다중 섹터 사이즈를 지원하는 낸드 플래시 메모리 기반의 저장장치를 위한 효율적인 FTL 매핑 관리 기법

(Efficient FTL Mapping Management for Multiple Sector Size-based Storage Systems with NAND Flash Memory)

임 승 호 † 최 민 ††
(Seung-Ho Lim) (Min Choi)

요 약 컴퓨터 시스템에서 Host와 저장장치간의 데이터 이동은 섹터를 기본 단위로 하고 있는데, 섹터 사이즈는 시스템마다 다른 가변적인 크기일 수 있다. 낸드 플래시 메모리는 구조상 페이지 사이즈와 섹터 사이즈 사이의 상관관계에 있어서, 섹터 사이즈가 낸드 플래시 메모리를 관리하는 방식에 상당한 영향을 미친다. 본 논문에서는 낸드 플래시 메모리 기반의 저장장치에서 효율적인 다중 섹터 사이즈를 지원하는 FTL 매핑 관리 기법을 제안하고, 그 관리 방법과 성능에 관하여 분석하여 본다. 본 논문에서 제안한 방식에 의하면 다중 섹터를 지원하는 낸드 플래시 메모리 저장장치를 효율적으로 관리하여 줄 수 있다.

키워드 : 낸드 플래시 메모리, 다중 섹터 사이즈, FTL

Abstract Data transfer between host system and storage device is based on the data unit called sector, which can be varied depending on computer systems. If NAND flash memory is used as a storage device, the variant sector size can affect storage system performance since its operation is much related to sector size and page size. In this paper, we propose an efficient FTL mapping management scheme to support multiple sector size within one NAND flash memory based storage device, and analyze the performance effect and management overhead. According to the proposed scheme, the management overhead of proposed FTL management is lower than conventional scheme when various sector sizes are configured in computer systems, while performance is less degraded in comparison with single sector size support system.

Key words : NAND Flash Memory, Multiple Sector Size, FTL

· 이 연구는 2010학년도 한국외국어대학교 교내학술연구비의 지원에 의하여 이루어진 것임
· 이 논문은 2010 한국컴퓨터종합학술대회에서 '다중 섹터 사이즈를 지원하는 낸드 플래시 메모리 저장장치의 FTL 매핑 관리 기법'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 한국외국어대학교 디지털정보공학과 교수
slim@hufs.ac.kr
†† 정 회 원 : 원광대학교 컴퓨터공학과 교수
mchoi@wonkwang.ac.kr

논문접수 : 2010년 8월 10일
심사완료 : 2010년 10월 27일

1. 서 론

컴퓨터에서 호스트와 저장장치 간의 데이터 전송 단위는 섹터를 기반으로 하게 된다. 일반적으로 섹터 사이즈는 일정한 크기로 인터페이스 표준에서 정의되어 왔다. 그런데 요즈음에는 이러한 섹터가 컴퓨터 시스템의 데이터 안정성을 더욱 더 보장하기 위해서 일반 유저 데이터와 데이터 보호를 위한 ECC 데이터를 포함하여 (Real Data + ECC)로 구성된다. 호스트에서 ECC를 포함하여 사이즈가 늘어난 섹터는 저장장치에서는 (Real Data + ECC)를 사이즈가 늘어난 섹터 데이터로 간주하게 된다. 그런데, ECC의 데이터 길이는 데이터 안정성에 따라서 호스트에서 가변적으로 설정해 주기 때문에 섹터 사이즈 역시 컴퓨터 시스템의 설정에 따라 다르게 설정해야 할 필요가 있다[1]. 이러한 다중 섹터 사이즈의 지원은 아주 데이터의 신뢰성을 보장해 주어야 하는 서버 시스템에서 주로 사용된다.

최근 컴퓨터의 저장장치는 낸드 플래시 메모리의 출현 이후로 낸드 플래시 메모리를 기반으로 하는 SSD (Solid State Device)와 같은 대용량의 저장장치로도 많이 연구 및 개발이 진행되고 있는 상황이다[2,3]. SSD는 HDD가 지원하는 수준의 인터페이스 프로토콜 표준을 준수하도록 연구 및 개발 되고 있다. 낸드 플래시 메모리는 데이터를 읽고 쓰는 단위가 페이지(page)라고 하는 단위로 정의되어 있으며, SSD 내부에서 낸드 플래시 메모리를 관리하는 기본 단위는 페이지 단위가 된다. 낸드 플래시 메모리는 특성상 플래시 변환 계층(Flash Translation Layer, FTL)이라고 하는 플래시 소프트웨어 계층을 이용하여 관리가 된다[4-6]. FTL은 플래시 논리 주소를 물리주소로 변환시켜주는 매핑 관리를 통해서 저장장치로서 사용될 수 있도록 호스트 시스템에 인

터페이스를 제공하는 소프트웨어 계층이라고 할 수 있다. 이러한 플래시 변환계층에 대한 성능 최적화 및 관리 최적화에 대한 연구는 많이 진행되어 왔다[7,8].

그런데, FTL의 매핑 단위 및 동작 단위는 낸드 플래시 메모리의 페이지를 기본으로 하지만, 호스트 시스템의 전송 단위인 섹터와 아주 밀접한 상관관계가 있다. 만약, 섹터 사이즈와 페이지 사이즈의 상관관계가 정수 비례단위가 되지 않은 경우에는 FTL 매핑 관리 및 동작 관리에 많은 오버헤드가 수반된다. 즉, 다중 섹터 사이즈를 지원하도록 FTL의 매핑 관리 및 동작관리가 이루어지지 않으면, 다중 섹터를 요구하는 컴퓨터 시스템의 성능은 좋을 수가 없다. 본 논문에서는 낸드 플래시 메모리 기반의 저장장치에서 효율적인 다중 섹터 사이즈를 지원하는 FTL 매핑 관리 기법을 제안하고, 그 관리 방법과 성능에 관하여 분석하여 본다.

2. 연구 배경

낸드 플래시 메모리의 기본 동작방식은 읽기, 쓰기, 지우기 연산으로 구성된다. 읽기와 쓰기는 페이지(Page)라고 하는 낸드 플래시 메모리 단위로 이루어지며, 지우기는 블록(Block)이라고 하는 단위로 이루어진다. 일반적으로 페이지의 크기는 2KB, 4KB, 8KB정도의 크기를 가지며, 한 블록은 64개 또는 128개의 페이지로 구성되어 128KB, 256KB, 512KB정도의 크기이다. 일반적으로 읽기, 쓰기, 지우기 연산의 수행시간은 각각, 15us, 200us, 2ms정도이다. 낸드 플래시 메모리는 한번 쓴 영역에 대해서 다시 데이터 쓰기 동작을 하기 위해서는 쓰기 이전에 지우기 연산을 수행해 주어야 한다. 즉, 쓰기 연산은 항상 지워져 있는 프리 영역에 이루어진다. 프리 영역이 없을 경우에는, 기존에 데이터가 쓰여진 영역 중 한 블록을 축출하여, 유효한 데이터만 복사를 한 후, 해당 블록을 지워줌으로써 프리 영역을 확보해야 한다. 이러한 과정을 가비지 콜렉션(GC, Garbage Collection)이라고 한다.

FTL은 이러한 낸드 플래시 연산의 오버헤드를 최소화하기 위해서 제안된 소프트웨어 계층이다. 즉, FTL은 호스트 시스템의 논리 주소를 변환하여 플래시 메모리의 물리주소로 변환시켜주는 매핑 관리를 해준다. 이러한 매핑 관리 및 매핑 변환을 통해서 호스트로부터 요청된 쓰기 연산에 대해서 낸드 플래시 메모리의 새로운 영역에 데이터를 쓰고, 이를 논리 주소와 물리주소로 연결시켜주는 작업을 한다. 또한, FTL은 프리 영역을 만들어주기 위한 GC작업도 수행한다.

FTL의 매핑 단위는 낸드 플래시 메모리 구조의 두 가지 기본단위인 블록 또는 페이지 단위로 구성할 수 있으며, 혹은 그 두 단위를 혼합하여 매핑 관리를 할 수 있다[4,5]. FTL의 매핑 단위를 블록 단위로 하면, 매핑 테이블의 사이즈를 줄일 수 있지만, 매핑 단위의 유연성을 감소시키므로 FTL의 동작방식에 오버헤드를 부가한다. 반면, FTL의 매핑 단위를 페이지 단위로 하면, 매핑 테이블 사이즈가 커지지만, 매핑 단위의 유연성이 증가하므로 FTL의 성능이 좋아진다.

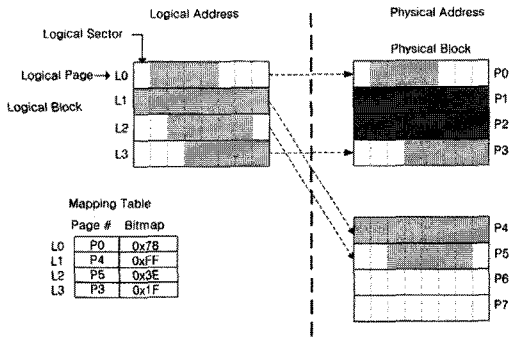

그림 1 페이지 단위의 FTL 매핑 관리

본 논문에서는 FTL의 매핑 단위를 페이지단위를 이용한 FTL을 기본 구조로 한다. 예를 들어, FTL에서 관리하는 낸드 플래시 메모리의 페이지 사이즈는 4KB이며, 블록 사이즈는 256KB, 그리고 호스트 시스템의 섹터 사이즈는 512B인 경우에 대한 페이지 매핑 단위의 FTL에 대한 동작방식은 그림 1과 같다. 그림 1에서와 같이 하나의 페이지는 8개의 섹터로 구성되며, 매핑 테이블은 페이지 단위로 구성이 된다. 그리고 섹터의 구분은 한 페이지 내에서 섹터 비트맵으로 표현하여 유효한 데이터를 관리해 준다. 만약, 기존에 저장된 하나의 논리 페이지 내에서 일부의 섹터가 변경이 되면, 기존의 논리 페이지가 저장된 물리 페이지로부터 데이터를 DRAM 영역으로 읽어 들여, 변경된 섹터의 데이터를 업데이트해 준 후, 새로운 물리 페이지에 데이터를 저장한다. 그런 후 페이지 매핑 테이블에, 해당 논리 페이지에 대한 물리 페이지 주소를 업데이트해 준다. 위와 같은 FTL의 매핑 단위 에서 페이지 내부의 일부 섹터의 변경 또는 섹터 데이터 쓰기 연산을 위해서는 한 섹터가 서로 다른 페이지에 걸쳐 있는 경우가 없기 때문에 기본적으로 한 페이지 읽기와 쓰기만으로 가능하다.

그러나, 만약 섹터사이즈의 배수가 페이지 사이즈로 나누어 떨어지지 않는다면 오버헤드가 많아지게 된다. 이러한 FTL의 오버헤드를 최소화 하면서 다중 섹터 사이즈를 지원하기 위한 FTL의 구조는 기존의 구조와는 다른 방식의 매핑 관리가 필요하다. 섹터 사이즈가 520B인 저장장치에 대한 경우를 살펴보자. 이 경우, 아주 쉽게 두 가지 방식의 FTL을 생각할 수 있다. 첫 번째는, 그림 2에 나온 경우로서, 페이지 사이즈가 4KB인

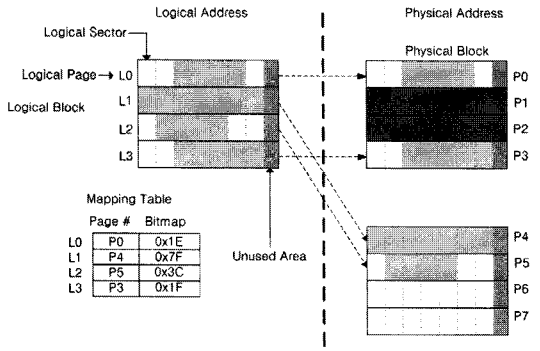

그림 2 분리 섹터 방식: 다중 섹터 사이즈를 지원하는 Naïve FTL

Naïve FTL

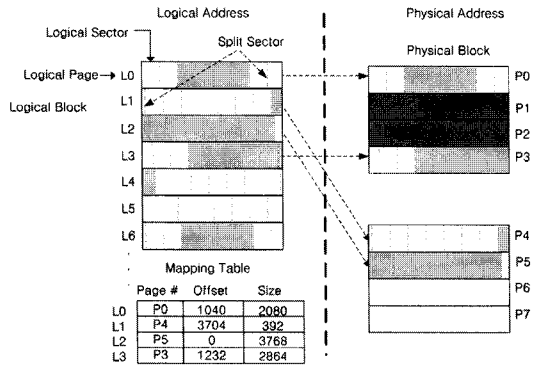

그림 3 연속 할당 방식: 다중 섹터 사이즈를 지원하는 Naive FTL

낸드 플래시 메모리에 7개의 섹터를 저장하도록 구성하며, 나머지 456B(=4KB-520X7)는 데이터를 저장하지 않는 방식이다. 이렇게 되면, 한 페이지에 7 섹터를 저장하며, 이를 섹터 비트맵으로 관리함으로써 섹터가 두 페이지에 연속적으로 걸쳐져 있는 영역이 존재 하지 않는다. 이는 기존의 낸드 플래시 메모리 연산과 같은 단순 복잡도의 연산 오버헤드가 존재하나, 한 페이지 내에서 7섹터 이외의 잉여영역을 사용하지 않음으로써 스토리지 낭비 오버헤드가 존재한다. 두 번째는 그림 3에서 나와있는 경우로써, 페이지의 양 끝 부분에 하나의 섹터를 분리(Split)하여 나누어 저장하는 경우이다. 이런 경우에 스토리지 낭비 없이 사용할 수 있지만, 매핑 관리가 아주 복잡해지며 분리 섹터(Split Sector)에 대한 낸드 플래시 메모리 연산에 대한 오버헤드가 발생하여 성능이 현저히 떨어지게 된다.

3. 다중 섹터 지원 FTL

본 논문에서는 위의 두 경우에서 발생하는 오버헤드

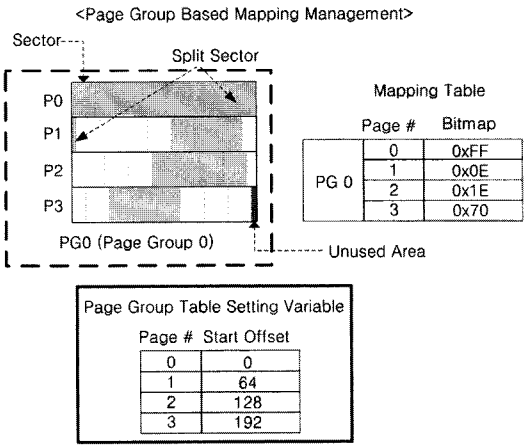

그림 4 제안된 페이지 그룹 방식의 다중 섹터 지원 FTL

인 스토리지 낭비와 낸드 플래시 메모리 연산 오버헤드를 없앨 수 있는 다중 섹터 사이즈 지원 FTL 매핑 관리 방법에 대해서 제안한다. 본 논문에서 제안하는 방식은 그림 4에 나와 있는 방식으로 FTL을 관리 한다. 즉, 본 논문에서 제안하는 방식은 페이지 그룹(Page Group, PG) 기반의 FTL 매핑 관리 방법이다. 그림 4에서 보는 바와 같이 페이지 그룹 기반의 FTL 매핑 관리 방법은 먼저, 페이지 사이즈에 맞아 떨어지지 않아서 나누어지는 분리 섹터의 수를 최소화하기 위해서 페이지를 그룹(PG)으로 묶는다. 그 후, PG의 첫 페이지부터 섹터를 할당해 나간다. 만약 하나의 섹터가 현재 페이지에 완전히 저장되지 않는 경우에는, 그 나머지 부분을 다음 페이지에 연속적으로 저장하며, 이 때 분리 섹터가 발생하게 된다. 이런 방식으로 계속 섹터를 할당하면, 한 페이지에 저장되는 완전한 섹터의 개수(N)은 (1)과 같이 나타낼 수 있다. 수식에서 P는 페이지 사이즈이며, S는 섹터 사이즈이다.

$$N = P / S \tag{1}$$

즉, PG내의 마지막 페이지를 제외한 모든 페이지는 한 페이지 내에서 분리되지 않고 저장되는 완전한 섹터의 개수가 N개가 유지되는 페이지의 수로 그룹은 생성된다. 한 PG의 모든 페이지는 분리 섹터를 가지게 되는데, 각 페이지의 분리 섹터의 오프셋(O_Pn)은 모든 PG에서 동일한 값을 가지며 (2)와 같이 계산되며, PG내의 페이지 수(PN)은 (3)과 같이 계산된다.

$$O\_P_n = \begin{cases} 0, & (n=0) \\ O\_P_{n-1} + ((N+1)S - P), & (0 < n < PN) \end{cases} \tag{2}$$

$$PN = O\_P_n + NS > P\text{인 } n \tag{3}$$

이와 같은 구성에 의하면 페이지 그룹 내에서, 맨 마지막 페이지를 제외한 페이지들은 매 페이지당 섹터의

개수는 완전한 섹터N개와 분리 섹터 하나, 즉, (N+1)의 섹터를 가지게 된다. 마지막 페이지는 N개의 완전한 섹터로만 구성되며 그 이후의 영역은 사용하지 않는 영역이다. 또한, 페이지 그룹 내의 모든 페이지에 대해서 매 페이지의 시작 섹터 오프셋은 식 (2)와 같이 계산하여 Page Group Table을 생성해 낼 수 있다. 즉, FTL 매핑 테이블을 구성할 때, 낸드 플래시 메모리의 페이지 사이즈가 정해지고 호스트 시스템에서 사용할 섹터 사이즈가 정해지면, 위의 수식에 의해서 페이지 그룹(PG)을 생성할 수 있으며, 페이지 그룹 내에서 각 페이지의 섹터 비트맵을 구성할 수 있다. 그림 4에서 예를 들은 바와 같이, PG0는 네 개의 페이지로 구성되며, 각 페이지당 섹터 개수는 P0, P1, P2는 8개이며, P3는 7개의 섹터로 구성된다. 그리고 각 페이지당 자신에게 포함된 섹터의 시작 섹터 오프셋(O_Pn)은 Page Group Table과 같이 구성된다.

이렇게 구성된 페이지 그룹을 이용하여 보다 효율적인 낸드 플래시 메모리 연산을 수행할 수가 있게 된다. 즉, 한 페이지 그룹 내에서 페이지 당 섹터 비트맵을 일정하게 구성할 수 있으며, 각 섹터당 시작 섹터 오프셋은 Page Group Table을 통해서 바로 참조할 수 있기 때문에, 해당 섹터의 위치를 찾는 오버헤드가 들지 않게 되어, 기존의 섹터 비트맵 연산과 유사한 연산 오버헤드를 수반한다. 그러나, 본 논문에서 제안하는 방식 역시 섹터 분리에 의한 오버헤드를 가진다. 분리 섹터는 낸드 플래시 메모리의 두 페이지에 연속해서 걸쳐져 있기 때문에, 하나의 두 페이지에 대한 낸드 플래시 메모리 연산이 수반되어야 한다. 이러한 오버헤드를 보충해 주기 위해서는 캐시를 잘 활용해야 한다.

4. 결과 및 분석

페이지 그룹 기반의 FTL 매핑 관리 기법에 대한 성능 분석을 하도록 한다. 그림 5는 페이지 사이즈 4KB, 섹터 사이즈 520B와 설정에 대해서, 본 논문에서 제안한 다중 섹터를 관리하는 FTL 매핑 관리 방법의 Sequential Read, Sequential Write, Random Read, Random Write인 네 가지 성능과 Capacity Impact에 대해서 분석 및 정리한 결과이다. Sequential Read/Write의 경우, 128KB(즉, 256 섹터)를 가정 하였으며, Random Read/Write의 경우, 2KB(즉, 4 섹터)를 가정하였다. 본 논문의 제안 방식에 대해서 앞에서 설명한 두 가지 Naïve한 다중 섹터 지원 방식과 기존의 기본적인 페이지 매핑 기반의 FTL과 비교한다.

먼저 Sequential Read의 경우, 분리섹터 방식은 7섹터마다 섹터가 분리되어 있기 때문에 그만큼 불필요한 연산이 늘어나게 된다. 이에 반해, 연속할당 방식과 페이지그룹 방식은 기존의 페이지 매핑 기반의 FTL 관리 방법에 비해서 거의 성능 차이가 없다. Sequential의 경우 페이지 크기보다 훨씬 큰 데이터를 낸드 플래시 메모리 읽기 연산을 통해서 읽기 때문에 분리 섹터에 의한 부가적인 낸드 플래시 메모리 연산이 거의 발생하지 않는다. Sequential Write 역시 마찬가지로 분리 섹터에 의한 플래시 메모리 연산 오버헤드가 발생하지 않는다고 볼 수 있다. 다만, 오버헤드가 되는 부분은 PG의 마지막 부분의 사용하지 않는 영역으로 인한 오버헤드가 있을 수 있는데, 이것은 전체 영역에서 차지하는 부분이 작기 때문에 크게 문제가 되지 않는다.

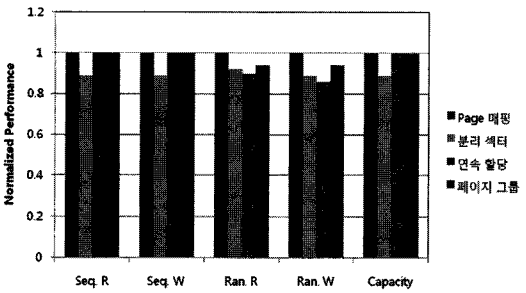

그림 5 다중섹터 지원 FTL에 대한 성능 분석 결과

랜덤 성능의 경우, Sequential과는 다르게 분리 섹터에 의한 영향을 많이 받기 때문에 성능의 영향이 있다. 다중 섹터를 지원하지 않는 FTL의 경우, 2KB의 Random Read에 대해서 5/11의 확률로 낸드 플래시 메모리 4KB 한 페이지만 읽는 연산과, 6/11의 확률로 낸드 플래시 메모리 2페이지를 읽는 연산으로 완성될 수 있다. 그러나, 다중섹터 지원을 위한 섹터 분리 방식과 페이지 그룹 방식의 FTL의 경우, 분리 섹터에 의해서 4/11의 확률로 1페이지 읽기 연산, 7/11의 확률로 2페이지 읽기 연산이 수반되므로, 낸드 플래시 메모리 연산의 오버헤드는 6%의 낸드 플래시 읽기 연산을 발생시킨다. 연속할당 방식의 경우, 매 Read에 대해서 낸드 플래시 메모리의 실제 Offset을 계산해 주어야 하기 때문에 많은 CPU 오버헤드가 더 발생한다.

Random Write의 경우는, 일반적으로 기존의 데이터가 저장된 페이지에 대해서 읽기를 한 후, 업데이트 되는 부분을 캐시 상에서 데이터를 갱신한 후, 낸드 플래시 메모리 쓰기를 수행하게 된다. 그러므로, 페이지 매핑 기반의 FTL의 경우, 평균적으로 2KB의 Random Write에 대해서 1.55번의 읽기와 쓰기연산으로 Random Write를 완성한다. 다중 섹터를 지원하기 위한 섹터 분리 방식과 본 논문에서 제시한 페이지 그룹 기반의 FTL 매핑은 1.64번의 읽기와 쓰기 연산으로 Random Write을 완성해야 하므로 12%의 낸드 플래시 메모리 연산

오버헤드가 발생한다. 연속할당 방식은 Random Read와 마찬가지로 Offset 계산에 의한 CPU 오버헤드가 추가된다.

스토리지 Capacity Impact의 경우는, 섹터 분리 방식이 매 7섹터마다 사용하지 않는 영역이 발생하므로 전체 Capacity의 11% 영역을 사용하지 못하는 오버헤드가 발생한다. 연속 할당 방식은 섹터를 연속적으로 할당하므로 Capacity 오버헤드는 발생하지 않지만, 모든 섹터의 Offset이 다르기 때문에 이것을 계산해주기 위한 CPU 오버헤드가 발생한다. 이에 반해, 본 논문에서 제시한 페이지 그룹 기반의 FTL 매핑은 페이지 그룹의 마지막 일부분만을 사용하지 않기 때문에 0.02%의 오버헤드만 발생하며, 모든 그룹 내에서 섹터의 Offset은 Table로 정해지기 때문에 Offset 계산에 의한 CPU 오버헤드도 발생하지 않는다.

5. 결 론

컴퓨터 시스템에서 호스트와 저장장치간의 데이터 전송 단위인 섹터 사이즈가 컴퓨터 시스템의 데이터 안정성을 더욱 더 보장하기 위해서 컴퓨터 시스템의 설정에 따라 다르게 설정해야 할 필요가 있다. 본 논문은 낸드 플래시 메모리 기반의 저장장치를 사용할 경우, 다중 섹터 사이즈를 효율적으로 지원할 수 있는 페이지 그룹 기반의 FTL 매핑 관리 방법을 제안하였다. 이를 이용하여, FTL의 매핑 관리 및 동작 방식에 오버헤드를 최소화함과 동시에 다중 섹터를 지원해 줌으로써 보다 안정성 높은 낸드 플래시 메모리 기반의 저장장치의 구현이 가능하다.

참 고 문 헌

[1] EMC Corporation, "EMC Symmetrix VMAX and VMware Virtual Infrastructure," White Paper, May. 2010.

[2] Samsung Corporation, "K9XXG08XXM Flash Memory Specification," 2009.

[3] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J.A. Tauber, "Storage Alternatives for Mobile Computers," In *Proceeding of First Symposium of Operating Systems Design and Implementation*, vol.1, pp.25-37, 1994.

[4] J. Kim, M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems," *IEEE Transactions on Consumer Electronics*, vol.48, no.2, pp.366-375, May. 2002.

[5] C. Park, W. Cheon, J. Kang, K. Roh, W. Cho, and J.-S. Kim, "A reconfigurable FTL(flash translation layer) architecture for NAND flash-based applications," *ACM Transactions on Embedded Computing Systems*, vol.7, pp.1-23, 2008.

[6] Intel Corporation, "Understanding the Flash Translation Layer (FTL) Specification," White Paper, 1998.

[7] Li-Pin Chang, Tei-Wei Kuo, "An efficient management scheme for large-scale flash-memory storage systems," In *Proceedings of the ACM symposium on Applied computing*, vol.1, pp.862-868, Mar. 2004.

[8] S.H. Lim and K.H. Park, "An efficient NAND flash file system for flash memory storage," *IEEE Transactions on Computers*, vol.55, no.7, pp.906-912, Jul. 2006.