http://dx.doi.org/10.3745/KIPSTA.2011.18A.4.135

동적 사상 테이블 기반의 버퍼구조를 통한 Solid State Disk의 쓰기 성능 향상

조 인 표[†] · 고 소 향[††] · 양 훈 모[††] · 박 기 호[†††] · 김 신 덕[††††]

요 약

본 연구는 플래시 메모리 기반의 고성능 SSD (Solid State Disk) 구조를 위하여 디스크 참조 특성에 적응적으로 구동하는 효율적인 버퍼 구조와 구동 기법을 설계한다. 기존 SSD는 삭제동작 횟수의 제약은 물론 읽기와 쓰기 동작에 대하여 비대칭적인 성능을 보이는 특징을 갖고 있다. 이러한 삭제동작 횟수와 쓰기 동작의 지연시간을 최소화 하기 위해서는 다중 플래시 메모리 칩들에 대해 쓰기 동작은 병렬적으로 수행하는 정도를 최대화하여 운영하여야 한다. 따라서 플래시 메모리 칩들에 대한 인터리빙 레벨 (interleaving level)을 최대화 하기 위하여, 본 논문에서는 혼합 위치 사상 기법 (hybrid address mapping)과 슈퍼 블록 (super-block) 기반의 SSD 구조에 대하여 성능 증대와 증가된 장치 수명을 제공하기 위한 효율적 버퍼 구조를 제안한다. 제안한 버퍼구조는 응용 수행특성을 기반으로 최적의 임의/순차쓰기를 구분하며, 수행 성능에 중요한 순차쓰기 정도의 크기를 증대시키는 동적 융합 방법, 구동되는 버퍼구조와 사상 테이블의 효율적인 관리 구조를 설계하였으며, 이를 통해 기존의 단순한 버퍼 운영기법에 비하여 35%의 성능향상을 제공한다.

키워드 : 대용량 저장장치, 비 휘발성 메모리, 버퍼 관리 구조, 인터리빙, 성능 분석

A Buffer Architecture based on Dynamic Mapping table for Write Performance of Solid State Disk

In-Pyo Cho[†] · So-Hyang Ko[††] · Hoon-Mo Yang[††] · Gi-Ho Park[†††] · Shin-Dug Kim[††††]

ABSTRACT

This research is to design an effective buffer structure and its management for flash memory based high performance SSDs (Solid State Disks). Specifically conventional SSDs tend to show asymmetrical performance in read and /write operations, in addition to a limited number of erase operations. To minimize the number of erase operations and write latency, the degree of interleaving levels over multiple flash memory chips should be maximized. Thus, to increase the interleaving effect, an effective buffer structure is proposed for the SSD with a hybrid address mapping scheme and super-block management. The proposed buffer operation is designed to provide performance improvement and enhanced flash memory life cycle. Also its management is based on a new selection scheme to determine random and sequential accesses, depending on execution characteristics, and a method to enhance the size of sequential access unit by aggressive merging. Experiments show that a newly developed mapping table under the MBA is more efficient than the basic simple management in terms of maintenance and performance. The overall performance is increased by around 35% in comparison with the basic simple management.

Keywords : Solid State Disks, Non-Volatile Memory, Buffer Management, Interleaving, Performance Evaluation

1. 서 론

최근 차세대 저장장치로서 비 휘발성 NAND 플래시 메모리 기반의 SSD (Solid State Disk) 저장장치의 사용이 증가하고 있다. 이러한 플래시 메모리 기반 저장장치는 하드 디스크와 비교하여 빠른 연산속도, 저 전력소비 등의 장점을 가지고 있다. 그러나, <표 1>에서 보는 바와 같이NAND 플래시 메모리의 삭제연산은 읽기와 쓰기 연산에 비해 상대

※ 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2011-0002536).
† 준 회 원 : (주)고영테크놀러지 R&D
†† 준 회 원 : 연세대학교 컴퓨터과학과 박사과정
††† 정 회 원 : 세종대학교 전자정보공학대학 컴퓨터공학과 조교수
†††† 종신회원 : 연세대학교 공과대학 컴퓨터과학과 교수
논문접수 : 2011년 1월 12일
수 정 일 : 1차 2011년 3월 21일
심사완료 : 2011년 5월 31일

적으로 많은 시간이 소요된다. 삭제연산 횟수와 쓰기 동작의 지연시간을 최소화 하기 위해서는 여러 독립된 메모리 칩(chip)에 대하여 동시 쓰기 동작을 병렬적으로 수행하는 정도를 나타내는 인터리빙 레벨 (interleaving level)을 최대화 하는 것이 요구된다. 따라서 본 논문에서는 혼합 위치 사상 (hybrid address mapping) 기법과 슈퍼 블록(super block) 구성을 동시에 적용하는 SSD에 대하여 더 나은 성능과 수명을 제공하기 위한 버퍼 구조와 관리기법을 제안한다. 본 연구에서는 기존 디스크 참조 시 발생하는 두 가지 주요한 수행 특징인, 특정 논리 주소에 집중적으로 발생하는 쓰기 동작과 순차 접근의 시간적 분산 특징을 해결하기 위하여, 순차와 랜덤 참조에 배정되는 버퍼 공간을 동적으로 운영하기 위한 사상 테이블을 구성하여 논리 슈퍼 블록 주소별로 데이터를 임시 저장, 운영한다.

또한 제안한 버퍼구조 내에 사상 테이블의 효율적인 관리와 동작을 위한 구조가 구현되었으며 이를 통해 기존의 단순한 버퍼에 비하여 35%의 성능향상을 보인다. 이러한 삭제 횟수의 최소화와 쓰기 성능의 개선을 위해서는 여러 메모리 칩에 걸쳐 병렬적으로 쓰기 작업을 수행해야 한다. 병렬적인 쓰기 작업을 위해서는 여러 플래시 메모리 칩으로 SSD를 구성해야 하며, 최대한으로 여러 플래시 메모리 칩이 병렬적으로 쓰기 작업을 수행할 수 있어야 한다. 이러한 메모리 칩의 병렬성에 대한 적용 정도를 인터리빙 레벨 이라 한다. 일반적으로 인터리빙 레벨을 최대화 시키면 SSD의 수명연장과 순차 성능향상을 제공하게 되며, 이를 위한 기존의 방법으로 다양한 위치 사상 방법과 뱅크 구성 기법들이 제안되어 왔다. 이 기법들 중에서 하이브리드 위치 사상과 뱅크 구성 기반으로 구동되는 슈퍼 블록 구성 운영이 최적의 기법으로 제안되었다[1,2,5]. 본 논문에서는 하이브리드 위치 사상 기법과 슈퍼 블록 구성 방법을 적용한 SSD의 기본구조에 구동되는 적응적 버퍼구조를 제안한다. 이 두 기법을 채택한 구조는 논리적으로 하나의 슈퍼 블록으로 조합되는 데이터만 병렬적으로 물리적인 슈퍼 블록으로 쓰기 작업이 수행될 수 있으므로 논리 슈퍼 블록 주소에 따라 데이터를 모아두고 최대한의 데이터를 병렬적으로 SSD에 쓰기 작업을 수행할 수 있도록 한다. 이와 같은 목표를 위하여 본 연구에서는 버퍼 구조 내에 동적 사상 테이블을 운영함으로써 버퍼 공간의 적응적 활용성과 인터리빙 레벨을 향상시킨다. 동적 사상 테이블을 이용한 제안한 버퍼 구조는 기존의 특별한 사상 테이블을 사용하지 않는 구조에 비하여 35% 쓰기 성능 향상시키고 삭제 동작을 유발시키는 주 원인인 블록 병합 동작을 80% 감소시킨다

본 논문의 구성은 다음과 같다. 2장에서는 버퍼 구조의 대상이 되는 SSD 구조에 쓰이는 기법들을 정리하였다. 3장에서는 버퍼 구조를 제안하기에 앞서 실험 대상 응용들의 쓰기 동작에 대한 특징과 분포를 분석하였다. 4장에서는 제안하는 버퍼의 자세한 구조와 구동방식을 설명하였다. 5장에서는 제안한 구조와 기존의 버퍼구조를 비교하여 성능평가를 제시하였다. 마지막으로 6장에서 결론을 도출하였다.

〈표 1〉 저장 매체의 연산 비교

저장 매체	연산 시간		
	읽기 (512B)	쓰기 (512B)	삭제
DRAM	2.56 μs	2.56 μs	해당없음
NAND 플래시	25 μs	200 μs	1.5 ms
디스크	12.4 ms	12.4 ms	해당없음

2. Flash Translation Layer와 Hybrid Super-block 구성

NAND 플래시 메모리의 비대칭적인 성능구조와 칩의 수명을 감소시키는 삭제 동작을 최소화하고 감추기 위해서는 여러 메모리 칩 및 블록에 걸쳐서 쓰기 동작이 이루어져야 한다. 이러한 동작을 수행하는 SSD내에 구동 시스템 소프트웨어가 FTL (flash translation layer)이다. FTL은 쓰기동작에 의해 작성되는 데이터가 최대한의 마모도 평준화 (wear leveling)가 발생할 수 있도록 위치 사상 관리와 블록 구성을 관리한다. 위치 사상은 섹터 사상 (sector mapping) [11], 블록 사상 (block mapping) [6,8], 하이브리드 사상 (hybrid mapping) [9,10]으로 나눌 수 있다. 일반적으로 파일 시스템으로부터 전달되는 디스크 참조 요청들은 읽기와 쓰기 동작의 단위는 섹터 단위로 이루어진다. 섹터 사상은 섹터 단위로 테이블을 구성하여 사상 하는 방식으로서 다른 사상 방법들에 비해 가장 효율적인 마모도 평준화가 이루어질 수 있다[11]. 그러나 섹터 단위로 논리주소가 물리주소에 대응되어야 하기 때문에 테이블의 사이즈가 매우 커질 수 있다. 이러한 테이블은 SSD 내에서 SRAM (static random access memory)에 상주하게 되는데 보통 SRAM은 빠른 접근과 단가의 제한으로 인하여 작은 사이즈로 설계된다. 따라서 큰 사이즈를 갖는 사상 테이블은 설계에 있어서 부담이 되는 요소이다. 두 번째로 블록 사상은 블록 단위로 논리주소와 물리주소를 사상 시켜준다. 섹터 사상에 비하여 테이블 사이즈의 크기는 현저히 적으나 효율적인 마모도 평준화를 수행하지 못하여 섹터 사상에 비해 많은 삭제 동작을 발생시킨다[6]. 하이브리드 사상 기법은 블록 사상 기법과 섹터 사상 기법을 함께 사용하는 방식이다. 블록 사상과 같이 블록 단위로 테이블을 구성하는 것은 같지만 블록 마다 여분의 공간을 두어 새롭게 작성하는 데이터의 논리주소의 전체를 비교함으로써 블록 내에 비어있는 새롭게 쓰기 동작이 이루어 질 수 있도록 한다[8]. <표 2>는 세가지 주소 사상 기법에 대해 요구되는 테이블 크기를 보여준다 [1]. 따라서 FTL은 위치 사상 테이블을 관리함과 동시에 칩의 구성에 대한 관리도 수행한다. 기억 장치를 병렬 모듈로 구성하여 동시에 접근이 가능하도록 하는 인터리빙 레벨을 최대화 하기 위해서는 단순하게 순차적으로 쓰기작업을 병렬적으로 수행시키는 단순 인터리브 구조와 여러 칩들에 걸쳐 블록들을 하나의 큰 사이즈의 블록으로 바라보는 슈퍼 블록 구조가 있다[2].

하이브리드 슈퍼 블록 구성은 슈퍼 블록 구성과 하이브리드 위치 사상 방법을 동시에 적용한 기법이다. 기본적으로 모든 블록들은 슈퍼 블록단위로 함께 묶여서 구성된다. 이 블록들 중 소수는 로그 슈퍼 블록으로 사용되며 로그 슈퍼 블록에 쌓인 데이터는 병합 동작을 통해 데이터 슈퍼 블록으로 변환된다[7]. 하이브리드 위치 사상 기법에 따라, 모든 슈퍼 블록은 논리적인 주소에 따라 하나의 물리적인 슈퍼 블록에 사상된다.

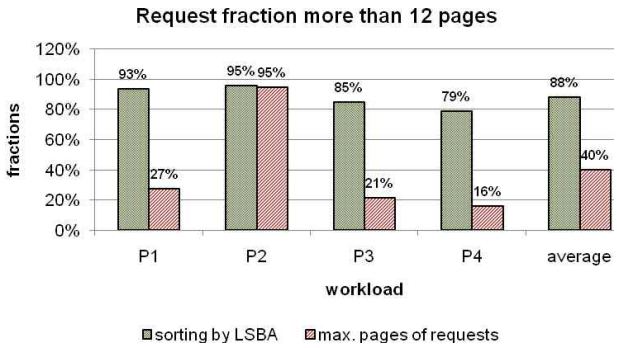
〈표 2〉 사상 기법에 따른 요구 테이블 사이즈

	Bytes for addressing		Total	
	128MB	8GB	128MB	8GB
Sector mapping	3 B	3 B	768KB	48MB
Block mapping	2 B	3 B	16KB	1536KB
Hybrid mapping	(2+1) B	(3+1) B	272KB	17920KB

각 섹터는 여유 공간을 갖게 되며 이 공간을 이용하여 새로 작성되어지는 데이터가 이미 쓰여져 있는 데이터 인지 완전히 새로운 데이터 인지 판단하게 된다. 그러므로 하이브리드 슈퍼 블록 구성에서 데이터가 병렬적으로 함께 최대 뱅크의 수 만큼 쓰려면 논리적으로 같은 슈퍼 블록 주소를 가져야 하기 때문에 버퍼 내에서부터 사상 테이블을 이용하여 논리적인 주소에 따라 데이터를 취합, 유지 함으로써 인터리빙 레벨을 향상 시킬 수 있다. 또한 버퍼 내에 일부 논리 주소로 집중되는 현상을 해결하기 위해 동적으로 사상 테이블을 구성함으로써 버퍼 공간의 활용 성을 높일 수 있다는 개선점을 갖고 있다. 본 논문에서는 하이브리드 슈퍼 블록 구성을 적용한 SSD에 대하여 이와 같은 개선점을 활용하기 위한 동적 사상 테이블을 이용한 효과적인 버퍼구조를 제안하고 자세한 구현 사항을 설명한다.

3. 대상 응용들의 쓰기 동작 패턴과 그에 따른 특성

SSD의 수행성능을 향상시키기 위해서는 기본적인 읽기와 쓰기 동작이 성능 요구조건을 충족시켜야 하며, 수행 시간을 최적화하는 지능적 버퍼 관리 구조가 요구된다. 따라서 상황에 지능적인 버퍼구조를 제안하기 위해 대상 응용 4가지를 선정하고 이 응용들의 특징을 파악해야 한다. 이 응용들은 PCMark05의 HDD 성능 테스트를 위해 선정된 벤치마크 중에서 버퍼구조에 따른 성능 효과를 볼 수 있을 정도로 긴 요청 사이즈를 갖는 4가지로 선택되었다. 첫 번째 트레이스 (P1)는 윈도우 업데이트에 대한 트레이스로서 여러 논리주소에 거쳐 작은 양의 임의적 쓰기 동작과 읽기 동작이 발생한다. 두 번째 트레이스 (P2)는 대량의 미디어 파일을 웹을 통해 다운로드 받는 트레이스로서 특정 논리주소에 쓰기 및 읽기 동작이 집중된다. 세 번째 트레이스 (P3)는



(그림 1) LSBA별로 누적 작성 페이지양과 쓰기 요청당 페이지수의 비교

〈표 3〉 대상 응용들의 트레이스 정보

Trace number	P1	P2	P3	P4	Average
Trace detail	Window Updates	Media download	Compress Decompress	General usage	
Request size (MB)	2065.98	2054.86	858.00	1458.54	1850.31
Write (%)	47%	99%	49%	32%	51%
Pages / write request	9.45	14.97	6.65	6.71	9.44
Accessed LSBA (%)	12%	13%	17%	16%	15%

100개의 파일에 대하여 압축을 수행하고 다시 그 압축을 해제한 트레이스로서 두 번째 트레이스와 마찬가지로 특정 논리주소에 쓰기 및 읽기 동작이 집중된다. 마지막으로 네 번째 트레이스 (P4)는 인터넷 브라우저, 미디어 플레이어, 워드 프로그램을 동시에 다중처리를 수행하여 취합된 것이다. 즉 선택된 트레이스들은 최대의 순차/임의 수행 특성을 포괄적으로 대표하는 트레이스로 구성되었다.

이 응용들의 분석을 통해 저장장치 쓰기 접근에 대한 두 가지 특징을 찾아낼 수 있다. 첫 번째 특징은 저장장치의 전체 논리 슈퍼 블록 주소 중에서 하나의 응용마다 평균 15%만이 집중적으로 접근 된다는 것이다. <표 3>은 각 응용들의 트레이스에 대한 총 트레이스 사이즈, 트레이스 중에서 쓰기 동작의 발생 비율, 한번의 쓰기 요청당 쓰여지는 페이지 수, 그리고 접근된 논리 슈퍼 블록 주소(Logical Super Block Address : LSBA)의 통계를 보여준다. 특히, 접근된 논리 슈퍼 블록 주소의 경우 총 논리 슈퍼 블록 주소 중 평균적으로 15%가 쓰기동작에 의하여 접근 된다. 두 번째 특징은 랜덤 쓰기 및 랜덤 읽기 동작으로 인해서 본래의 순차 쓰기 동작은 시간적으로 분석하였을 경우, 분할되어 진행된다는 것이다. 랜덤 동작은 여러 논리 주소에 걸쳐 1페이지 이하의 데이터를 작성하며 순차 쓰기의 비하여 잦은 덮어 쓰기 동작을 발생시킨다. 순차 쓰기는 하나의 논리 주소에 1페이지 초과의 데이터를 작성하며 랜덤 동작에 비하여 적은 덮어 쓰기 동작을 발생시킨다. (그림 1)은 요청에 따른 최대 페이지와 논리 슈퍼 블록주소에 따른 누적 양에 대하여 12페이지 이상의 페이지를 포함하게 되는 논리 슈퍼 블록 주소의 비율이다. X-축은 사용된 벤치마크를 보여주며, Y-축은 12페이지 이상의 요청 발생 빈도를 제공한다.

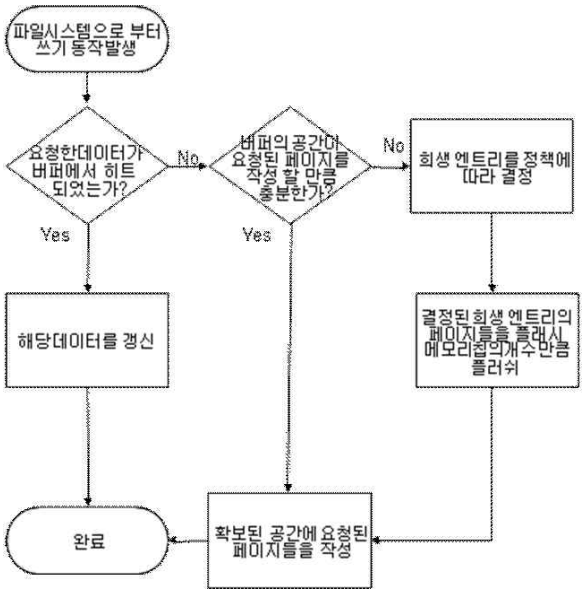
또한 일반적인 32GB의 하드디스크에서 발생한 논리적 접근을 통계화한 것이며 요청에 따른 최대 페이지와 논리 슈퍼 블록 주소에 따른 누적 양에 대하여 12페이지 이상의 페이지를 담게 되는 논리 슈퍼 블록 주소의 비율을 보여준다. 그림에서와 같이 한번의 쓰기 요청당 쓰여지는 페이지양은 누적 양에 비하여 48% 적게 12페이지 이상의 요청을 하지 않는다. 즉, 쓰기 요청은 랜덤 쓰기 요청에 의하여 간섭되고 그에 따라 전체 쓰기 동작 요청당 수행되는 페이지 수가 적어짐을 의미한다.

본 논문은 이 두 가지 특징을 적응적 그리고 동적으로 이용하는 버퍼구조를 제안한다. 첫 번째 특징인 특정 논리 주소에 집중적으로 발생하는 쓰기 동작에 대한 버퍼 동작을 효율적으로 처리하기 위해서 동적으로 페이지 단위에 의거 버퍼 공간을 할당한다. 두 번째 특징인 랜덤 접근으로 인한 순차 접근의 분할을 해결하기 위해 버퍼에서 사상 테이블을 구성하여 논리 슈퍼 블록 주소별로 데이터를 임시 저장한다.

4. 동적 사상 테이블을 이용한 버퍼 구조

동적 사상 테이블을 이용한 버퍼 구조의 동작은 데이터를 버퍼에 저장하는 과정과 저장된 데이터를 플래시 메모리 칩으로 옮겨 저장하는 (flushing operation) 동작으로 나눌 수 있다. 이를 운영하는 버퍼구조는 SSD 내부에 플래쉬 메모리 칩 모듈 상단에 위치하고 있으며, 버퍼의 대한 동작은 SSD의 프로세서가 운영한다. 버퍼에서의 동작을 나타내는 구동 동작 흐름 차트는 (그림 2)와 같다. 파일 시스템으로부터 쓰기 동작이 발생되면 다수의 페이지가 버퍼로 전송된다. 이 페이지들이 버퍼에서 적중 (hit) 할 경우 버퍼내의 해당 데이터들을 갱신하고, 쓰기 동작이 완료된다. 버퍼 내에 새롭게 쓰여져야 할 경우, 이 페이지들을 모두 버퍼에 보관할 충분할 공간이 확보되어 있는지 확인한다. 충분한 공간이 없을 경우 버퍼 내 공간 관리정책에 의해 선택된 하나의 논리 슈퍼 블록 주소를 선정하고, 이 곳에 링크되어 버퍼링 되어있는 페이지들을 플래시 메모리 칩의 개수만큼 플래시 메모리 칩에 옮겨 쓰게 된다. 이 때에 발생하는 옮겨 쓰는 동작을 플러시 동작이라 한다. 플러시 동작을 통하여 확보된 공간에 새롭게 쓰기동작이 수행되고 완료된다.

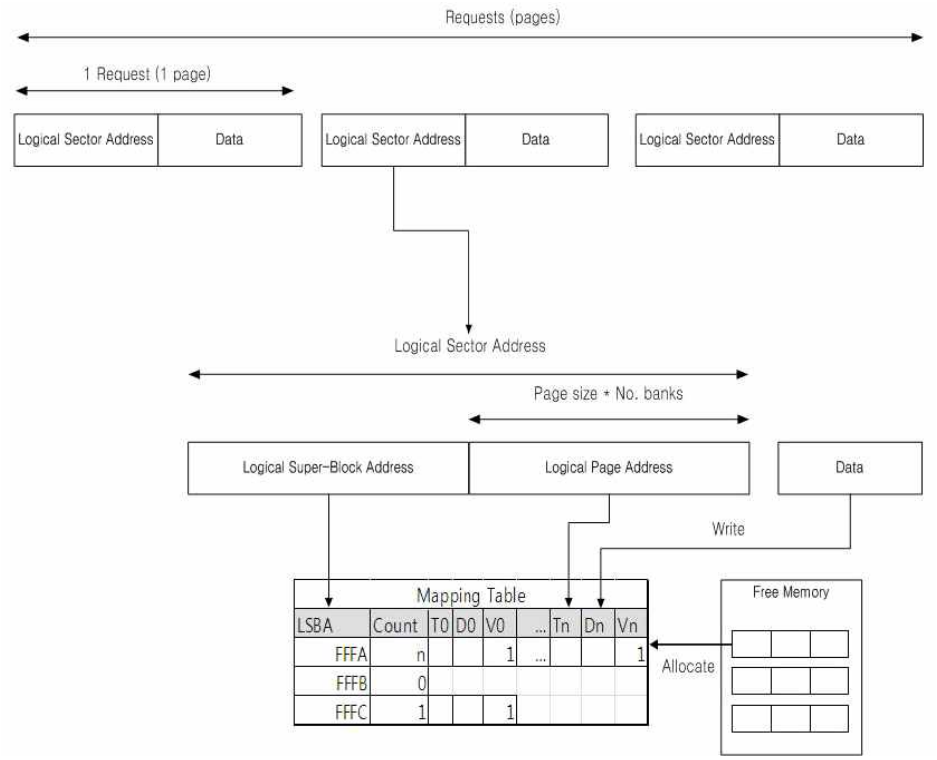
(그림 3)은 호스트로부터 전달된 디스크 참조 요청과 각각의 페이지에 대하여 버퍼에 데이터를 저장 시키는 방법과 과정을 보여준다. 호스트 시스템을 통해서 복수의 쓰기 요청들마다 논리 섹터 주소와 쓰여져야 할 데이터를 갖게 된다. 또한 사상 테이블은 논리 슈퍼 블록 주소마다 이에 연계된 헤더를 갖게 되며 하나의 헤더는 여러 엔트리들을 링크드 리스트 (linked list) 형식으로 취합, 관리하게 된다. 하나의 엔트리는 태그(tag), 데이터(data), 유효비트(valid bit)로 구성된다. 각각의 페이지는 해당 페이지가 사상 되어야 하는 논리 섹터 주소를 갖고 있다. 이 논리 섹터 주소를 하나의 슈퍼 블록이 갖는 섹터 수(페이지당 섹터 수와 뱅크의 개수를 곱한 수)로 나누어 그 나머지는 하나의 슈퍼 블록 내에서의 논리 페이지 주소를 나타내고, 몫은 논리 슈퍼 블록 주소를 나타내게 된다. 논리 슈퍼 블록 주소는 하나의 헤더를 색인 하기 위해 사용되고 논리 페이지 주소는 해당 슈퍼 블록 내에서 태그로 사용 된다. 태그를 비교하고 현재 작성해야 하는 데이터가 새롭게 작성되어야 한다면, 가용 메모리 공간으로부터 하나의 엔트리를 할당 받게 된다.



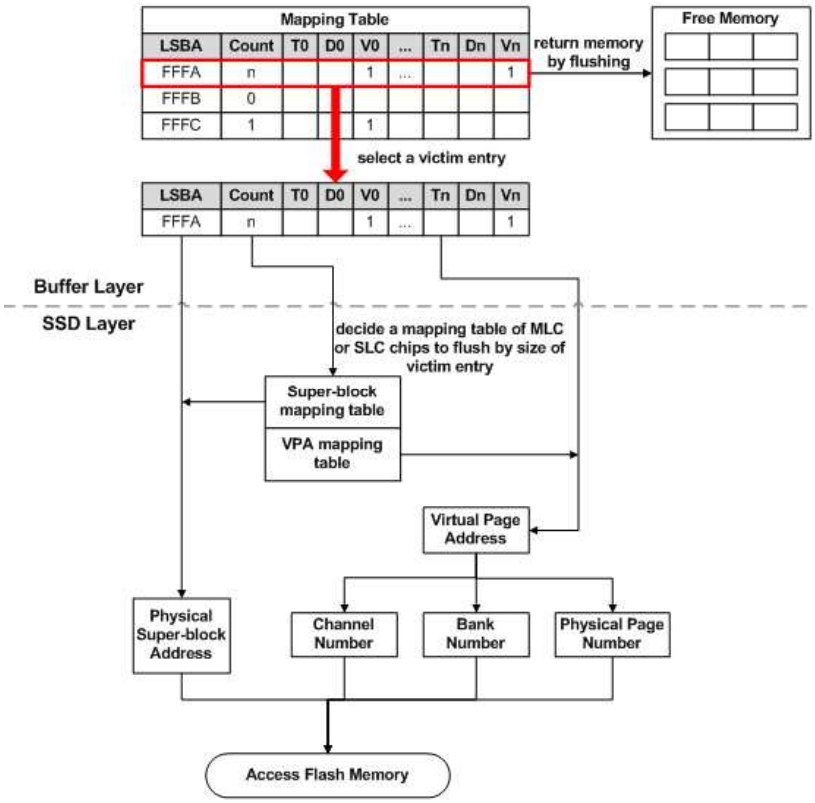
(그림 2) 제안하는 버퍼의 동작 흐름 차트

또한 (그림 3)에서 보여지는 바와 같이 사상 테이블에서 하나의 슈퍼 블록을 선정하고 그 슈퍼 블록이 담고 있는 엔트리들을 플래시 메모리 칩으로 옮겨 쓰는 플러시 동작 과정도 제시되어 진다. (그림 3(가))와 같이 이 구조에서는 LRU (least recently used) 테이블을 통해 LRU 순서와 업데이트 해야 할 페이지수에 의거하여, 버퍼에서 플래시 메모리 칩으로 옮겨 써야 할 슈퍼 블록이 결정된다. (그림 3(나))는 결정된 희생 엔트리 (victim entry)는 해당 논리 슈퍼 블록 주소에 대응되는 물리 슈퍼 블록 주소(Physical Super Block Address : PSBA)로 옮겨 실제적인 데이터 쓰기가 수행된다. 이때 물리 슈퍼 블록 안에 페이지 주소는 각 물리 슈퍼 블록 마다 갖게 되는 가상 페이지 주소 (Virtual Page Address : VPA) 사상 테이블에 의하여 가상 페이지 주소로 결정되게 된다. 이 가상 페이지 주소가 주어진 SSD 구조내의 채널, 뱅크, 물리 페이지를 결정한다. 전환된 물리 슈퍼 블록 주소 및 가상 페이지 주소로 희생 엔트리에 페이지들이 옮겨 쓰여지게 된다.

(그림 4)는 제안하는 버퍼 구조의 사상 테이블을 운영하기 위한 자세한 구동 구조와 데이터 구조를 보여준다. 그림에서와 같이 헤더의 색인은 해쉬 (hash)테이블을 통해서 구현된다. 색인된 헤더는 해당된 논리 슈퍼 블록 주소의 첫 번째 엔트리를 링크드 리스트 포인터를 통해 가리키게 된다. 같은 슈퍼 블록 내에 위치 하고 있는 엔트리들은 링크

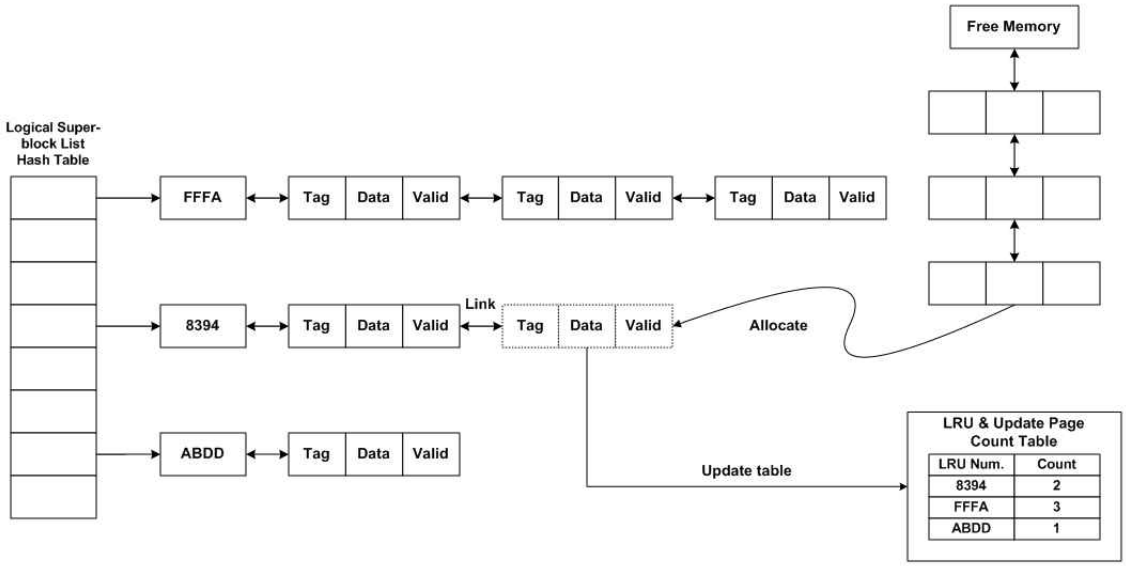


(가) 슈퍼 블록 선정



(나) 논리슈퍼블록주소(LSBA)에서 물리슈퍼블록주소(PSBA)로 변환

(그림 3) 호스트 참조요청과 버퍼 내 페이지 단위 데이터 저장 방법



(그림 4) 버퍼의 사상 테이블을 구성하는 데이터 구조

드 리스트를 통해 연결된다. 자유 메모리 공간은 링크드 리스트를 통해 구현되며 버퍼 내에 비어있는 공간들을 모아 놓은 리스트를 형성한다. 엔트리의 추가 할당이 필요로 하게 되면, 자유 메모리 공간으로부터 엔트리를 할당 받고 해당 리스트의 꼬리에 연결되게 된다. 새로운 페이지가 작성되게 되면 LRU 순서로 접근된 슈퍼 블록 주소가 정렬되며 업데이트 해야 할 페이지 수가 명시되는 테이블이 함께 갱신되게 된다.

5. 성능 평가

본 연구에서 제시하는 동적 사상 테이블을 이용한 제안한 버퍼 구조의 우수성을 검증하기 위한 실험을 수행하였고, 이를 위하여 SSD의 특성을 반영한 시뮬레이터를 설계/구현하고 실험환경에 맞도록 설계 인자들을 설정하였다. 이번 장에서는 설정된 실험환경을 설명하고 이에 따른 실험 결과를 분석한다.

5.1 실험 환경

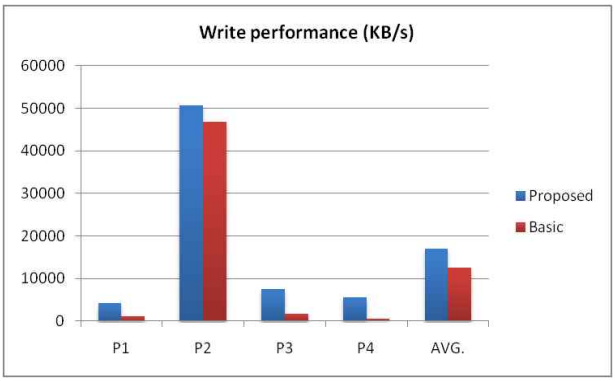
제안한 버퍼의 성능평가를 위하여 시뮬레이터를 구현하고 <표 4>와 같은 사양의 플래시 메모리 칩을 탑재한 SSD를 기준으로 실험을 수행 하였다. 일반적으로 쓰기 동작의 지연시간은 읽기 동작에 비하여 16배 더 소비되며, 쓰기 동작 진행 시 추가적으로 발생 가능한 삭제 동작의 경우에 60배 더 소요되는 것으로 설정하였다. 읽기 동작과 쓰기 동작은 페이지 단위로 수행 가능하지만 삭제 동작은 반드시 블록 단위로 수행되어야 한다. 하나의 블록이 갖는 페이지의 수는 128개 이며 SSD를 구성하는 플래시 메모리 칩의 개수인, 뱅크 개수는 32개로 가정하였다. 그러므로 하나의 슈퍼 블록이 갖게 되는 물리적인 페이지의 개수는 4096 페이지이다. 로그 블록으로 쓰이는 슈퍼 블록은 32개로 한정하여 실험을 진행하였다.

〈표 4〉 실험에 적용된 플래시 메모리 사양

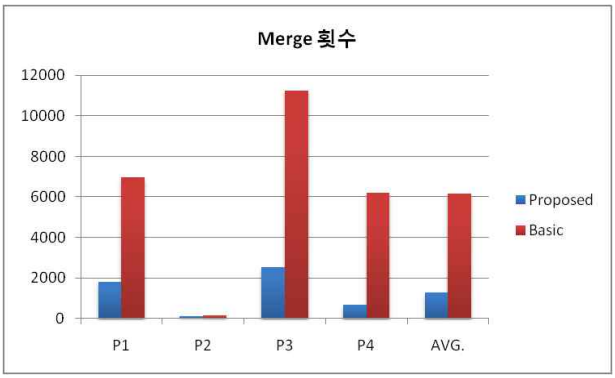
Total Capacity(GB)	32GB
Read Latency(μs)	50
Write Latency(μs)	800
Erase Latency(μs)	3000
No. Banks	32
Page Size(KB)	4
No.pages in a block	128
No.Log - Super block	32

5.2 실험 분석 결과

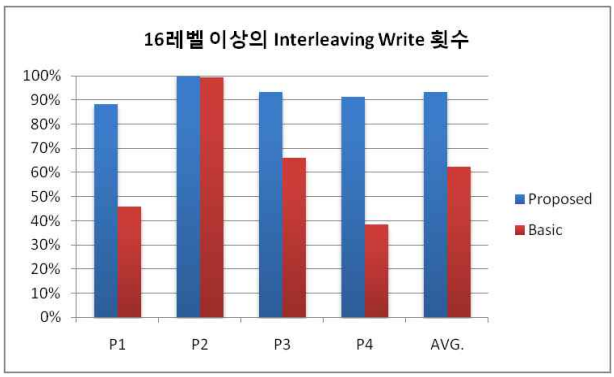
본 실험을 통하여 제시되는 실험결과는 기존에 제시된 SSD 버퍼 구조와의 비교를 통해 분석된다. 기존 버퍼 구조는 FIFO 순서로 데이터를 사상 테이블을 참조하지 않고 쌓아두는 구조로 설계 운영된다[3][4]. (그림 5)는 버퍼 운영의 효과를 분석하기 위하여 제안하는 구조와 기존 버퍼의 쓰기 성능을 보여준다. 벤치마크 P2의 경우, 순차적인 접근 쓰기를 간섭하는 랜덤쓰기의 비율 자체가 적기 때문에 제안하는 구조와 기존의 구조가 쓰기 성능에 있어서 큰 차이를 보이지 않는다. 그 외에 나머지 응용들에서는 모두 제안하는 구조가 기존의 구조에 비하여 큰 성능 향상을 제공함을 보여준다. 이 그림에서 보여준 바와 같이 제시하는 방법은 평균적으로 35% 향상된 쓰기 성능을 보여준다. 또한 제시하는 버퍼 운영기법에서 유발 되는 부수적인 효과를 분석하기 위하여, (그림 6)은 각 응용 별로 발생한 병합 동작의 횟수를



(그림 5) 기존 구조와 제안하는 구조의 쓰기 성능 비교



(그림 6) 기존 구조와 제안하는 구조의 병합(Merge) 동작 발생 횟수



(그림 7) 기존 구조와 제안하는 구조의 Interleaving 효율 비교

보여준다. 병합 동작은 두 개의 슈퍼 블록을 물리적으로 삭제하고 하나의 완전하게 비어있는 슈퍼 블록을 생성함으로써 유효하지 않은 데이터들을 수거하고 빈 공간을 확보하는 작업이다. 이 동작의 발생빈도와 비례하여 삭제동작과 덮어쓰기 동작이 추가적으로 발생하므로 SSD의 수명에 영향을 준다. 그러므로 이러한 병합 동작은 적으면 적을수록 SSD의 전체적인 수명 연장에 도움이 된다. 제안하는 구조는 기존의 구조에 비하여 평균적으로 약 80% 적게 병합 동작을 수행한다.

또한 성능에 주요한 요인 분석을 위하여, 인터리빙의 운영 정도를 분석하였다. 향상된 쓰기 성능과 적어진 병합 동작 횟수는 (그림 7)과 같이 제안하는 구조가 기존의 구조에 비하여 효율적인 인터리빙 쓰기 동작을 보이기 때문에 가능하다. 주어진 뱅크 개수의 절반 이상으로 유발되는 인터리빙 쓰기 동작은 제안하는 구조가 기존의 구조에 비하여 약 30% 더 많이 발생한다. 인터리빙 레벨을 증대하여 쓰기 동작을 수행할수록 여러 페이지를 병렬적으로 여러 칩에 걸쳐서 저장할 수 있기 때문에 쓰기 성능이 향상된다. 이 결과는 제안한 구조가 순차접근의 집중성과 랜덤접근의 간섭에 대한 특징을 반영하였기 때문이다.

6. 결 론

하이브리드 슈퍼 블록 사상 기법을 적용한 SSD는 논리적인 슈퍼 블록단위의 데이터를 물리적인 슈퍼 블록으로 병렬적으로 최대 뱅크 개수만큼 작성 가능하다. 그러나 응용들의 두 가지 특성, 순차접근의 집중성과 임의 접근의 간섭으로 인해 최대 뱅크 개수만큼 병렬적으로 인터리빙 쓰기 동작을 하지 못하는 경우가 발생하게 된다. 이를 해결하기 위해 본 논문에서는 버퍼 내에서 논리 주소 단위로 운영되는 동적 사상 테이블을 구성하고 운영함으로써 응용들의 두 가지 특성을 반영하였다. 그 결과 32개의 뱅크를 가진 SSD에서 16레벨 이상의 인터리빙 쓰기가 제안한 구조가 기존의 구조에 비하여 30% 더 발생하였고 이는 쓰기 성능을 약 35% 향상 시키고 병합 동작을 80% 줄이는 효과를 보였다.

참 고 문 헌

[1] T.S. Chung, D.J. Park, S. Park, D.H. Lee, S.W. Lee, H.J. Song, “A survey of Flash Translation Layer”, Journal of Systems Architecture: the EUROMICRO Journal, Vol.55, pp.332-343, 2009.

[2] J.U. Kang, H.S. Jo, J.S. Kim, J.W. Lee, “A Superblock-based Flash Translation Layer for NAND Flash Memory”, EMSOFT’06, 2006.

[3] L.P Chang, “Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs”, Design Automation Conference, pp.428-433, 2008.

[4] J.S Park, H.K Bahn and K. Koh, “Buffer Cache Management for Combined MLC and SLC Flash Memories Using both Volatile and Nonvolatile RAMs”, IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp.228-235, 2009.

[5] Cagdas Dirik, Bruce Jacob, “The Performance of PC Solid-State Disks (SSDs) as a Function of Bandwidth, Concurrency, Device Architecture, and System Organization”, ISCA’09, 2009.

[6] A. Ban, "Flash file system", United States Patent, no. 5,404,485, April 1995.

[7] S.Y. Kim and S.I. Jung, “A Log-based Flash Translation Layer

for Large NAND flash memory", Advanced Communication Technology, Vol.3, pp.1641-1644, February, 2006.
[8] J.D. Kim, J.M. Kim, S.H. Noh, S.L. Min, Y.H. Cho, "A Space-Efficient Flash Translation Layer for Compact Flash Systems", IEEE Transactions on Consumer Electronics, Vol.48, No.2, May, 2002.
[9] J.H. Yoon, E.H. Nam, H. Kim, B.S. Kim, S.L. Min and Y. Cho, "Chameleon: A High Performance Flash/FRAM Hybrid Solid State Disk Architecture", IEEE Computer Architecture Letters, Vol.8, No.1, January-June, 2008.
[10] J.U. Kang, J.S. Kim, C.I. Park, H.J. Park and J.W. Lee, "A Multi-channel Architecture for High-performance NAND flash-based storage system", Journal of systems architecture: the EUROMICRO Journal, Vol.53, Issue 9, September, 2007.
[11] S.W. Lee, D.J. Park, T.S. Chung, D.H. Lee, S.W. Park, H.J. Song, "A log buffer-based flash translation layer using fully-associative sector translation", Journal of Embedded Computing Systems(TECS), Vol.6, Issue 3, July, 2007.

조 인 표
e-mail : pyotel@supercom.yonsei.ac.kr
2009년 연세대학교 컴퓨터과학과(학사)
2011년 연세대학교 컴퓨터과학과(공학석사)
2011년~현 재 (주)고영테크놀러지 R&D
관심분야 : SSD(Solid State Disk), 임베디드 시스템

고 소 향
e-mail : shko@yonsei.ac.kr
2007년 명지대학교 컴퓨터소프트웨어학과(학사)
2009년 명지대학교 컴퓨터소프트웨어학과(공학석사)
2009년~2010년 (주)디엠에스 정보화팀 사원
2010년~현 재 연세대학교 컴퓨터과학과 박사과정
관심분야 : 차세대 메모리, 파일시스템

양 훈 모
e-mail : hmyang@yonsei.ac.kr
2003년 연세대학교 컴퓨터과학과(학사)
2005년 연세대학교 컴퓨터과학과(공학석사)
2005년~현 재 연세대학교 컴퓨터과학과 박사과정
관심분야 : 메모리 시스템 구조, 성능 분석

박 기 호
e-mail : ghpark@sejong.ac.kr
1993년 연세대학교 전산과학과(학사)
1995년 연세대학교 컴퓨터과학과(공학석사)
2000년 연세대학교 산업시스템공학과 (공학박사)
2002년~2008년 삼성전자 LSI사업부 프로세서구조연구소 선임연구원
2008년~현 재 세종대학교 전자정보공학대학 컴퓨터공학과 조교수
관심분야 : System on Chip(SOC), 임베디드 시스템 설계

김 신 덕
e-mail : sdkim@yonsei.ac.kr
1982년 연세대학교 전자공학과(학사)
1987년 Electrical & ComPuter engineering at University of Oklahoma(공학석사)
1991년 Electrical & Computer engineering at Purdue University(공학박사)
1982년~1986년 삼성전자 컴퓨터 R&D 연구원 전자계산학과 조교수
1987년~1988년 삼성종합연구소 컴퓨터 R&D 주임연구원
1988년~1991년 Purdue Univ. 연구조교
1992년~1993년 삼성종합기술원 정보시스템 연구소 선임연구원
1993년~1994년 광운대학교 공과대학 컴퓨터공학과 조교수
1994년~1996년 연세대학교 이과대학 컴퓨터과학과 조교수
1997년~2002년 연세대학교 공과대학 컴퓨터과학과 부교수
2002년~현 재 연세대학교 공과대학 컴퓨터과학과 교수
관심분야 : 차세대 메모리, 모바일 웹 브라우져, 유비쿼터스 컴퓨팅