특집

Solid State Disk를 위한 주소 매핑 기법

박현철·신동군 (성균관대학교)

I. 서 론

낸드(NAND) 플래시 메모리는 저전력 소모, 빠른 처리 속도, 무소음, 경량이라는 장점을 지녔다. 최근의 급격한 가격 하락에 힘입어 현재 낸드 플래시 메모리는 각종 전자제품에 저장 매체로 탑재되어 사용되고 있다. 또한, 최근에는 SSD (Solid State Disk)라는 형태로 HDD를 대체할 수 있는 대용량 저장매체로써 활용되기 시작하고 있다. 초창기 낸드 플래시 메모리는 한 셀(cell)로 한 비트(bit)를 나타내는 SLC(Single-level Cell) 형태이었지만, 이후 등장한 MLC (Multi-level Cell)는 한 셀로 여러 비트를 표현할 수 있기 때문에 용량 대비 가격이 낮아지고, NAND의 보급에 큰 기여를 하였다. 비록 MLC가 SLC에 비해 느린 처리 속도와 짧은 수명을 지녔지만 집적도가 높기 때문에 가격 측면에서 유리하다.

SSD는 여러 개의 플래시 칩을 병렬로 연결하여 총 용량과 처리 속도를 모두 증가시킨 저장 장치이다. SSD는 HDD에 비해 처리 속도, 전력 소비량, 발생되는 소음, 내구성 등 여러 방면에서 뛰어난 성능을 보여준다. 이런 여러 장점에도 불구하고 아직 SSD의 가격은 HDD에 비해 상대적으로 높기 때문에, 완전히 HDD를 대체하기엔 한계가 있다. SSD의 가격을 낮추기 위해 MLC SSD가 많이 출시되고 있는 데, SLC SSD에 비해 성능이 느리고 수명이 짧다는 단점이 있다.

우수한 성능의 SSD를 구현하기 위해서는 NAND의 특성과 SSD의 구조에 대한 이해가 필요하다. 낸드 플래시는 한 번 기록된 데이터의 갱신이 불가능하며, 읽기 속도보다 쓰기속도가 느리고, 읽기/쓰기 단위는 페이지(Page)인 반면에 지우기 단위는 블록(Block)이다. 이러한 특성을 고려하여 SSD를 제어할 수 있는 소프트웨어의 도움이 필수적인데 이 소프트웨어를 FTL(Flash Translation Layer)라고 한다.

FTL은 호스트(Host)에서 보낸 I/O 요청의 논리 주소를 NAND 상의 물리 주소로 바꿔주는 주소 변환 기능과 빈 공간을 확보하는 폐영역 회수(Garbage Collection), 블록들의 수명을 균등하게 조절하는 마모도 관리(Wear leveling)등의 기능을 가지고 있다. FTL의 주소 변환에서 중요한 점은 변환 단위를 결정하는 문제이다. 만약 지나치게 큰 매핑 단위를 사용한다면 데이터 기록시 오버헤드가 증가하며 SSD의 공간 활용

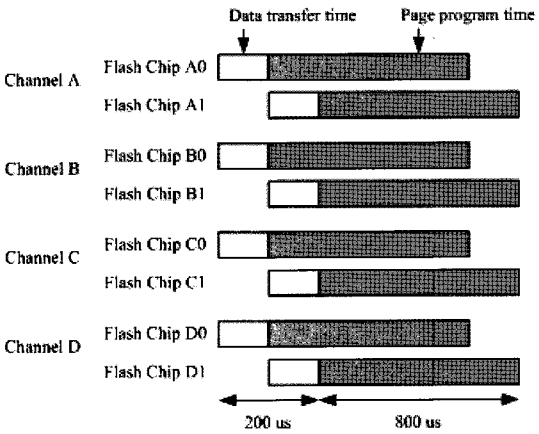
률이 떨어진다. 반면에, 작은 매핑 단위를 사용한다면 처리 속도가 빨라지지만, 보관해야할 매핑 데이터의 양이 커지는 문제가 있다. 그러므로 변환 단위를 주의 깊게 선택해야 하며, 여러 개의 NAND를 병렬로 접근하는 SSD의 구조를 고려한 FTL을 필요로 한다.

본고에서는 SSD의 효율성을 높이기 위한 주소 매핑 기법을 설계하기 위해 고려해야할 사항들을 제시하고, SSD에서 쓰이는 여러 주소 매핑 기법들을 설명하고 비교한다. Ⅱ장에서는 SSD의 구조와 기본적인 주소 매핑 기법에 대해서 설명하고, Ⅲ장에서는 여러 개의 주소 매핑 단위를 사용하는 다중 주소 매핑 기법들에 대해서 설명한다. Ⅳ장에서는 전체 내용을 정리하고 결론짓는다.

Ⅱ. SSD에서의 병렬 처리와 기본적인 주소 매핑 기법

1. SSD의 병렬 처리 구조

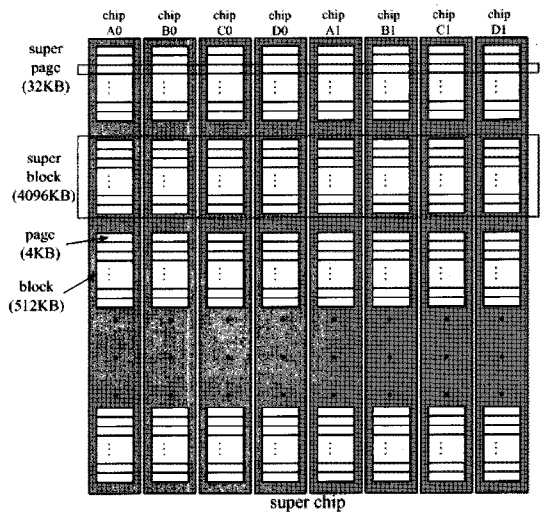
SSD는 여러 개의 NAND들을 병렬 구조로 구성하기 위해 다중 채널과 다중 웨이(Multi-channel & multi-way) 구조를 사용한다. 이 구조에서는 각 채널마다 낸드 플래시 콘트롤러가 있어서 데이터 전송과 연산이 동시에 처리될 수 있다. 그리고 하나의 채널에는 여러 개의 플래시 칩이 연결되어 각 웨이마다 데이터 전송 시간은 겹쳐질 수 없지만 플래시 메모리에 대한 연산 수행 시간은 겹쳐질 수 있다. 그러므로, 4채널 & 2웨이 구조를 SSD에 적용할 경우, <그림 1>과 같이 8개의 칩에 동시에 접근할 수 있으며, 채널들 간의 처리 시간은 완전히 겹쳐질 수 있지만 각 채널들에 2 웨이로 연결된 칩들의 처리시간은 오



〈그림 1〉 4-channel & 2-way의 병렬 처리

로지 페이지 쓰기 시간만 겹쳐지게 된다.

이렇게 동시에 접근될 수 있는 칩들을 모아서 슈퍼칩(superchip)이라고 부를 수 있다. 그리고 각 칩 내에서 같은 주소를 갖는 블록들을 모아서 슈퍼블록(superblock)이라고 하며, 각 칩 내에서 같은 주소의 페이지들을 묶어서 슈퍼페이지(슈퍼블록)라고 부른다. 4채널 & 2웨이 구조에서의 예가 <그림 2>에 나와 있다. <그림 2>에서는 동시에 접근 가능한 8개의 칩을 모아서 슈퍼



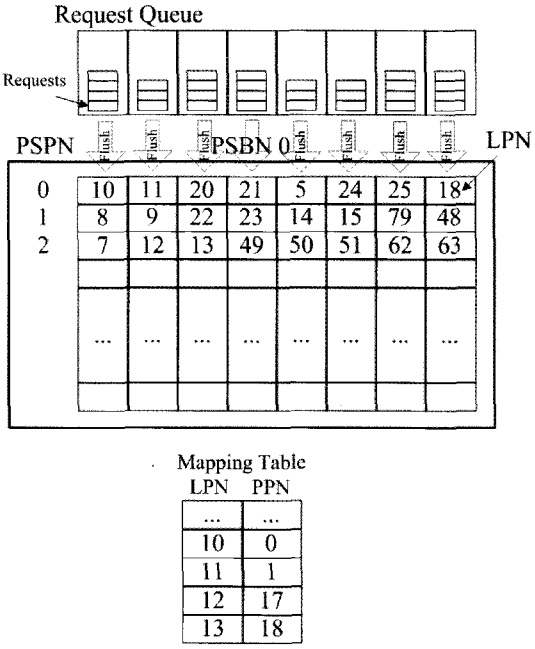
〈그림 2〉 SSD에서 처리 단위들의 구성도

칩이라고 하고, 슈퍼블록은 동시에 지워질 수 있는 8개의 블록으로 구성되어 있으며, 슈퍼페이지는 동시에 읽기/쓰기가 가능한 8개의 페이지로 구성되어있다. 만약 SSD의 FTL이 SSD를 페이지와 블록이 아닌 슈퍼칩과 슈퍼블록, 슈퍼페이지를 이용하여 관리한다면, SSD의 읽기/쓰기 단위는 슈퍼페이지고 지우기 단위는 슈퍼블록이 된다.

이러한 SSD 구조에서 데이터를 기록하는 방식은 크게 두 가지로 구분해 볼 수 있다. 첫 번째는 각 칩마다 연산들이 독립적으로 수행되는 asynchronous 모드이며, 두 번째는 슈퍼페이지 단위로 동일한 명령을 동시에 처리하게 되는 synchronous 모드이다.

2. Asynchronous 모드와 Synchronous 모드

Asynch 모드는 각 칩마다 연산이 독립적으로 수행될 수 있는 상태이다. 호스트로부터 쓰기 요청이 전달되면 FTL은 새로운 데이터를 기록할 칩을 선정하고 해당 칩에서 수행될 명령들이 모여있는 가상의 명령 큐(Request Queue)에 쓰기 명령를 삽입한다. 따라서 각 큐들은 서로 독립적으로 각각의 칩에 처리할 명령을 보내는 것이다. <그림 3>에 async 모드의 예를 보여주고 있다. 8개의 칩에 명령들을 보내는 8개의 큐가 존재하고 각 칩마다 독립적으로 한 명령씩 처리된다. 따라서 논리적으로 연속된 페이지들이라도 동시에 처리되지 않을 수 있으며, 각 칩에서 다른 위치에 기록될 수 있다. 즉, 각 논리 페이지별로 데이터가 기록될 칩 번호, 물리 블록 번호, 물리 페이지 번호가 결정되며 기존의 낸드 플래시 시스템의 페이지 수준 매핑과 동일하다.



〈그림 3〉 Asynchronous 모드와 페이지 매핑

Asynch 모드를 사용한 페이지 매핑은 작업 중이지 않은 아무 칩에나 데이터를 기록하면 되기 때문에 병렬 처리가 최대한 보장된다. 또한, 데이터 갱신시에 이미 기록된 데이터와 새 데이터를 합치는 비용인 병합 비용이 크지 않다는 장점을 가지고 있다.

하지만 페이지 수준의 매핑 정보를 관리하려면 매핑 테이블의 크기가 너무 커져서, SSD 내부의 SRAM에 매핑 정보를 모두 올리기가 힘들다. 따라서 DFTL[2]처럼 전체 매핑 테이블의 일부만을 메모리에 로딩하는 요구 로딩(demand loading)기법이 주로 사용되고 있다.

Async 모드에서는 페이지가 어느 칩에나 기록될 수 있기 때문에 연속된 논리 주소의 데이터가 한 칩에 몰리면 성능이 나빠질 수 있다. 예를 들어, <그림 3>에서 LPN 10부터 LPN 13까지 논리적으로 연속된 페이지 4개는 한꺼번에 읽어올 수 없고 2번에 나눠서 읽어야 한다. 또한 아무

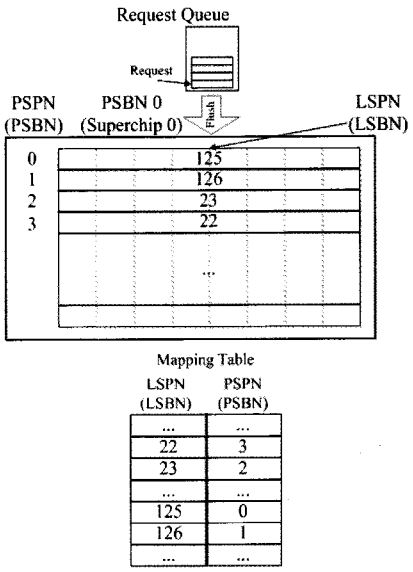
리 큰 데이터라도 페이지 단위로 나눠서 처리해야하고 데이터를 어느 칩에 기록할 것인지 결정해야하기 때문에 FTL의 작업량이 많아져서 오히려 처리 속도가 느려질 수 있다.

Asynch 모드를 위해서 페이지 매핑 외에도 블록 매핑을 사용할 수도 있지만 한 칩에 데이터가 집중되어 오히려 병렬 처리를 방해하므로 좋은 방법은 아니다.

Synch 모드는 논리적으로 연속된 데이터의 처리를 여러 칩들에서 동시에 수행하는 것이다. 칩들에서 동시에 수행될 수 있는 명령들은 같은 명령이어야 하며, 접근되는 칩은 다르지만 칩 안에서의 위치는 동일하다. 그러므로, 동시에 접근될 수 있는 페이지들을 모아서 하나의 단위로 쓸 수 있는데 이 단위가 슈퍼페이지이다. 또한 동시에 지워질 수 있는 블록들을 모아서 슈퍼블록이라고 한다. 그러므로, synch 모드에서는 FTL이 각각의 칩마다 큐를 설정하는 것이 아니라 슈퍼칩에 보낼 명령들이 기다리는 하나의 큐를 사용하게 된다. Synch 모드에서는 RAID-0 기법과 유사하게 순차적인 데이터가 여러 칩들에 분산되어 기록되기 때문에 페이지 매핑처럼 한 칩에 쓰기 요청이 집중되는 경우를 막을 수 있으며 페이지 매핑보다 매핑 정보의 크기가 작다는 장점이 있다.

<그림 4>에 synch 모드가 묘사되어 있다. Synch 모드에서는 최소의 매핑 단위가 슈퍼페이지이다. 그러므로, 슈퍼페이지 단위와 슈퍼블록 단위의 매핑을 고려해 볼 수 있다.

또한, 3장에서 설명할 하이브리드 매핑도 가능하다. 슈퍼페이지 매핑은 논리 슈퍼페이지(LSP) 번호를 물리 슈퍼페이지(PSP) 번호로 변환하며, 슈퍼블록 매핑은 논리 슈퍼블록(LSB) 번호를 물리 슈퍼블록(PSB) 번호로 변환한다.



〈그림 4〉 Synchronous 모드와 슈퍼페이지(슈퍼블록) 매핑

슈퍼페이지 매핑은 슈퍼블록 매핑에 비해서 매핑 단위가 작기 때문에 데이터 기록시에 병합 비용이 작지만, 매핑 테이블의 크기가 상대적으로 크다. 반대로 슈퍼블록 매핑은 매핑 단위가 크기 때문에 매핑 테이블의 크기가 작지만 작은 크기의 데이터가 기록될 때 병합 비용이 매우 크다는 단점이 있다.

이렇듯, 매핑 기법에 따라서 병합 비용에 의한 성능과 매핑 정보의 크기가 다르기 때문에 매핑 기법의 결정은 매우 중요한 문제이다. <표 1>은

〈표 1〉 매핑 수준별 매핑 정보 크기 (128GB SSD)

매핑수준		엔트리 크기	엔트리 갯수	전체 크기
페이지		4bytes	32M	128MB
슈퍼페이지		3bytes	4096K	12MB
슈퍼블록		2bytes	32K	64KB
하이브리드	로그 블록	3bytes	400K	1.2MB
	데이터 블록	2bytes	29K	

128GB SSD에 대해서 각 매핑 기법별 필요한 매핑 정보의 크기를 보여주고 있다. 하이브리드 매핑은 전체 스토리지 용량의 10%가 로그블록이라는 가정으로 계산하였다.

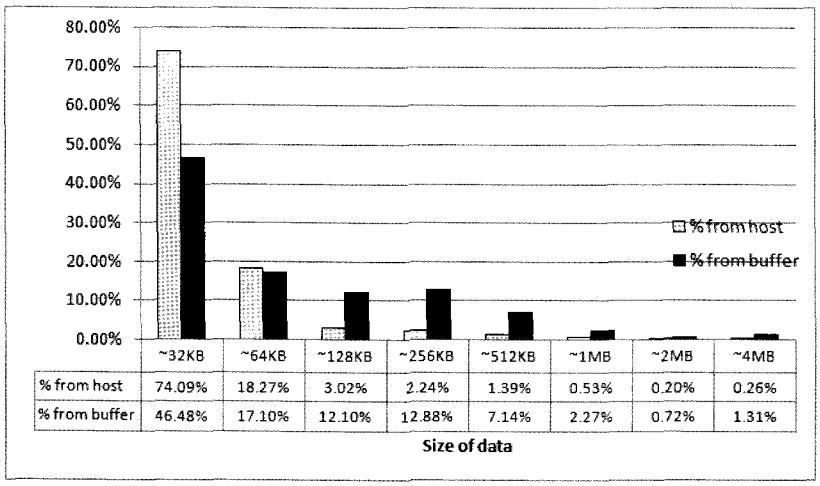
3. SDRAM 버퍼 캐시의 효과

SSD에서 또 한 가지 고려해야 할 사항은 내부의 버퍼 캐쉬의 관리이다. SSD는 일반적으로 버퍼 캐시로 사용될 수 있는 DRAM을 탑재하고 있으며, 호스트에서 전달된 데이터들이 DRAM에서 버퍼링되어 큰 크기로 모여져서 NAND 칩들에 기록되기 때문에 SSD의 낸드 플래시 칩에 실제로 기록되는 데이터는 슈퍼페이지 단위로 처리해도 무방할 만큼 커지게 된다. 그러므로, 페이지보다 큰 슈퍼페이지나 슈퍼블록 단위를 사용하는 것이 효율적이다. <그림 5>는 호스트에서 전달된 데이터들의 크기별 분포와 32MB 버퍼를 거친 후 플래시 칩들에 기록되는 데이터들의 크기별 분포를 비교하였다.

회색 막대는 NTFS 파일시스템을 사용하는 호스트에서 SSD로 내려온 데이터의 크기별 분포이고 검은색 막대는 버퍼에서 플래시 칩으로 내려오는 데이터의 크기별 분포이다. 가로축은 데이터들을 크기별로 8개 구간으로 나타낸 것이며 첫 번째 구간은 0KB ~ 32KB, 두 번째 구간은 33KB ~ 64KB으로 구간의 넓이는 두 배씩 증가하여 최대 4MB까지 포함한다. <그림 5>에서 보는 것처럼 64KB 이하의 데이터들의 비율은 버퍼를 거치면서 줄어들었고 이보다 큰 데이터들의 비율은 오히려 증가하였다. 결론적으로 플래시 칩에 기록되는 대부분의 데이터들이 페이지보다 크기 때문에 페이지보다 더 큰 단위의 주소 매핑 단위를 사용하여도 데이터 기록시 병합 비용이 크게 증가하지 않을 것임을 알 수 있다.

III. 다중 수준 주소 매핑 기법들

페이지 매핑, 슈퍼페이지 매핑, 슈퍼블록 매핑은 모두 한 종류의 매핑 단위를 쓰는 기법들이다. 하나의 매핑 단위를 쓰면 FTL은 단순해지지만,



	~32KB	~64KB	~128KB	~256KB	~512KB	~1MB	~2MB	~4MB
% from host	74.09%	18.27%	3.02%	2.24%	1.39%	0.53%	0.20%	0.26%
% from buffer	46.48%	17.10%	12.10%	12.88%	7.14%	2.27%	0.72%	1.31%

〈그림 5〉 버퍼링 전후의 데이터 크기별 비율

다양한 크기의 데이터 접근이 발생할 경우에 매핑의 효율성이 떨어지게 된다. 여기서 매핑의 효율성이란 데이터의 크기와 매핑 단위의 크기가 유사하도록 하여 병합 비용을 줄이고 매핑 테이블 관리 비용도 적정한 수준을 유지하는 것을 말한다.

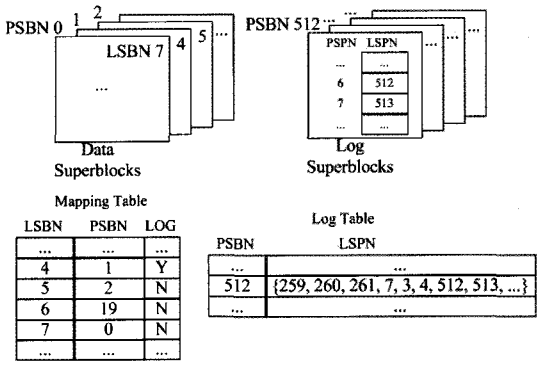
따라서 다양한 크기의 데이터 접근을 효율적으로 처리하기 위해서 2개 이상의 매핑 단위를 쓸 수 있다. 이번 장에서는 로그 버퍼(Log buffer)를 사용해 2개의 매핑 단위를 제공하는 하이브리드 매핑과 로그 버퍼 없이 다수개의 매핑 단위를 지원하여 데이터에 가장 적합한 매핑 단위를 선택해서 사용하는 다중 수준 주소 매핑 기법을 설명한다.

1. 하이브리드 매핑

하이브리드 매핑으로 다양한 기법들이 연구되었다. 그 중 대표적인 기법이 BAST[3], FAST[4]이다. 이 기법들은 매핑 단위로 페이지와 블록 두 가지를 지원한다. 그리고 블록들을 데이터 블록과 로그 블록들로 나눠서 관리하게 되는데 데이터 블록에서 쓰이는 매핑 단위는 블록이고, 로그 블록에서 쓰이는 매핑 단위는 페이지이다. 로그 블록들은 로그 버퍼로 쓰이게 되는 데, 로그 버퍼는 데이터 블록에서 갱신되는 데이터가 페이지 단위로 우선 기록되는 장소이다. 즉, 로그 버퍼는 데이터 블록에서 발생하는 병합 비용을 줄이기 위해서 데이터 블록에 기록하기 전에 데이터가 임시적으로 머무는 버퍼라고 할 수 있다. BAST에서 한 개의 로그 블록에는 오직 한 개의 데이터 블록의 새 데이터만 기록할 수 있다. 반면 FAST는 한 개의 로그 블록에 모든 데이터 블록의 새 데이터를 기록할 수 있다. FAST가 로그 블록들을 더 효율적으로 활용하는 방법이므로 본 장에서는 FAST에 대해서 설명하도록 하겠다.

FAST를 SSD에 적용한다면 FAST는 슈퍼페이지와 슈퍼 블록의 매핑 단위를 지원하고, 슈퍼블록들을 데이터 슈퍼블록과 로그 슈퍼블록으로 나눠서 관리하는 주소 매핑 기법이라고 할 수 있다. SSD에 FAST가 적용되어 데이터가 기록된 예가 <그림 6>에 나타나있다.

<그림 6>에서 슈퍼블록 당 슈퍼페이지 개수가 128개이라고 하자. 만약 논리 슈퍼페이지 번호(LSPN) 513번의 데이터를 읽어야 한다면 이 데이터가 속한 논리 슈퍼블록 번호(LSBN)는 513 / 128 = 4이므로 매핑 테이블에서 LSBN 4를 찾아간다. LSBN 4는 물리 슈퍼블록(PSBN) 1에 매핑되어 있다. 그러나 LOG 표시가 Y로 되어있어 해당 블록에 대한 로그 블록이 할당되어 있다는 것을 알 수 있고, 이 데이터의 최신의 데이터가 로그 버퍼에 있을 수 있다. 따라서 로그 버퍼를 우선 확인해서 로그 버퍼에 있으면 로그 슈퍼블록에서 읽어오고 없을 경우에는 데이터 슈퍼블록에서 읽어온다. 최신 데이터의 위치를 얻기 위해 로그 테이블로 가서 LSPN 513이 있는 PSBN을 찾는다. PSBN 512의 8번째 칸에



〈그림 6〉 SSD에 FAST가 적용된 예

LSPN 513이 기록되어 있으므로 LSBN 513은 PSBN 512의 물리 슈퍼페이지 번호(PSPN) 7에 기록되어있다.

FAST는 데이터를 일단 슈퍼페이지 단위로 기록하기 때문에 병합 비용이 크지 않으며, 매핑 테이블들의 크기도 슈퍼페이지 매핑보다 작다. FAST의 단점은 데이터 블록의 매핑 테이블과 로그 테이블 둘 다 확인해야 하기 때문에 주소 변환에 시간이 걸릴 수 있다. 그리고 로그 버퍼의 크기를 충분히 할당하지 않으면 로그 버퍼의 효과가 줄어들 수 있다. 게다가 SSD에서 로그 버퍼는 그 활용도가 떨어질 수 있는 데, 그 이유는 SSD 내부에는 이미 DRAM 버퍼 캐시가 존재하기 때문에 데이터들이 어느 정도 모아지고 갱신된 상태로 낸드 칩들에 기록되기 때문이다. 만약 버퍼 캐시에서 내보내지는 데이터들이 충분히 갱신되어 이후 장기간 갱신되지 않는다면 로그 버퍼의 사용은 오히려 해가 될 수 있다. 갱신이 거의 일어나지 않을 데이터이기 때문에 데이터 슈퍼블록에 곧장 기록해도 되는데, 로그 버퍼를 거치기 때문에 불필요하게 두 번 기록하는 것이 되기 때문이다.

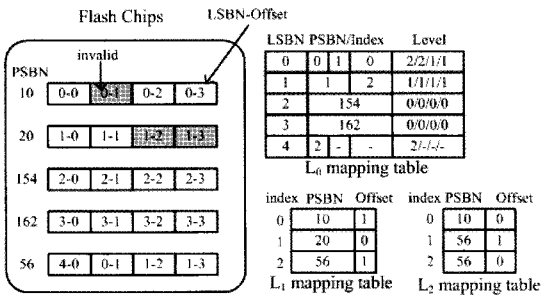
그러므로 FAST는 버퍼 캐시가 있더라도 NAND 칩들에서 데이터 갱신이 활발히 일어나고 기록되는 데이터의 크기가 충분히 크지 않을 경우에 효과적이다.

2. 다중 수준 매핑

MLAM(Multi-level Address Mapping)이라 불리는 다중 수준 매핑 기법은 다수개의 매핑 단위를 지원하여 낸드 칩에 기록되는 데이터에 가장 적당한 크기의 매핑 단위로 데이터를 매핑하는 방법이다. 데이터의 크기와 비슷한 매핑 단위를 사용하므로 로그 버퍼 없이도 병합 비용을 줄일 수 있다. MLAM은 사용 환경의 데이터 접근 패턴에 따라서 적절한 매핑 단위들을 결정한다. 이때 선택할 수 있는 최소 단위는 슈퍼페이지이고 최대 단위는 슈퍼블록이다. 그리고 선택된 매핑 단위들의 크기는 서로 2^x배 또는 $2^{1/y}$배 (x, y는 정수) 가 되도록 한다. 이렇게 하여 매핑 테이블 관리 비용을 줄인다.

<그림 7>은 슈퍼블록, 슈퍼블록의 1/2, 슈퍼블록 1/4의 총 3개의 매핑 단위를 지원하는 MLAM의 예이다. 따라서 최대 매핑 단위는 슈퍼블록이고 최소 매핑 단위는 슈퍼블록의 1/4 단위이고, 각각의 단위를 L0, L1, L2라고 부른다.

처음에 LSBN 0, 1, 2, 3이 PSBN 10, 20, 154, 162에 각각 기록되어 있었다고 가정하자. 호스트로부터 LSBN 0과 1의 일부분을 업데이트하고 또한 새롭게 LSBN 4의 일부를 기록하는 쓰기 요청이 전달되었을 때, MLAM 기법은 갱신되는 논리 슈퍼블록을 4조각으로 분리하여 관리한다. 그러므로, PSBN 56에 LSBN 4, 0, 1의 일부분을 모아서 그림과 같이 기록한다. 그리고, 매핑 레벨을 수정하게 되는 데, LSBN 0의 첫 번째와 두 번째 1/4 슈퍼블록과 LSBN 4의 첫 번째 1/4 슈퍼블록은 L2 매핑을 사용하여 기록하며, LSBN 0의 세 번째와 네 번째 1/4 슈퍼블록과



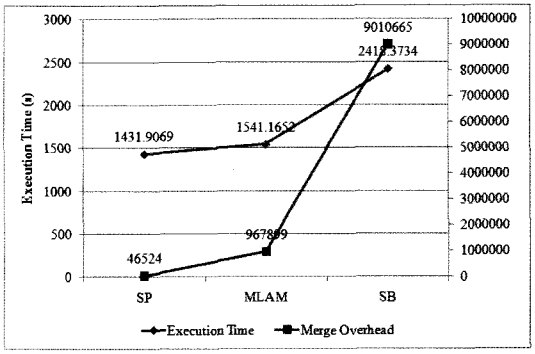
〈그림 7〉 3가지 매핑 수준을 지원하는 MLAM의 예

LSBN 1의 모든 1/4 슈퍼블록은 1/2 슈퍼블록 단위의 L1 매핑을 이용하여 기록한다.

이러한 매핑 수준 정보는 L0 매핑 테이블의 level 필드에 기록된다. 만약 L0로 매핑되어 있으면 L0 매핑 테이블의 PSBN/Index 필드는 PSBN을 가리키게 되고 L1이나 L2로 매핑되어 있으면 PSBN/Index필드는 해당 매핑 레벨 테이블의 Index를 가리키게 된다. 예를 들어, LSBN 0의 첫 번째 1/4 데이터를 접근하려고 하면 L0 테이블의 LSBN 0인 행에서 level 필드의 해당 offset 값이 2이므로 L2로 매핑되어 있음을 의미한다. L0 테이블의 PSBN/Index 필드의 첫 번째 칸의 값인 0은 L2 테이블의 Index 0을 의미한다. L2 테이블에서 Index 0의 (PSBN, Offset) = (10, 0)이므로 물리 슈퍼블록 10번의 0번 부분에 (LSBN, Offset) = (0, 0)의 데이터가 기록되어 있다는 의미이다.

<그림 7>의 플래시 칩에 (PSBN, Offset) = (10, 1)은 무효화(invalid) 상태이다. 이것은 LSBN 0이 L0로 PSBN 10에 매핑되어 있다가 (LSBN, Offset) = (0, 1)이 갱신되면서 (PSBN, Offset) = (56, 1)로 옮겨진 후 무효화되었기 때문이다. 이와 같이, MLAM에서는 슈퍼블록의 특정 영역만 갱신되어도 해당 영역의 매핑 정보만 바뀌게 되므로 슈퍼블록 매핑에 비해서 병합 비용을 줄일 수 있으며, 슈퍼페이지 매핑에 비해서 매핑 테이블의 크기를 줄일 수 있다. 즉, MLAM 기법은 데이터의 입력 패턴에 따라서 매핑 수준을 가변적으로 결정함으로서 적은 매핑 정보로도 좋은 성능을 제공할 수 있는 것이다.

<그림 8>은 슈퍼페이지 매핑과 MLAM, 슈퍼블록 매핑의 총 수행 시간과 병합비용을 비교한 것이다. 이 실험은 ext3파일 시스템을 쓰는 리눅스 환경에서 수행되었고, iozone의 -a 옵션 실행 시 I/O 요청들을 blktrace로 수집하여 트레이스를 만들었다. 이 트레이스를 32MB DRAM을 탑재한 4채널 2웨이 32GB SSD를 모방하는 시뮬레이터로 실행시킨 것이다. 따라서 이 시뮬레이터는 SSD의 병렬처리와 DRAM 버퍼 캐시의 동작을 시뮬레이션할 수 있다. 총 수행 시간은 트레이스의 모든 명령들을 처리하는 데 걸린 시간이고 갱신되지 않았지만 매핑 단위를 유지하기 위해 복사된 페이지의 개수로 병합 비용을 측정하였다.



〈그림 8〉 주소 매핑 기법 별 성능 비교

MLAM에서 지원되는 매핑 단위들은 32KB (슈퍼페이지)부터 4MB(슈퍼블록)까지 총 8개의 매핑 단위를 지원한다. 이렇게 다양한 매핑 단위를 지원함으로서 iozone의 -a 옵션 실행시 발생하는 다양한 크기의 쓰기 요청을 효과적으로 처리할 수 있다.

<그림 8>에의 왼쪽 세로축은 총 처리 시간을 초 단위로 나타내고 오른쪽 세로축은 병합 비용인 복사된 페이지의 개수를 나타내며, 하단의 가로축은 각각 슈퍼페이지 매핑, MLAM, 슈퍼블록 매핑을 가리킨다. 슈퍼블록 매핑의 경우 상대적으로 긴 처리시간과 많은 페이지 복사가 발생했지만, MLAM은 슈퍼페이지 매핑과 근사한 결과를 보여준다. 만약 슈퍼페이지 매핑에서 매핑 테

이블을 NAND 칩들에 저장해두고 접근되는 일부만 SRAM에 올리도록 하였다면 매핑 정보의 읽고 쓰는 비용 때문에 슈퍼페이지 매핑의 실험 결과는 MLAM의 결과와 더 비슷해지거나 더 나빠질 수도 있다. 병합 비용의 차이에 비해 총 처리 시간의 차이는 상대적으로 적은데 이는 SSD의 병렬 처리로 인하여 여러 페이지들이 동시에 이동할 수 있기 때문이다.

Ⅳ. 결 론

SSD는 빠른 처리속도와 저전력, 안정성 같은 여러 장점이 있지만, 여전히 높은 가격으로 인해 MLC 플래시 메모리를 사용한 제품이 많이 개발되고 있다. SSD의 비용을 절감하고 고성능을 제공하기 위해서는 SSD에 내장된 FTL이 호스트로부터의 I/O 요청을 효과적으로 처리할 수 있어야 한다. FTL의 중요한 기능인 주소 변환은 논리 주소를 물리 주소로 바꿔주는 것인데, 여기에 쓰이는 매핑 기법에는 페이지 매핑, 슈퍼페이지 매핑, 슈퍼 블록 매핑, 하이브리드 매핑 등을 고려해 볼 수 있다. 본고에서는 각 매핑 기법들의 장단점을 분석하고 다중 수준 주소 매핑(MLAM)이라는 새로운 기법도 소개하였다. 페이지 매핑은 유연하게 쓰기 명령들을 처리할 수 있지만 매핑 테이블의 크기가 크고, 한 칩에 연속된 데이터가 기록되게 되면 오히려 성능이 나빠질 수 있다. 슈퍼페이지 매핑과 슈퍼블록 매핑은 SSD의 병렬 처리 구조를 반영하지만 매핑 단위를 하나만 지원하기 때문에 매핑 단위의 크기에 따른 장점과 단점이 명확하게 구분되어 드러난다. 하이브리드 매핑은 슈퍼페이지와 슈퍼블록 두 개의 매핑 단위를 지원하는 데 슈퍼블록들의 일부를 로그 버퍼로 두기 때문에 SDRAM 버퍼 캐시의 크기가 작거나 버퍼 캐시의 효과가 작은 환경에 적합하다. MLAM은 로그 버퍼를 사용하지 않고 데이터의 크기에 맞는 매핑 단위 사용함으로서 적은 크기의 매핑 정보로도 좋은 성능을 제공할 수 있다. 결론적으로 SSD의 주소 매핑 기법을 설계할 때엔 SSD의 병렬처리와 버퍼 캐시의 효과, 그리고 워크로드의 특성을 고려해서 설계해야 효율적인 기법을 설계해야 한다.

참고문헌

[1] T.S. Chung, D.J. Park, S. Park, D.H. Lee, S.W. Lee, and H.J. Song, "A survey of Flash Translation Layer," Journal of System Architecture, Vol.55, No.5-6, 2009.

[2] Aayush Gupta, Youngjae Kim, Bhuvan Urgaonkar, "DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings," ASPLOS '09, Washington, DC, USA, March, 2009.

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

[4] S.W. Lee, W.K. Choi, and D.J. Park, "FAST : An efficient flash translation layer for flash memory," EUC Workshops 2006, pp.879-887, 2006.

저자소개

박 현 철

2009년 2월 성균관대학교 컴퓨터공학과 학사

주관심 분야 : 플래시 메모리, SSD, 컴퓨터 구조

신 동 군

1994년 2월 서울대학교 계산통계학과 학사
2000년 2월 서울대학교 전산과학과 석사
2004년 8월 서울대학교 컴퓨터공학부 박사
1994년 1월~1998년 2월 KCC 정보통신
2004년 11월~2007년 2월 삼성전자 책임연구원
2007년 3월~현재 성균관대학교 정보통신공학부 조교수

주관심 분야 : 플래시 메모리, 임베디드 시스템, 컴퓨터 구조