말한다.

$$p_w = kC_w \tag{3}$$

$$P_w = \sum_{i=1}^{n} p_{w_i} = k \cdot n_d \cdot C_w \tag{4}$$

$$P'_w = n_r C_r + n_w C_w + n_e C_e \tag{5}$$

식 (3)은 데이타베이스의 한 페이지를 기록할 때 플래시 메모리에서 소요되는 시간(p_w)을 나타낸다. 예를 들어, k가 4일 때 p_w는 0.8ms가 된다. 데이타베이스에서 n_d번의 쓰기 연산이 일어날 때 가장 최적의 시간 p_w는 식 (4)와 같다. 실제 플래시 메모리에 페이지 쓰기 연산을 수행하는데 소요된 시간 P'_w는 식 (5)와 같다.

그러므로 식 (6)에서와 같이 최적 비용에 대한 실제 비용의 비(a)를 구하여 플래시 메모리에 대한 성능을 평가할 수 있다. 즉, a는 데이타베이스의 페이지 쓰기 연

$$\alpha = \frac{P'_w}{P_w} = \frac{n_r C_r + n_w C_w + n_e C_e}{n_d \cdot p_w}$$
$$= \left(\frac{n_r}{n_d} C_r + \frac{n_w}{n_d} C_w + \frac{n_e}{n_d} C_e\right) \Big/ p_w \quad (6)$$

산을 처리하는데 있어서 실제로 발생한 비용을 예상 비용으로 나누어 그 비율을 나타낸 것이다. 이것은 칩 인터리빙에 따라 변하는 값이므로 하드웨어의 특성을 고려한 것으로 한 개의 페이지를 기록할 때 드는 총 비용과 최소 비용의 비율이다.

표 9 플래시 메모리에서의 α

	CIL 1	CIL 2	CIL 4
FMAX	2.09	2.32	2.72
여유 영역 기법	2.68	4.06	8.80
로그 블록 기법	19.52	40.48	89.76

표 9는 칩 인터리빙에 따른 각 FTL의 α를 나타낸 것이다. 칩 인터리빙이 1일 때 FMAX의 α는 2.09임을 알 수 있다. 이것은 FMAX 알고리즘을 사용할 때 데이타베이스의 한 페이지를 기록하는데 플래시 메모리에서 소요되는 시간이 실제 플래시 메모리에 페이지를 기록하는 시간의 2.09배가 걸림을 나타내는 것이다. 각 알고리즘에서 칩 인터리빙이 커질수록 α값이 증가한다. 이것은 한 번에 읽거나 쓸 수 있는 섹터의 개수가 증가하여 성능을 증진시킬 수 있으나, 로그 블록, 복사블록과 여유 영역 등의 활용도가 낮고, 가상 블록이 커질수록 페이지를 찾는 데 걸리는 시간이 길어지기 때문이다. FMAX의 칩 인터리빙에 따라 α값이 천천히 증가하는 데 비해 여유 영역 기법이나 로그 블록 기법의 α값은 약 2배의 비율로 증가함을 알 수 있는데 이는 여유 영역 기법나 로그 블록 기법에서 가상 블록이 커짐에 따라 각 쓰기 연산은 줄어들지만 읽기 연산의 수가 급격히 늘어나고 블록의 활용도가 낮아지기 때문이다. 반면 FMAX의 읽기 연산의 수는 표 7에서 보는 것처럼 오히려 줄어드는 것을 알 수 있다.

4.3 평균적인 응답 시간

플래시 메모리는 데이타베이스에서 I/O를 요청하게 되면 빠른 시간 내에 결과를 반환해야 한다. 본 논문에서는 데이타베이스의 한 I/O를 처리하는데 걸리는 평균적인 시간을 계산하였다. 데이타베이스의 한 I/O를 처리하는데 걸리는 평균 시간이 작을수록 효율적으로 처리한다고 할 수 있다. 표 9에서 보면 FMAX와 여유 영역 기법의 평균 응답 시간은 칩 인터리빙이 커질수록 평균 응답 시간이 줄어드는 반면, 로그 블록 기법의 평균 응답 시간은 커짐을 알 수 있다. 그 이유는 4.2절에서 설명한 것과 같이 탐색해야 하는 섹터의 개수가 증가하기 때문이다.

표 10 플래시 메모리에서의 평균 응답 시간

	CIL 1	CIL 2	CIL 4
FMAX	1.67	0.93	0.55
여유 영역 기법	2.14	1.62	1.76
로그 블록 기법	15.62	16.19	17.95

5. 결 론

플래시 메모리의 FTL 알고리즘들은 순차적이거나 서너 개의 블록을 동시에 참조할 때는 좋은 성능을 발휘한다. 이러한 패턴을 보이는 윈도우즈 파일 시스템에서 몇 개의 FTL 알고리즘은 좋은 성능을 보였다[5]. 하지만 데이타베이스와 같은 임의의 위치에 쓰기 연산을 수행하는 경우에는 나쁜 성능을 보인다. 본 논문에서는 TPC-C 벤치마크를 수행하여 I/O 패턴을 얻어 이를 각각의 FTL 알고리즘에 적용시켜 그 성능을 평가했다.

대표적인 FTL 알고리즘 중에 디지털 카메라, 휴대폰, PDA 등에서 성능이 매우 좋은 로그 블록 기법은 데이타베이스 시스템에서 가장 좋지 않은 성능을 보였다. 또한 이러한 알고리즘들은 칩 인터리빙과 같은 하드웨어의 기능을 잘 이용하지 못하는 경향을 보였다. 이는 데이타베이스 시스템에 맞는 새로운 저장 기법이 필요함을 의미한다. 새로운 저장 기법은 하드웨어 특성을 잘 활용하고, 플래시 메모리의 단점을 효과적으로 극복할 수 있는 기법이어야만 한다. 데이타베이스 시스템에 맞는 새로운 저장 기법이 개발되어야만 휴대용 저장 장치로서 플래시 메모리가 진정한 가치를 발휘할 것이다.

참 고 문 헌

[1] Eran Gal and Sivan Toledo. Algorithms and Data Structures for Flash Memories. ACM Computing Surveys. Volume 37, Issue 2. June 2005. Pages: 138-163 : 2005.

[2] 박원주, 박성환, 박상원, 윈도우즈 기반 플래시 메모리의 플래시 변환 계층 알고리즘 성능 분석, 한국정보과학회, 정보과학회논문지 : 컴퓨팅의 실제, Vol.13, 2007. 11, pp. 213-225.

[3] Sang-Won Lee, Bongki Moon. Design of Flash-Based DBMS: An In-Page Logging Approach. ACM SIGMOD International Conference on Management of Data, Beijing, China, June, 2007.

[4] Tae-Sun Chung, Dong-Joo Park, Sangwon Park, Dong-Ho Lee, Sang-Won Lee, and Ha-Joo Song, System Software for Flash Memory: A Survey, EUC 2006, Aug. 2006.

[5] 박원주, 유현석, 박성환, 김도윤, 박상원. 윈도우즈 파일 시스템에서 플래시 메모리의 FTL 알고리즘 성능 분석. 한국정보과학회 학술대회, 2005. 7.