DOI: 10.3745/KIPSTA.2008.15-A.6.335

하이브리드 플래시 메모리를 위한 적응적 가비지 컬렉션 기법

임 수 준† · 신 동 군††

요 약

본 논문에서는 SLC와 MLC를 모두 가진 하이브리드 플래시 메모리를 효율적으로 사용하기 위한 적응적 가비지 컬렉션 기법을 제안한다. 하이브리드 플래시 메모리는 속도가 빠른 SLC 영역과 용량대비가격이 저렴한 MLC 영역으로 이루어져 있기 때문에 SLC 영역을 로그 버퍼로, MLC 영역을 데이터 블록으로 사용하는 것이 효율적이다. 제안하는 가비지 컬렉션 기법은 MLC 영역의 쓰기 속도가 매우 느리다는 점을 고려하여 SLC 로그버퍼에 기록된 데이터 중에서 더 이상 갱신되지 않으며 MLC 영역으로의 이동 비용이 적은 데이터를 MLC 영역의 데이터 블록으로 이동시키고, 자주 갱신될 데이터는 SLC 내부에서 이동시킴으로써 SLC 영역의 빈 공간을 확보한다. 또한 적응적 기법을 사용하여 입출력의 패턴을 관찰하여 가비지 컬렉션의 기준 값을 변화시킨다. 실험 결과, 본 논문에서 제안한 기법은 기존에 소개된 플래시 메모리 관리 기법에 비하여 하이브리드 플래시 메모리의 특징을 효율적으로 사용하여 성능을 향상시켰으며 워크로드에 따라서 최적에 가까운 가비지 컬렉션 기준 값을 찾아내는 것을 확인할 수 있었다.

키워드 : 플래시 메모리, 플래시 변환 계층, 하이브리드 스토리지, 가비지 컬렉션, 임베디드 시스템

Adaptive Garbage Collection Technique for Hybrid Flash Memory

Soojun Im† · Dongkun Shin††

ABSTRACT

We propose an adaptive garbage collection technique for hybrid flash memory which has both SLC and MLC. Since SLC area is fast and MLC area has low cost, the proposed scheme utilizes the SLC area as log buffer and the MLC area as data block. Considering the high write cost of MLC flash, the garbage collection for the SLC log buffer moves a page into the MLC data block only when the page is cold or the page migration invokes a small cost. The other pages are moved within the SLC log buffer. Also it adjusts the parameter values which determine the operation of garbage collection adaptively considering I/O pattern. From the experiments, we can know that the proposed scheme provides better performance compared with the previous flash management schemes for the hybrid flash and finds the parameter values of garbage collection close to the optimal values.

Keywords : Flash Memory, Flash Translation Layer, Hybrid Storage, Garbage Collection, Embedded System

1. 서 론

플래시 메모리는 하드디스크에 비하여 높은 안정성과 전력 소모가 적다는 장점으로 인하여 휴대용 가전 기기의 저장장치로 사용되고 있다. MP3 플레이어나 휴대전화, 디지털 카메라 등의 휴대용 기기가 널리 보급됨에 따라 플래시 메모리의 수요도 급속히 증가하는 추세이다.

플래시 메모리는 기존의 저장장치와 다른 몇 가지 특징들을 가지고 있다. 플래시 메모리는 읽기와 쓰기, 삭제의 세 가지 연산을 지원한다. 읽기와 쓰기는 페이지 단위로, 삭제는 블록 단위로 수행된다. 삭제 연산은 블록 전체의 비트 셀의 값을 '1'로 초기화시킨다. 쓰기 연산은 페이지의 일부 비트 셀의 값을 '0'으로 바꾸어 준다. 그리고 플래시 메모리는 '삭제 후 쓰기 가능' 하다. 즉, 데이터를 기록한 블록을 수정하는 것은 불가능하며 값을 변경하기 위해서는 삭제 연산을 수행하여 블록을 지운 다음 쓰기 연산을 사용해야 한다. 또한 삭제의 단위가 블록임에 비하여 쓰기의 단위는 페이지이며 한 블록 내에서 페이지는 순차적으로 쓰여야 하는

※ 이 논문은 2007년도 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2007-331-D00358).
† 정 회 원 : 성균관대학교 전기전자컴퓨터공학과 석사과정
†† 정 회 원 : 성균관대학교 정보통신공학부 전임강사
논문접수 : 2008년 9월 23일
심사완료 : 2008년 10월 13일

특징이 있다. 즉, 블록 내부의 첫 번째 페이지부터 마지막 페이지까지 순서대로 기록할 수 있으며 현재 쓰인 페이지 이전의 내용을 다시 쓰는 것은 불가능하다.

이와 같은 플래시 메모리의 특성 때문에 플래시 메모리를 사용하는 시스템에서는 플래시 변환 계층(FTL: Flash Translation Layer)을 이용하여 상위 계층의 논리 주소를 실제 플래시 메모리의 물리 주소로 변환하여 데이터에 접근한다. 이러한 주소 변환 기법에는 블록 수준 변환, 페이지 수준 변환, 그리고 하이브리드(hybrid) 변환 기법이 있다.

블록 수준 변환은 논리 주소를 물리 주소로 변환하기 위한 테이블을 블록 단위로 관리하며, 블록 내부에서의 페이지 위치는 논리 주소와 물리 주소가 동일하게 기록된다. 그러므로 페이지를 찾을 때 블록 단위로 주소를 변환하여 페이지에 접근하며 변환 테이블에 요구되는 저장 공간이 작다는 장점이 있다. 그러나 갱신이 불가능한 플래시 메모리의 특징 때문에 한 페이지의 갱신 요청에도 새로운 블록을 할당받아 블록 내의 다른 페이지들을 모두 복사해야하므로 쓰기 성능이 매우 나빠진다는 것이 단점이다.

페이지 수준 변환 방식은 논리 주소와 물리 주소 사이의 변환을 페이지 단위로 관리한다. 그러므로 블록의 임의의 위치에 페이지를 저장할 수 있으며, 데이터를 갱신하는 경우 이전 데이터를 무효화(Invalidate)시키고 새로운 페이지만을 기록하기 때문에 연산 속도가 비교적 빠르다. 그러나 모든 페이지의 변환 주소를 테이블로 관리하기 때문에 이를 위해 많은 저장 공간이 필요하다.

하이브리드 변환 방식은 페이지 수준 변환과 블록 수준 변환의 두 기법을 동시에 사용한다. 하이브리드 변환 방식에서 플래시 전체의 블록은 로그 블록과 데이터 블록으로 구분된다. 로그 블록은 로그 버퍼로도 일컬어지며 쓰기 요청이 발생하면 로그 버퍼에 데이터를 기록하게 된다. 또한 로그 버퍼는 페이지 단위로 데이터를 관리하며 일반적인 블록 방식에 비하여 쓰기 연산에 소요되는 비용을 줄일 수 있다. 로그 버퍼를 모두 사용하여 빈 공간이 없는 경우 데이터 블록으로 데이터를 이동시키며 이 연산을 합병(Merging)이라고 한다. 합병된 데이터는 블록 단위로 관리되는 데이터 블록에 저장되며 블록 수준 변환 기법을 사용하여 관리한다. 합병 연산은 완전 합병(full merge), 부분 합병(partial merge), 교체 합병(switch merge)의 3가지 종류로 나누어진다 [1]. 부분 합병과 교체 합병은 로그 블록의 모든 페이지들이 논리주소에 의해 결정되는 블록 내의 해당 오프셋에 기록되었을 때 사용할 수 있는 방법으로 페이지 복사의 수가 적고 삭제 연산도 줄어들기 때문에 작은 비용으로 수행 가능하다. 반면 완전 합병은 많은 페이지를 복사해야 하며 여러 블록을 삭제하기 때문에 비용이 크다.

낸드 플래시 메모리는 싱글 레벨 셀(SLC:Single Level Cell)과 멀티 레벨 셀(MLC:Multi Level Cell)의 두 가지 종류로 나누어진다. SLC 플래시 메모리는 하나의 비트 셀(bit cell)에 1-bit의 정보를 저장할 수 있으며 MLC 플래시 메모리는 2-bit의 정보를 저장할 수 있다 [2]. SLC는 MLC에 비하여 읽기와 쓰기 연산의 속도가 빠르며 쓰기와 삭제 연산을 비교적 많이 수행할 수 있어 안정성이 높은 반면 MLC는 같은 크기의 비트 셀에 두 배의 용량을 기록할 수 있으므로 SLC에 비해서 많은 저장 공간을 확보할 수 있는 장점이 있다.

근래에 SLC와 MLC를 함께 쓰는 플래시 메모리 저장 장치가 개발되고 있다. 하이브리드 플래시 메모리는 앞서 살펴본 SLC와 MLC를 함께 사용하여 각각의 장점을 취하는 것을 목적으로 개발되었다. 먼저 SLC와 MLC 칩들을 병행하여 다수의 낸드 플래시로 저장 장치를 구성하는 방법이 있으며, 대용량 반도체 디스크(SSD:Solid State Disk)에서 이러한 방법을 사용하는 기법 [3]이 제안되었다. 반면에, 하나의 칩에 SLC와 MLC 블록을 함께 가지는 칩도 개발되었는 데, 예를 들어 삼성에서 개발한 Flex-OneNAND [4]와 도시바에서 개발한 mobileLBA-NAND [5]는 각각의 블록을 SLC나 MLC로 설정하여 사용 가능하다. 칩 내부의 블록들을 SLC 영역과 MLC 영역으로 구분할 수 있으며, 각 영역에 할당된 블록의 수에 따라 전체 플래시 칩의 용량이 결정된다. 예를 들어 SLC 블록 하나의 용량이 256KB이며 MLC 블록 하나의 용량이 512KB일 때 1024개의 블록을 가진 칩에서 SLC 영역에 256개의 블록을 할당하고 MLC 영역에 나머지 768개의 블록을 할당하면 전체 칩의 용량은 458MB(=65MB +393MB)로 구성된다.

<표 1> [4]은 하이브리드 플래시 메모리에서 SLC 블록과 MLC 블록의 특징을 나타낸다. 두 종류의 블록에서 한 페이지의 크기는 4KB로 동일하나 블록 내부의 페이지 개수가 SLC 블록은 64개인 반면에 MLC 블록은 128개로서 MLC 블록이 SLC 블록에 비하여 2배의 용량을 갖는다. 또한 읽기 속도는 큰 차이가 없지만 쓰기 속도는 MLC 블록이 SLC 블록에 비하여 매우 느리다는 특징이 있다. 또한 블록을 쓰고 지울 수 있는 횟수는 SLC가 MLC에 비하여 5배 정도 많다. 이러한 특징은 일반적인 SLC 칩이나 MLC 칩과 다소 차이가 있다. 예를 들어 일반적인 SLC 칩에서 한 페이지를 읽는 속도는 MLC 칩에 비하여 매우 빠르다.

본 논문에서는 하이브리드 플래시 메모리를 위한 새로운 FTL 기법을 제시한다. 이 FTL은 하이브리드 변환 방식을 기반으로 SLC 영역을 MLC 영역의 데이터 블록의 로그 버퍼로 사용하여 SLC 영역의 빠른 속도와 MLC 영역의 큰 용량을 활용하는 것을 목적으로 한다.

〈표 1〉 SLC와 MLC 플래시 블록의 특성

	SLC	MLC
페이지 크기	4KB	
블록 크기	256KB (64페이지)	512KB (128페이지)
읽기 시간(1페이지)	45us	50us
쓰기 시간(1페이지)	240us	1ms
삭제 시간(1블록)	500us	
삭제 가능한 횟수	50K	10K

논문의 내용은 다음과 같다. 2장에서는 이제까지 제시된 플래시 메모리 관리 기법들을 간략히 소개할 것이며 3장에서는 가비지 컬렉션 기법과 데이터 블록에 직접 쓰기 및 실행 중에 각 기준 값을 변경하는 알고리즘을 기술한다. 4장에서는 시뮬레이션을 사용하여 제시한 기법의 성능을 평가하고, 마지막으로 5장에서 결론과 향후 연구 과제에 대하여 언급하고자 한다.

2. 관련 연구

로그 버퍼 기반 FTL 기법은 로그 블록과 데이터 블록의 연관 정책에 따라 (그림 1)과 같이 1:1 로그 블록 변환(BAST) [1]과 1:N 로그 블록 변환(FAST) [6]으로 나누어진다. 이러한 기법은 얼마나 많은 로그 블록이 데이터 블록에 연관되는지에 따라 정의된다.

1:1 기법에서 하나의 로그 블록은 단 하나의 데이터 블록의 쓰기 연산을 위한 공간으로 사용된다. (그림 1)(a)은 5개의 데이터 블록 B0, B1, B2, B3, B4와 2개의 로그 블록 L0, L1을 갖는 BAST FTL을 나타내고 있으며 각각의 블록이 4개의 페이지로 이루어진다고 가정한다. 페이지 p0과 p1에 대한 갱신 요청이 있을 때 이 페이지들은 로그 블록에 기록되고 데이터 블록의 해당 페이지들은 무효화 된다. 로그 블록 L0과 L1은 각각 데이터 블록 B0과 B1에 대한 쓰기 요청을 위해 할당되었다. BAST와 같은 1:1 로그 블록 변환 방식은 로그 블록에 대한 합병 연산의 수가 많아질 수 있는 가능성이 있다. (그림 1)(a)에 추가로 p8, p12, p1, p5, p9, p13의 순서로 페이지 쓰기 연산이 요청되었다면 각 쓰기마다 매번 합병 연산이 발생한다. 또한 로그 블록들은 1개의 페이지만 사용한 상태에서 지워지므로 로그 블록을 비효율적으로 사용하는 결과를 낳는다. 즉 쓰기 패턴이 불규칙적일 경우에는 1:1 변환에서는 로그 블록의 잦은 교체로 인하여 성능이 나빠진다.

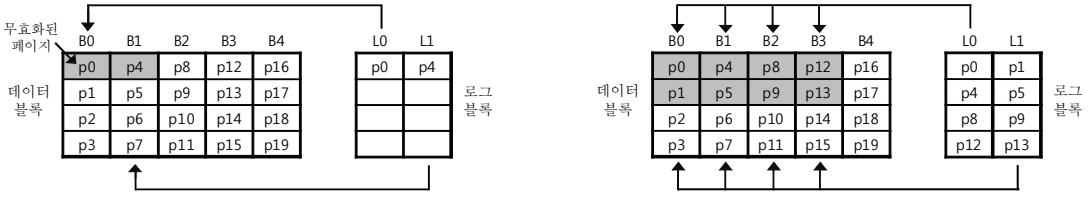
이러한 문제를 해결하기 위하여 1:N 변환 방식을 사용하는 FAST FTL 기법이 제안되었다. (그림 1)(b)와 같이 1:N 방식에서 로그 블록은 동시에 여러 데이터 블록에 대한 버퍼로 사용되기 때문에 불규칙적인 쓰기 패턴에 의해 발생하는 로그 블록의 빈번한 교체(log block thrashing) 문제가 해결되었다. 하지만 1:N 변환에서는 몇 개의 데이터 블록이 로그 블록과 관련이 있는가를 나타내는 로그 블록 연관도가 높아지게 된다. (그림 1)(b)의 로그 블록 L1에 대한 합병이 발생하면 16번의 페이지 복사가 발생하게 된다(16페이지 = 4블록 × 4페이지). 이는 로그 블록 L1은 4개의 데이터 블록 B0, B1, B2, B3을 위해 사용되기 때문이다. 그러므로 FAST 방식에서는 블록 합병의 횟수는 적으나 한 번 합병 연산을 수행하는데 드는 비용은 크다는 특징이 있다.

최근에는 지역성을 고려한 변환 방식인 LAST [7]가 소개 되었으며 이 기법에서는 로그 버퍼를 순차 로그 버퍼와 핫 랜덤 로그 버퍼, 콜드 랜덤 로그 버퍼로 구분하여 가비지 컬렉션 연산(Garbage Collection)의 비용을 줄이고 있다. 또한 Superblock [8]이나 SAST [9]와 같은 N:K 로그 변환 기법이 제안되었으며 이 기법에서는 N개의 데이터 블록에 대해서 K개의 로그 블록을 연관시키는 방법을 사용하고 있다. 또한 합병 비용을 고려하여 로그 블록을 데이터 블록과 합병시키지 않고 로그 블록 내부에서 데이터를 이동시키는 기법[10]도 제안되었다. 이 기법에서는 로그 블록에서 유효한 페이지의 수를 기준으로 합병 연산과 로그 블록 내부에서의 데이터 이동 중에서 더 효과적인 연산을 선택하여 수행하는 것으로 효율적인 가비지 컬렉션을 수행한다.

하지만 지금까지의 연구들은 SLC와 MLC 블록을 모두 가진 하이브리드 낸드 플래시의 특징을 고려하지 않고 있다. 본 연구에서는 하이브리드 낸드 플래시의 특징을 고려한 가비지 켈렉션 기법을 처음으로 제시하고 있다.

3. 하이브리드 플래시 메모리 관리 기법

하이브리드 플래시 메모리는 SLC 영역과 MLC 영역으로 나누어지며 접근이 빠른 SLC 영역에 응용 프로그램을 기록하고 용량이 큰 MLC 영역에 대용량의 데이터를 저장하는 용도로 개발되었다. 이러한 용법은 SLC와 MLC의 특징에 기인한다. 그러나 하이브리드 플래시 메모리는 MLC



(a) 1:1 로그 블록 변환(BAST)

(b) 1:N 로그 블록 변환(FAST)

(그림 1) 로그 버퍼 기반 FTL 기법

영역의 읽기 속도가 SLC 영역에의 읽기와 큰 차이를 보이지 않는 반면 쓰기 연산의 경우 MLC 영역이 SLC 영역보다 훨씬 느리다는 특징이 있다. 때문에 읽기 연산보다 쓰기 연산의 속도가 전체 성능에 큰 영향을 미친다. 이러한 점을 감안하여 각 영역에 기록할 데이터를 구분할 필요 없이 SLC 영역을 자주 갱신되는 데이터의 쓰기 명령을 수행하는 공간으로 활용하는 것으로 플래시 메모리를 효율적으로 사용할 수 있다.

하이브리드 플래시 메모리를 위한 FTL은 [11]에서 처음으로 제시되었다. 이 기법에서는 SLC 블록이 로그 버퍼가 아닌 핫 데이터(hot data)를 저장하기 위한 공간으로 할당되며 MLC 블록을 콜드 데이터(cold data)를 저장하기 위한 공간으로 사용한다. 그러나, 해당 기법에서는 페이지 수준 변환을 사용하기 때문에 변환 테이블이 크며 SLC 영역이 비교적 많이 요구되는 단점이 있다.

본 논문에서는 SLC 영역을 로그 버퍼로 사용하며 크기가 큰 MLC 영역을 데이터 블록으로 할당한다. 그러므로, 파일 시스템에서 쓰기 명령이 요청되면 FTL은 우선 SLC 영역의 로그 버퍼에 데이터를 기록한다. SLC 영역의 로그 버퍼는 페이지 수준 변환을 사용하며 MLC 영역의 데이터 블록은 블록 수준 변환을 사용하는 하이브리드 변환 기법을 사용하며 1:N 로그 블록 변환 방식을 채택하고 있다. 만약 쓰기 요청에 대하여 로그 버퍼의 빈 공간이 더 이상 없어 이를 처리할 수 없는 경우에는 가비지 컬렉션을 수행한다. 가비지 컬렉션 연산은 로그 버퍼 내부의 무효화된 페이지를 지우며 더 이상 접근이 없는 콜드페이지(cold page)를 MLC 영역의 데이터 블록으로 이동시켜 로그 버퍼에 빈 공간을 확보한다.

3.1 가비지 컬렉션 기법

기존의 로그 버퍼 기반 FTL에서 합병 연산은 로그 블록과 데이터 블록의 모든 유효한 페이지를 복사하여 연산을 수행한다. 그러나 하이브리드 플래시 메모리는 SLC와 MLC의 쓰기 속도의 차이가 매우 크기 때문에 합병 연산의 비용을 고려해야 한다. MLC 영역의 데이터 블록으로 이동시키는 데이터는 더 이상 쓰기 연산이 발생하지 않을 수록 유리하다. 그러므로 가비지 컬렉션을 수행할 때 자주 접근이 되지 않은 콜드 페이지만 SLC 영역에서 MLC 영역으로 복사해야 한다.

가비지 컬렉션 시 페이지를 이동시키는데 드는 비용도 고려할 필요가 있다. 데이터 블록 내에서 극소수의 페이지만 갱신되어 SLC 로그 버퍼에 기록되어 있는 경우에는 해당 블록의 소수의 페이지를 병합하기 위해서 많은 수의 페이지를 MLC 영역에서 복사해야 한다. 이 경우 해당 페이지가 속한 블록을 병합하는 것보다 그 블록에 추가 업데이트가 발생하는 것을 기다리기 위해서 병합을 최대한 미루는 것이 유리하다. 그러므로 가비지 컬렉션 연산 시에 논리 블록에 할당된 MLC 영역의 데이터 블록 내부에 유효한 페이지의 수에 따라 병합 여부를 결정해야 한다.

가비지 컬렉션은 데이터 블록 구분, 데이터 블록으로의 페이지 이동, 로그 버퍼 내부에서의 페이지 이동으로 구성된다.

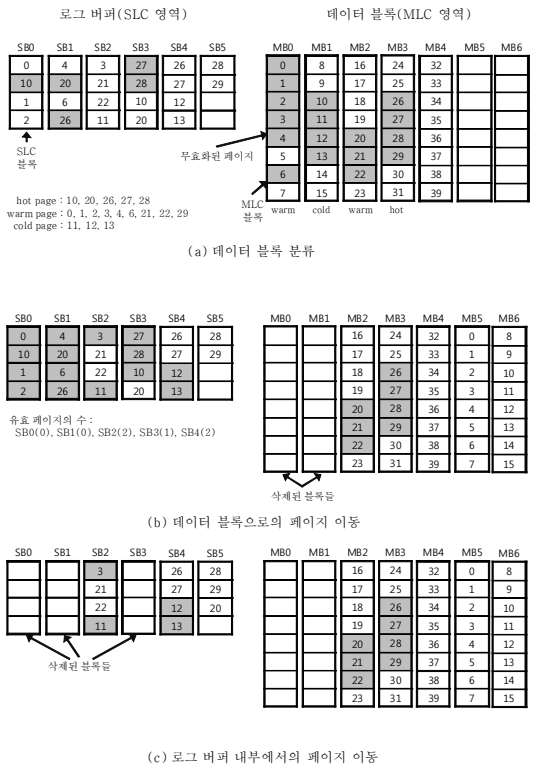
3.1.1 데이터 블록 구분

가비지 컬렉션에서 데이터를 이동시킬지 여부를 결정하기 위하여 각 페이지들을 분류한다. SLC 로그 버퍼의 페이지들은 데이터가 최근 접근된 빈도에 따라 핫 페이지, 콜드 페이지, 웜 페이지(warm page)의 세 종류로 구분된다. 핫 페이지는 접근이 자주 일어나는 페이지를 말한다. 가장 최근의 가비지 컬렉션 이후에 해당 논리 페이지에 쓰기가 발생한 횟수가 기준 값 P_{hot}을 초과하면 그 페이지를 핫 페이지로 간주한다. 또한 P_{cold} 번 이상 가비지 컬렉션이 발생했음에도 쓰기 연산 및 병합에 의한 페이지 이동이 없었던 페이지를 더 이상 접근이 없을 것으로 간주하고 콜드 페이지로 정의한다. 그 외의 페이지는 모두 웜 페이지이다. 예를 들어 (그림 2)에서 핫 페이지는 10, 20, 26, 27, 28 페이지이며 11, 12, 13의 경우 콜드 페이지로 분류했다.

SLC 로그 버퍼에 연관된 MLC 데이터 블록 역시 세 종류로 분류한다. 본 기법에서는 각 데이터 블록이 로그 블록에 가지고 있는 페이지 중 핫 페이지와 콜드 페이지, 웜 페이지가 각각 몇 개씩 있는지에 따라 블록을 분류한다. 데이터 블록에 연관된 콜드 페이지의 개수가 B_{cold} 개 이상이면 해당 블록을 콜드 블록(cold block)으로 간주하며 데이터 블록이 콜드 블록이 아니면서 핫 페이지 개수가 B_{hot}개 이상인 블록은 핫 블록(hot block)으로 본다. 나머지 블록은 웜 블록(warm block)이다. (그림 2)에서 데이터 블록 MB0, MB1, MB2, MB3은 로그 블록에 갱신된 페이지를 가지고 있다. 각 데이터 블록에 연관된 로그 버퍼의 페이지들을 (hot, warm, cold) 순으로 표시하면 데이터 블록 MB0, MB1, MB2, MB3 에는 각각 (0, 6, 0), (1, 0, 3), (1, 2, 0), (3, 1, 0) 개의 페이지가 로그 버퍼에 위치한다. 그러므로 B_{hot}과 B_{cold} 값을 각각 2라고 가정하면 MB1은 콜드 블록이며 MB3는 핫 블록, MB0과 MB2는 웜 블록이다. 이러한 방식으로 데이터 블록을 구분하여 핫 블록을 제외한 콜드 블록과 웜 블록을 병합의 대상으로 삼는다.

3.1.2 데이터 블록으로의 페이지 이동

가비지 컬렉션에 의해서 로그 블록 내의 각 페이지는 만약 연관된 데이터 블록이 콜드 블록이라면 해당 페이지는 모두 MLC 영역으로 이동된다. 연관된 데이터 블록이 웜 블록인 경우에는 유효한 페이지의 개수에 따라서 이동 여부를 결정한다. 예를 들어 (그림 2)(a)에서 MB0은 2개 페이지가 유효하며 MB2는 5개의 유효한 페이지를 가지고 있다. 데이터 블록에 속한 페이지의 수가 많을수록 SLC 로그 버퍼의 공간은 적게 차지하는 반면 병합 연산을 수행할 때 새로운 블록을 할당받아 MLC 영역 내부에서 복사해야 할 페이지 수가 늘어난다. 그렇기 때문에 데이터 블록 내부에 유효한 페이지가 많을수록 병합 연산을 지연시키는 것이 유리하다.



(그림 2) SLC 로그 버퍼의 가비지 컬렉션

그러므로 본 기법에서는 웜 블록의 유효한 페이지가 Θ개 이상일 때에는 로그 블록의 페이지를 데이터 블록으로 이동시키지 않는다. Θ가 4라고 가정하면 MB2는 병합 대상에서 제외되며 MB0을 대상으로 선정하여 로그 버퍼의 0, 1, 2, 3, 4, 6 페이지가 데이터 블록으로 복사되고 예전 페이지는 무효화 된다. 페이지 이동을 마치면 모든 페이지가 무효화된 MB0과 MB1은 삭제되어 빈 공간이 된다. Θ는 MLC 데이터 블록으로 이동하는 페이지의 양을 결정하며 페이지가 이동한 양에 따라 로그 버퍼에 확보되는 여유 공간의 양이 변하므로 가비지 컬렉션 발생 빈도에도 영향을 미친다.

3.1.3 로그 버퍼 내부에서의 페이지 이동

데이터 블록으로 로그 버퍼 내부의 페이지들을 이동시키면 로그 버퍼 내의 많은 페이지가 무효화된다. 그러므로, 로그 블록의 유효한 페이지들만 새로운 로그 블록에 복사한 다음 블록을 삭제하여 로그 버퍼의 여유 공간을 확보할 수 있다. 만약 하나의 로그 블록의 모든 페이지가 무효화되었다면 새 로그 블록에 페이지를 복사하는 비용 없이 블록을 삭제할 수 있다. 반면 로그 블록의 페이지가 대부분 유효하다면 새로운 로그 블록에 복사할 페이지가 많아질 것이다. 하지만 무효화된 페이지의 수는 적기 때문에 새로 확보할 수 있는 공간이 줄어 연산 비용 대비 효과가 감소된다. 즉 로그 블록의 유효한 페이지 수에 따라 로그 버퍼 내부에서의 페이지 이동의 효과가 달라지므로 로그 블록이 기준 값 δ개보다 적은 수의 유효한 페이지를 가지고 있는 경우에만 로그 블록의 페이지를 새로운 로그 블록으로 이동시키고 그렇지 않은 경우에는 로그 블록을 그대로 유지하는 정책을 사용한다. (그림 2)(c)는 δ값을 2라고 가정하고 SB3의 페이지 20은 SB5로 복사되며 SB0과 SB1, SB3을 삭제하여 로그 버퍼의 여유 공간을 확보하는 것을 보여주고 있다.

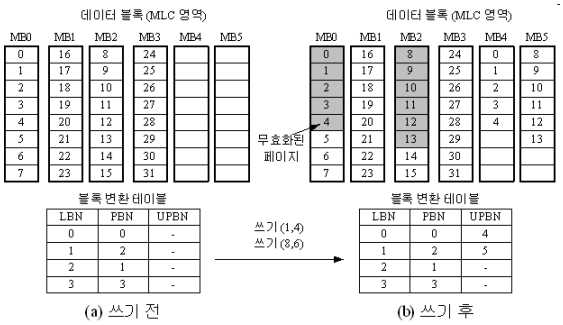
3.2 데이터 블록에 직접 쓰기

FAST에서는 순차 로그 블록에서 순차적인 쓰기 요청을 처리함으로서 이후 합병 연산 시 부분 합병이나 교체 합병을 사용할 수 있다. 그러나 본 논문에서는 모든 로그 버퍼는 SLC 영역에 할당하고 모든 데이터 블록은 MLC 영역에 할당하므로 로그버퍼에 순차 로그 블록을 할당해도 교체 합병 기법을 사용할 수 없다. 그러므로 순차적인 데이터를 처리하기 위해서 SLC 영역에 순차 로그 블록을 두는 대신 SLC 로그 버퍼를 거치지 않고 직접 데이터 블록에 기록하는 우회(bypassing) 기법을 사용한다.

일반적으로 크기가 큰 쓰기 요청은 낮은 시간 지역성을 갖는 순차적인 쓰기인 경우가 많다[7]. 그러므로 데이터를 SLC 로그 버퍼에 기록한 뒤 갱신되지 않은 채 MLC 영역의 데이터 블록으로 이동시키는 대신에 처음부터 MLC 영역에 기록한다면 불필요한 이동 연산을 줄일 수 있다. 본 기법에서는 쓰기 요청의 섹터 길이가 기준 값 α 이상인 경우에는 해당 쓰기 요청을 SLC 로그 버퍼가 아닌 MLC 영역의 데이터 블록에 바로 보낸다. 대부분의 파일시스템은 저장장치에 대용량의 데이터를 저장할 때 쓰기를 몇 개의 요청으로 나누어 비교적 작은 크기의 쓰기 요청 여러 개로 처리한다. 플래시 메모리의 특성상 데이터의 갱신이 되지 않기 때문에 블록 단위 변환을 사용하는 데이터 블록에서 작은 크기의 연속적인 쓰기 요청은 많은 양의 페이지 복사를 유발할 수 있다. 그러므로 업데이트 블록(Update block)을 추가로 할당하여 연속적인 쓰기 요청을 처리할 수 있다.

예를 들어 (그림 3)에서는 쓰기(1, 4)(논리적인 페이지 오프셋 1부터 4개의 페이지를 쓰기)가 FTL에 요청되었을 때 업데이트 블록 MB4를 할당하여 0, 1, 2, 3, 4의 다섯 페이지를 적고 MB0의 해당 페이지들을 무효화 시킨다. 쓰기 요청이 없었음에도 0번 페이지를 기록하는 것은 플래시 메모리의 물리적 특성 상 1, 2, 3번 페이지를 기록한 이후 0번 페이지를 쓸 수 없기 때문이다. 0번 페이지를 복사해두는 것으로 업데이트 블록을 합병할 때 새로운 블록을 할당받아 복사할 필요 없이 교체 합병이나 부분 합병을 사용할 수 있다. 블록 변환 테이블은 업데이트 블록을 처리하기 위해서 각 논리 블록 주소(LBN: Logical Block Number)마다 실제 물리 블록 번호(PBN: Physical Block Number)와 업데이트 블록의 물리 블록 번호(UPBN: Update Physical Block Number)를 갖는다.

업데이트 블록을 모두 사용하여 빈 공간이 없는 경우 데이터 블록을 삭제하고 변환 정보를 바꾸어 업데이트 블록을 데이터 블록으로 교체한다. 또한 업데이트 블록의 유효한



(그림 3) MLC 영역의 업데이트 블록 사용

페이지에 새로운 쓰기 요청이 들어온 경우에도 업데이트 블록을 데이터 블록과 합병하여 하나의 업데이트 블록만 데이터 블록에 할당되도록 한다. 전체 업데이트 블록의 최대 개수를 제한하여 새로 업데이트 블록을 할당할 때 그 개수를 넘어서는 경우에는 이전에 할당되었던 업데이트 블록을 합병한다. 합병되는 업데이트 블록은 비용을 최소화하기 위해서 비어있는 페이지가 가장 적은 것을 선정한다.

쓰기 요청을 SLC 로그 버퍼에서 처리할 것인지 아니면 MLC 데이터 블록으로 바로 보낼 것인지를 결정하기 위해서 본 기법에서는 쓰기 요청의 길이(L)와 MLC 데이터 블록에서 마지막으로 쓰인 페이지로부터의 오프셋(O)을 참조한다. 예를 들어 쓰기(1, 4)에 이어 연속적으로 쓰기(6, 2) 가 요청된 경우 이전에 쓰인 마지막 오프셋이 4이므로 이전에 쓰인 페이지로부터의 오프셋은 1이 된다. 쓰기 요청의 길이 L이 α보다 크며 이전에 쓰인 페이지 오프셋과 현재 쓰기 요청의 첫째 페이지 오프셋의 차이 O가 β보다 작은 경우, 해당 쓰기 요청을 순차적인 쓰기로 간주하며 SLC 로그 버퍼를 거치지 않고 직접 MLC 데이터 블록에 기록한다. 즉 데이터 블록에 직접 쓰기를 수행하는 조건은 다음과 같다.

$$L \geq \alpha \quad \text{and} \quad O \leq \beta$$

α와 β의 값을 바꾸어 주는 것으로 SLC 로그 버퍼의 사용량을 조절할 수 있다. 만약 SLC 영역에 쓰기 요청이 너무 많아 로그 버퍼가 너무 빨리 소모된다면 α값을 감소시키고 β을 증가시킴으로써 더 많은 쓰기 요청이 SLC 로그 버퍼를 우회하도록 할 수 있다. 반대로 MLC 데이터 블록에 너무 많은 쓰기 요청이 발생하여 MLC 내부에서의 병합 연산이 잦은 경우 α값을 증가시키고 β을 감소시켜 MLC에의 쓰기 요청을 줄일 수 있다. 이러한 기법을 사용하여 SLC 영역의 블록들과 MLC 영역의 블록들의 쓰기/지우기 횟수를 조절할 수 있다.

3.3 적응적 가비지 컬렉션

각 저장장치는 사용되는 분야, 응용프로그램, 파일 시스템 등에 따라 다양한 패턴의 쓰기 요청을 처리한다. 때문에 앞서 제시한 기법에서 사용하는 기준 값 Θ, P_{cold}, B_{cold} 등의 최적 값은 항상 달라진다. 그러므로 고정된 기준 값이 아닌 현재 저장 공간의 사용 양태에 따른 기준 값을 적용해주어야 한다.

위에서 제시한 기법은 데이터를 논리 블록 단위로 구분하여 블록 내의 데이터들이 최근에 자주 접근되는지 아닌지를 살피는 것으로 시간 지역성을 활용하여 로그 버퍼를 관리한다. 만약 쓰기 요청이 높은 시간 지역성을 가지고 있다면 로그 버퍼 내의 페이지가 대부분 무효화되어 이후 합병 연산에서 수거되는 SLC 로그 블록의 개수가 많아질 것이다. 그러나 쓰기 요청이 낮은 시간 지역성을 갖는 경우, 해당 페이지들이 콜드 페이지가 되어 데이터 블록으로 이동하기 전까지 변경되지 않아 로그 버퍼의 공간을 많이 차지하며, 합병 연산에 의해 수거되는 SLC 로그 블록의 개수는 줄어들어 빈번한 합병 연산이 발생하게 된다.

본 논문에서는 Θ와 P_{cold}값을 실행 시간에 조절하여 SLC 로그 버퍼를 효과적으로 활용하는 기법을 제시한다. SLC 로그 버퍼의 빈 공간이 비교적 적으며 대부분의 데이터가 시간 지역성이 거의 없다면 이 데이터들을 MLC 데이터 블록으로 이동시키는 것이 유리하다. 그러므로 이런 경우에는 Θ와 P_{cold}값을 증가시켜 더 많은 페이지를 SLC 로그 블록에서 MLC 데이터 블록으로 이동시킨다. 반면 SLC 로그 버퍼에 기록된 데이터들이 시간 지역성이 높다면 최대한 해당 페이지들을 SLC 로그 버퍼 내에 유지하는 것이 좋으므로 Θ와 P_{cold}값을 감소시킨다. 합병 연산 시 수거된 블록의 개수가 전체 로그 블록의 갯수의 80% 이상일 때는 Θ와 P_{cold}값을 감소시키며, 30% 이하일 때에는 Θ와 P_{cold}값을 증가시킨다. 실행 시간에 Θ와 P_{cold}값들을 변경함으로서 다양한 쓰기 패턴에 알맞게 콜드 블럭과 웜 블럭을 수거하는 기준을 수정 가능하다.

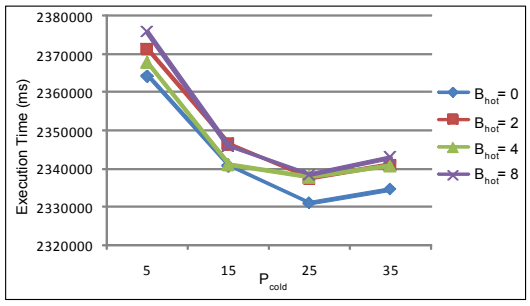
4. 성능 평가

본 논문에서 제안한 기법을 시뮬레이션으로 구현하여 평가하였다. 실험을 위해 마이크로소프트 윈도우XP의 FAT32와 NTFS 환경에서 워드프로세서, 동영상 재생, 웹브라우징, 게임 등의 다양한 데스크탑 응용프로그램을 수행하는 과정에서의 입출력 정보를 수집하였다. 또한 같은 파일을 계속 갱신하는 경우와 대용량의 파일에 랜덤하게 접근하는 경우를 고려하기 위해서 각각 IOzone과 Postmark 벤치마크를 사용하여 입출력 정보를 수집해 실험에 사용하였다. 하이브리드 플래시메모리의 특성은 <표 1>을 참조하였으며 전체 블록의 갯수는 20480개로 설정하였다.

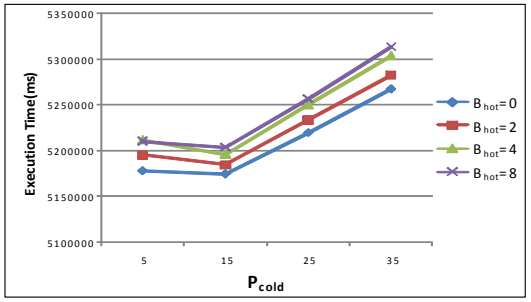
먼저 데이터 블록의 분류가 성능에 미치는 영향을 평가하고자 P_{hot}, P_{cold}, B_{hot}, B_{cold}값을 변경해가며 실험하였다. 또한 로그 버퍼 내부에서의 이동에 대한 성능을 실험하기 위해서 δ를 변경하면서 실행 시간의 변화를 측정하였다. SLC 영역의 블록 갯수(N_{SLC})는 80으로 설정하였다. 모든 실험에서 P_{hot}값은 0으로 고정시켰으며 로그 버퍼를 관리하는 각 기준

값들에 따른 변화를 살펴보기 위해서 데이터 블록에 직접 쓰기 기법을 제외하고 실험하였다. (그림 4)와 (그림 6)은 FAT32 파일시스템에서 (그림 5)과 (그림 7)은 NTFS 파일 시스템에서 각각 측정한 P_{cold}와 B_{hot}, B_{cold}, 그리고 δ에 따른 성능의 변화를 보여준다. 두 파일시스템 모두 B_{hot}이 0일 때 가장 좋은 성능을 보였다. 이는 논리 블록 내의 한 페이지가 한번이라도 갱신이 일어났다면 해당 블록을 핫 블록으로 간주하는 것으로 현재 접근이 일어난 블록이 이후에도 접근될 가능성이 높기 때문에 병합을 하지 않는 것이 유리하다는 것을 보여준다. 반면 P_{cold}와 B_{cold}값이 작아질수록 콜드 블록의 수가 늘어나 MLC 영역으로의 이동이 증가하여 로그 버퍼의 공간을 많이 확보하게 된다. FAT32의 입출력 요청에 대해서 P_{cold}와 B_{cold}가 각각 25, 12일 때 가장 좋은 성능을 보였으나 NTFS에서는 P_{cold}와 B_{cold}가 각각 15, 6일 때 더 좋은 결과가 도출되었다. 이는 데이터의 특성에 따른 것으로 FAT32의 입출력 데이터가 NTFS에 비하여 시간 지역성이 높기 때문이다. δ의 최적 값은 40으로 측정되었다. δ값이 너무 작으면 로그 블록에서 무효화된 페이지가 수거되는 양이 줄어들어 가비지 컬렉션이 빈번하게 발생하며, 너무 큰 경우 SLC 영역의 로그버퍼에서 이동되는 페이지의 수가 늘어나 페이지 복사의 비용이 커진다.

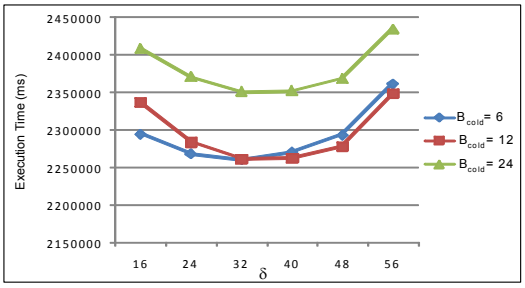
(그림 8~10)은 Θ값이 입출력 연산에 어떤 영향을 미치는지 보여주고 있다. Θ가 0인 경우 웜 블록의 페이지들은 SLC 영역에서 MLC 영역으로 이동하지 않는다. 때문에 병합 연산 시 수거 되는 로그 블록이 적어져서 (그림 10)에서 보는 바와 같이 가비지 컬렉션이 빈번하게 발생한다. Θ값을



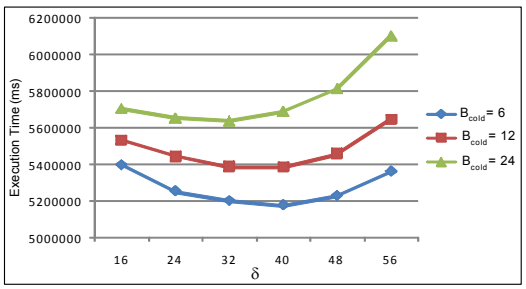
(그림 4) Bhot과 Pcold에 따른 성능 변화(Θ=64, δ=40, Phot=0, Bcold=12, FAT32)



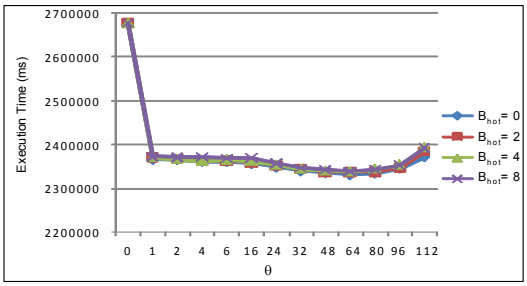
(그림 5) Bhot과 Pcold에 따른 성능 변화(Θ=32, δ=40, Phot=0, Bcold=6, NTFS)



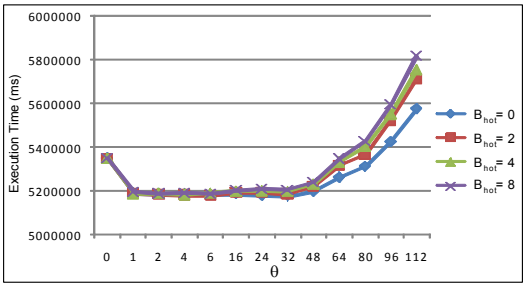
(그림 6) δ에 따른 성능 변화(Θ=64, Pcold=25, Bhot=0, FAT32)



(그림 7) δ에 따른 변화(Θ=32, Pcold=15, Bhot=0, NTFS)

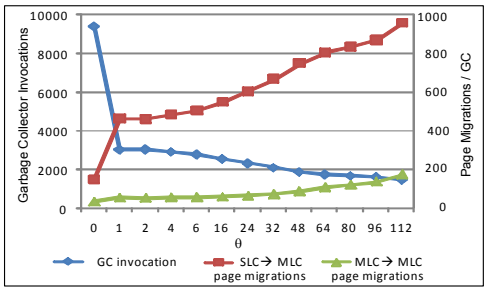


(그림 8) Θ에 따른 성능 변화(δ=40, Pcold=25,Bcold=12, FAT32)

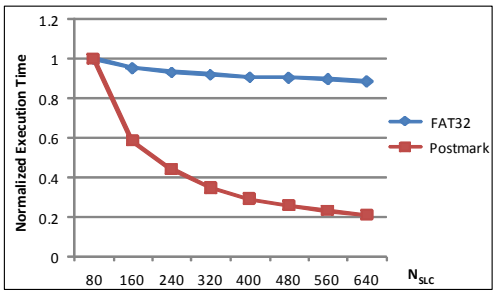


(그림 9) Θ에 따른 변화(δ=40, Pcold=15, Bcold=6, NTFS)

점점 증가시킴에 따라 회수하는 페이지의 숫자가 늘어나 성능이 향상됨을 볼 수 있다. 그러나 FAT32의 경우 Θ값을 64 이상, NTFS의 경우 32 이상으로 증가시키면 전체 성능이 오히려 나빠지는 것을 볼 수 있다. 이는 (그림 10)에서와 같



(그림 10) Θ에 따른 가비지 컬렉션 변화(δ=40, Pcold=25, Bcold=12, FAT32)

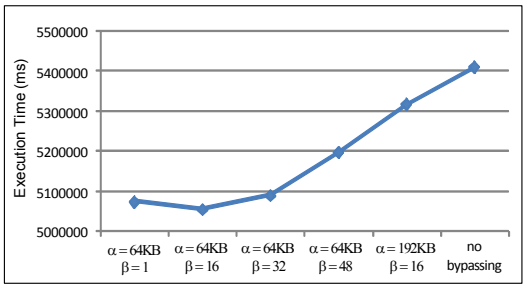


(그림 11) SLC 영역 크기에 따른 성능 비교

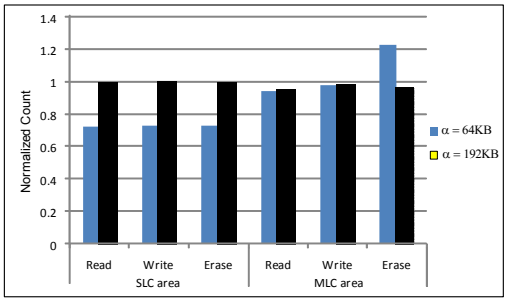
이 MLC에서 MLC로 이동하는 페이지와 MLC에서 SLC로 이동하는 페이지의 수가 필요 이상으로 증가하기 때문이다.

(그림 11)은 SLC 영역의 크기에 따른 실행 시간을 나타낸다. N_{SLC}가 증가함에 따라 로그 버퍼의 공간이 늘어나 가비지 컬렉션 발생 빈도가 줄어들어 실행 시간이 줄어든다. 그러나 FAT32에 비하여 Postmark에 대한 실험에서 성능의 감소폭이 매우 컸다. 그 이유는 FAT32의 입출력 패턴이 높은 시간 지역성을 가지고 있으며 많은 파일을 랜덤하게 쓰고 지우는 Postmark 실험에 비해서 지역성을 잘 활용하기 때문에 SLC 영역의 공간이 크지 않더라도 충분한 성능을 제공하기 때문이다. 반면 Postmark 벤치마크는 입출력 패턴이 불규칙적이기 때문에 SLC 영역이 작을수록 가비지 컬렉션이 자주 발생한다. 일반적인 입출력은 대부분 높은 시간 지역성을 가지기 때문에 SLC 영역의 크기가 작아도 좋은 결과를 보임을 알 수 있다.

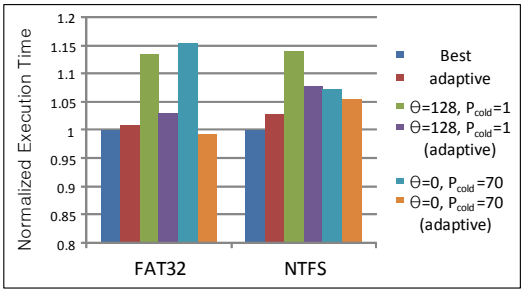
또한 α와 β값을 변경해가면서 로그 버퍼를 거치지 않고 직접 데이터 블록에 기록하는 우회 기법의 성능을 실험하였다. NTFS 파일시스템에서 응용프로그램과 대용량의 파일을 복사하였을 때의 입출력 정보를 실험에 사용하였다. (그림 12)에서는 α와 β값을 변경해가며 I/O 실행시간을 측정하였으며 모든 데이터가 SLC 로그 버퍼를 거치는 것과 비교하여 7% 가량의 성능 향상이 있었다. β값이 작아질수록 MLC 내부에서의 이동과 병합 연산이 줄어들어 성능이 향상된다. 그러나 너무 작은 값 (β=1)을 사용하면 SLC 로그 버퍼에서 처리되는 쓰기 요청이 늘어나 성능이 감소한다. (그림 13)은 우회 기법을 사용했을 때 플래시 메모리의 연산 횟수를 나타낸다. 각각의 결과는 모든 데이터를 SLC 로그 버퍼에서



(그림 12) 로그 버퍼 우회 기법의 성능 변화



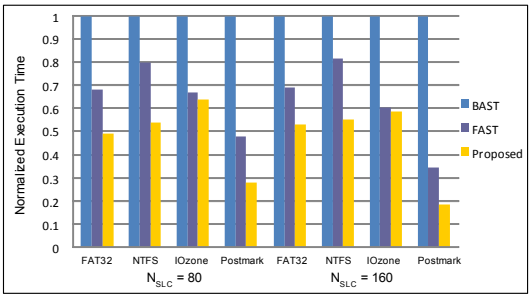
(그림 13) α값에 따른 각 연산 횟수 비교



(그림 14) Θ와 Pcold값을 수정함에 따른 성능의 변화

처리하는 경우에 정규화한 값으로 표현하였다. α값이 192KB일 때 SLC 영역의 연산은 큰 차이가 없는 반면 MLC 영역에의 연산은 다소 감소하였다. 이에 반해 α값이 64KB일 때 SLC 영역에의 연산이 급격하게 줄어들었으며 MLC 영역에서의 삭제 연산은 업데이트 블록 병합에 의하여 크게 증가하였다. 페이지의 읽기와 쓰기 연산은 오히려 감소하였음을 볼 때, 업데이트 블록과 데이터 블록의 교체가 효과적으로 발생하고 있음을 알 수 있다.

(그림 14)은 Θ와 P_{cold}값을 달리해가며 실행 시간에 각 기준 값을 적응적 가비지 컬렉션 기법을 사용하여 수정하였을 때와 고정된 값을 사용하였을 때의 차이를 보여준다. 고정된 값을 사용한 경우는 3가지 경우가 있는 데, 먼저 이전 실험에서 찾아낸 최적의 Θ와 P_{cold}값을 사용한 경우로 FAT32의 경우에는 Θ=64, P_{cold}=12, NTFS는 Θ=32, P_{cold}=12로 설정한 경우이다 (Best). 다른 2가지 경우는 최적 값과 크게 다른 극단적 설정 값을 이용한 경우로 Θ=128, P_{cold}=1로 설정한 경우와 Θ=0, P_{cold}=70으로 설정한 경우이다. 적응



(그림 15) 기존 기법과의 성능 비교

적 가비지 컬렉션 기법도 역시 3가지 경우에 대해서 실험하였는데, 고정 값을 사용한 3가지 경우와 동일한 값으로 파라미터를 처음에 세팅해두고 적응적 기법을 적용한 것이다. FAT32와 NTFS 두 가지 실험 모두 최적 값과 크게 다른 고정된 Θ와 P_{cold}값을 사용하면 성능이 저하됨을 알 수 있다. 반면 적응적 가비지 컬렉션 기법을 사용하였을 때 초기 값이 최적 값과 크게 다름에도 불구하고 좋은 결과로 수렴해가며, 고정적인 최적의 값을 적용하였을 때와 비슷한 성능을 보였다.

마지막으로 본 논문에서 제시한 기법을 이전에 개발된 FTL 기법인 BAST 및 FAST와 비교하였다. 모든 FTL 기법에서 SLC 영역은 로그 버퍼로 사용되었다. (그림 15)은 각각 FAT32, NTFS, IOzone, Postmark를 대상으로 SLC 영역의 크기를 N_{SLC}=80, N_{SLC}=160으로 바꾸어 실험한 결과이다. 입출력 데이터의 패턴에 상관없이 모든 실험에서 제안한 기법이 좋은 성능을 보였다.

5. 결 론

본 논문에서는 하이브리드 플래시 메모리를 위한 새로운 소프트웨어 관리 기법을 제시하였다. 제시된 기법은 하이브리드 변환 FTL을 기반으로 하고 있으며 SLC 영역을 로그 버퍼로 사용한다. 가비지 컬렉션 알고리즘은 로그 버퍼의 페이지들을 지역성과 이동 비용에 따라 선택적으로 수거하여 SLC 영역과 MLC 영역의 공간의 특징을 활용한다. 실험 결과를 통해서 입출력 패턴을 고려하여 알고리즘에 사용되는 여러 가지 변수들의 값을 적절히 지정하는 것이 성능에 큰 영향을 주는 것을 알 수 있었다. 향후 연구 방향으로 SLC 영역과 MLC 영역 크기에 따른 성능과 수명의 비교, 전체 플래시 메모리의 수명을 조절하는 기법 등을 고려하고 있다.

참 고 문 헌

[1] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho. “A space-efficient flash translation layer for compact flash systems,” IEEE Transactions on Consumer Electronics, Vol.48, No.2, pp.366-375, 2002.

[2] T. Choe et al. “A dual-mode NAND flash memory: 1-Gb multilevel and high-performance 512-Mb single-level modes,” IEEE Journal of Solid-State Circuits, Vol.46, Issue 11, 2001.

[3] L. Chang. “Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs,” Proc. of Asia and South Pacific Design Automation Conference (ASPDAC), pp.428-433, 2008.

[4] Samsung Electorinics, 4Gb Flex-OneNAND M-die, http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Product_FlexOneNAND.html

[5] Toshiba America Electronic Components, Inc., mobileLBA-NAND, http://www.toshiba.com/taec.

[6] S. W. Lee, D. J. Park, T. S. Chung, W. K. Choi, D. H. Lee, S. W. Park, and H. J. Song. “A log buffer based flash translation layer using fully associative sector translation,” ACM Transactions on Embedded Computing Systems, Vol.6, No.3, 2007.

[7] S. Lee, D. Shin, and J. Kim. “LAST: locality-aware sector translation for NAND flash memory-based storage systems,” Proc. of SPEED'08, Salt Lake City, Utah Feb., 2008.

[8] J. U. Kang, H. Jo, J. S. Kim, and J. Lee. “A superblock-based flash translation layer for NAND flash memory,” in Proc. International Conference on Embedded Software, pp.161-170, 2006.

[9] S. Y. Park, W. Cheon, Y. Lee, M. S. Jung, W. Cho and H. Yoon. “A Re-configurable FTL (Flash Translation Layer) Architecture for NAND Flash based Applications,” in Proc. of International Workshop on Rapid System Prototyping, pp.202-208, 2007.

[10] J. Lee, S. Kim, H. Kwon, C. Hyun, S. Ahn, J. Choi, D. Lee, and S. H. Noh, “Block Recycling Schemes and Their Cost-based Optimization in NAND Flash Memory Based Storage System,” Proc. of EMSOFT'07 Salzburg, Austria, Sep., 2007.

[11] S. H. Park, J. W. Park, J. M. Jeong, J. H. Kim, and S. D. Kim. “A mixed flash translation layer structure for SLC-MLC combined flash memory system,” Proc. of SPEED'08 Salt Lake City, Utah, Feb., 2008.

임 수 준

e-mail : lang33@skku.edu

2008년 성균관대학교 컴퓨터공학과(학사)

현 재 성균관대학교 전기전자컴퓨터공학과 석사과정

관심분야 : 임베디드 시스템, 저전력 시스템 설계, 플래시 메모리 관리 기법

신 동 군

e-mail : dongkun@skku.edu

1994년 서울대학교 계산통계학과(학사)

2000년 서울대학교 대학원 전산과학과 (이학석사)

2004년 서울대학교 대학원 컴퓨터공학부 (공학박사)

2004년~2007년 삼성전자 소프트웨어연구소 책임연구원

2007년~현 재 성균관대학교 정보통신공학부 전임강사

관심분야 : 임베디드 시스템, 실시간 시스템, 저전력 시스템 등