논문 14-39B-07-01
The Journal of Korea Information and Communications Society '14-07 Vol.39B No.07
http://dx.doi.org/10.7840/kics.2014.39B.7.425

멀티채널과 멀티웨이 구조의 NAND 플래시 SSD를 위한 효율적인 웨어레벨링 알고리듬

김 동 호•, 황 선 영°

An Efficient Wear-Leveling Algorithm for NAND Flash SSD with Multi-Channel and Multi-Way Architecture

Dong-ho Kim•, Sun-young Hwang°

요 약

본 논문은 멀티채널과 멀티웨이 구조를 가진 SSD의 구조적 특성을 감안한 웨어레벨링 알고리듬을 제안한다. 제안된 알고리듬은 쓰기 요청이 도착했을 때 DRAM에 저장된 데이터를 논리주소 접근 빈도에 따라 핫 데이터와 콜드 데이터로 나누고, 블록 소거횟수의 편차를 줄이도록 데이터를 할당한다. 콜드 데이터를 소거횟수가 많은 블록에 할당하여 소거횟수 증가를 억제한다. 멀티채널과 멀티웨이 구조의 SSD 시뮬레이터에 다양한 어플리케이션에서 얻어진 트레이스를 적용하여 검증한 결과, 기존의 웨어레벨링 알고리듬을 사용하는 경우에 비해 블록의 소거횟수의 차이가 평균 9.3% 줄어들고 총 소거횟수가 평균 4.6% 감소하였다.

Key Words : Nand flash memory, SSD, Wear-leveling, Multi-channel, Multi-way

ABSTRACT

This paper proposes a wear-leveling algorithm that exploits the properties of SSD memories with multi-channel and multi-way architecture. When a write request arrives, the proposed algorithm classifies the stored data in DRAM buffer into hot or cold according to logical address access frequency, and performs data allocation to reduce deviation of block erase counts. It lowers the chance of increasing erase count by allocating cold data to blocks which have high erase count. Effectiveness of the proposed algorithm is verified by executing various applications on a multi-channel, multi-way SSD simulator. Experimental results show that differences in erase count among blocks is reduced by an average of 9.3%, and total erase count decreases by 4.6%, when compared to previous wear-leveling algorithm.

Ⅰ. 서 론

프로세서의 성능이 점차 향상됨에 따라 메모리에 요구되는 사양도 높아진다. 디스크 기반 저장매체는 연속한 주소 접근에 좋은 성능을 보이나 임의 주소에 접근할 때 성능이 저하된다. 플래시 메모리는 연속한 주소 접근과 임의 주소 접근 속도가 모두 빠르고 멀티레벨 셀 기술의 도입으로 단가를 낮춰 디스크 기반 저장매체를 대체하여 사용이 증가하고 있다[1]. 플래시 메모리는 낮은 전력 소모, 비휘발성, 경량화, 작은 크

※ 본 연구는 삼성전자(주)의 지원으로 수행 하였습니다(No. 201270038.03).
• First Author : Sogang University Department of Electric Engineering, dogdh@sogang.ac.kr, 학생회원
° Corresponding Author : Sogang University Department of Electric Engineering, hwang@sogang.ac.kr, 종신회원
논문번호 : KICS2014-05-195, Received May 26, 2014; Revised June 12, 2014; Accepted June 13, 2014

기의 장점을 가지고 있어, 기존의 디스크 기반 주 저장장치를 대체하는 플래시 메모리 기반의 주 저장장치의 형태로 SSD가 제안되었다[2-8]. 플래시 메모리를 주 저장장치로 사용하기 위해 고려할 특성들이 존재한다. 현재 사용되는 파일 시스템은 디스크 기반 저장매체를 기준으로 구성되었다. 섹터 단위로 쓰기가 가능한 디스크와 달리 플래시 메모리는 페이지 단위로 쓰기가 가능하다. 덮어 쓰기가 가능한 디스크와 달리 플래시 메모리는 다른 값을 기록하기 전에 페이지를 지우는 동작이 앞서야 한다. 소거동작은 페이지 단위로 지울 수 없어 블록 단위로 수행되며, FTL에 의해 쓰기와 지우기 단위의 불일치의 문제가 해결된다[2,3]. 그림 1은 플래시 메모리를 주 저장장치로 사용하는 파일 시스템 구조를 나타낸다[4]. FTL 영역은 논리 주소를 플래시 메모리의 실제 주소로 매핑하는 기능 외에, 플래시 메모리의 마모를 늦추는 기능을 한다[9,10]. 반영구적으로 사용 가능한 디스크 기반 저장매체와 달리 플래시 메모리는 쓰고 지울 때의 전기적 부하에 의해 마모된다[11]. 웨어레벨링은 블록이 소거되는 횟수를 플래시 메모리 전반적으로 균등하게 분산시켜 블록이 지나치게 빨리 마모되는 것을 방지한다.

SSD에는 신속한 데이터 처리를 위해 복수의 컨트롤러와 칩들이 탑재된다[12,13]. 칩과 컨트롤러를 기준으로 채널이 나뉘어 병렬적으로 데이터가 처리되며, 하나의 컨트롤러로 복수의 칩에 접근하기 위해 멀티웨이 구조를 사용한다. 주소 접근 시간을 줄이기 위해 SSD가 멀티웨이구조에 멀티플레인 어드레싱을 적용하는 경우, 소거동작의 단위가 블록에서 슈퍼블록으로 바뀌므로 웨어레벨링 알고리듬도 이에 맞춰 바뀌어야 한다. 본 논문에서는 멀티채널, 멀티웨이 구조의 특성에 맞는 웨어레벨링 기법을 제안한다. 본 논문에서 제안하는 방법은 웨어레벨링을 수행할 때 구조적 특성을 고려하여 성능을 향상시킨다. 2장에서는 구조적 특성, 문제점, 그리고 관련 연구를 제시한다. 3장에서는 제안한 방법을 설명하고, 4장에서는 결과 및 분석을 보인다. 5장은 결론을 내리고 향후 연구 방향을 제시한다.

Ⅱ. 연구 배경

본 절에서는 멀티채널과 멀티웨이 구조가 적용된 SSD에 대해 설명하고 기존 웨어레벨링 연구를 소개한다.

2.1 멀티채널과 멀티웨이 SSD 구조

낸드 플래시 버스의 데이터 전송속도는 I/O 인터페이스의 데이터 전송속도에 비해 낮은 속도를 보인다. SSD는 전송속도의 불일치 문제를 해결하기 위해 멀티채널 구조를 사용하고, 채널마다 별도의 플래시 컨트롤러를 할당하여 병렬 처리한다[12-14]. 그림 2는 4-채널 2-웨이 SSD의 구조를 나타낸다[13]. SSD 컨트롤러는 호스트 인터페이스 로직을 통해 호스트의 읽기/쓰기 요청을 전달받아 데이터를 처리하며, DRAM 버퍼는 쓰기 요청 버퍼링을 위해 사용된다.

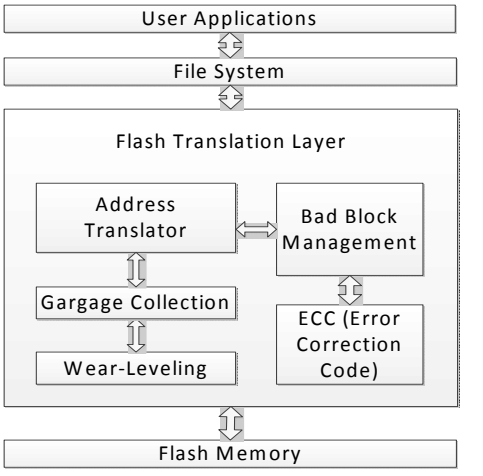

그림 1. 플래시 메모리 기반의 파일 시스템 구조.
Fig. 1. Flash memory-based file system architecture.

각 채널별로 할당된 플래시 컨트롤러는 SSD 컨트롤러로부터 읽기/쓰기 요청을 전달 받아 데이터를 메모리에 저장하거나, 메모리로부터 데이터를 읽어 SSD 컨트롤러에 전송한다. 하나의 플래시 컨트롤러로 복수의 플레인을 제어하기 위해 플레인 별로 읽기/쓰기 요청을 순차적으로 전달하는 인터리빙 방식이 제안되었다[13]. 인터리빙 방식은 한 채널에 존재하는 플레인에

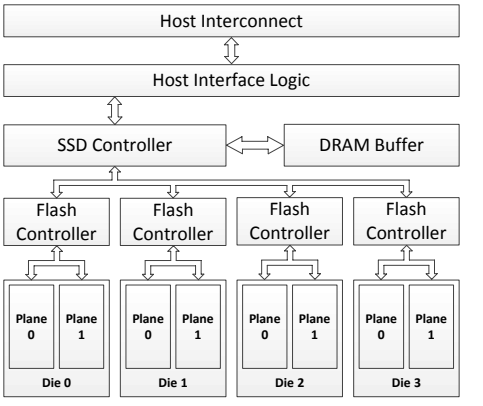

그림 2. 4-채널과 2-웨이 SSD 구조.
Fig. 2. A 4-channel and 2-way SSD architecture.

파이프라이닝을 적용하여 요청을 순차 처리하여, 동시에 복수의 플레인에 접근하지 못하는 단점이 있다. 멀티 플레인 어드레싱 방식은 인터리빙 방식의 단점을 해소하기 위해 제안되었으며, 하나의 컨트롤러로 둘 이상의 플레인을 제어한다[15]. 그림 2는 그림에서 다이0에 위치한 플레인 0과 플레인 1이 컨트롤러를 공유함을 보인다. 각각의 플레인은 워드 라인을 공유하여 하나의 컨트롤러로 두 플레인에 동시에 접근 가능하다.

슈퍼블록은 멀티 플레인 어드레싱의 장점을 살리기 위한 데이터 처리 단위로써, 워드 라인을 공유하는 블록들로 구성되며 지우기 동작의 기본 단위가 된다. 그림 3은 실제 슈퍼블록 구성의 예를 보인다. 다이 0에서 두 플레인에 위치한 블록들은 양옆으로 인접한 블록끼리 워드 라인이 연결되어 있어, 두 블록이 하나의 슈퍼블록을 구성하여 멀티 플레인 어드레싱 방식을 사용한다. 슈퍼블록 방식을 사용하여 플래시 메모리 접근 속도를 높일 수 있으나, 기존에는 블록 단위로 수행이 가능한 소거동작 단위와 오동작할 때 폐기하는 단위가 슈퍼블록으로 고정되는 단점이 있다.

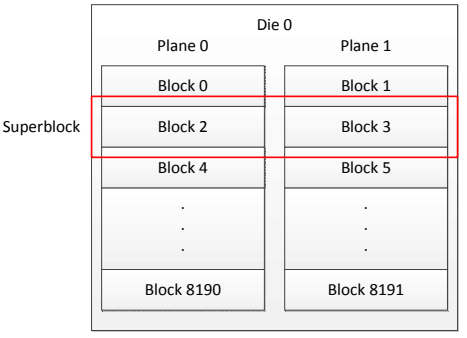

그림 3. 슈퍼블록 형성의 예.
Fig. 3. Example of superblock clustering.

2.2 웨어레벨링

웨어레벨링은 플래시 메모리의 수명을 극대화하기 위해 블록들이 수명을 다하는 시점을 일치하게 만들기 위한 기법으로, 블록의 소거횟수를 균등하게 컨트롤하기 위한 것이다. 전체 블록의 총 소거횟수가 같을 때, 블록의 소거횟수가 균등한 경우는 소거횟수가 편중된 경우와 비교하여 수명이 다한 블록의 수가 적다. 기존의 웨어레벨링 기법은 소거횟수가 많은 블록의 소거동작을 줄이기 위해 연구되었다. 가비지 콜렉션 단계에서 웨어레벨링을 수행하는 알고리듬은 블록의 소거횟수와 블록 내 유효 페이지 수를 기준으로 가비지 콜렉션을 수행한다[16]. 이 방식은 소거동작의 효율성을 높이고 블록 소거횟수를 평준화하나, 희생 블록을 선택하는 과정에서 발생하는 오버헤드가 크다. 쓰기 동작이 완료된 후 블록들의 소거횟수를 확인하여 조건에 따라 소거횟수의 차이가 줄어들도록 블록에 저장된 데이터를 교환하는 방식이 제안되었으나, 데이터의 접근 패턴이 변화하는 경우 성능이 떨어지고 데이터 교환에 추가적인 오버헤드가 발생한다[17].

기존의 연구는 플래시 메모리 구조에 무관하게 범용적으로 적용 가능한 웨어레벨링 기법에 대해 이뤄졌다. 멀티채널과 멀티웨이 구조를 가진 SSD는 소거동작이 슈퍼블록 단위로 수행되므로, 기존 방식을 바꾼 환경에 대한 추가적인 고려 없이 적용할 경우 동작 단위의 차이로 인해 웨어레벨링이 비효율적으로 수행될 가능성이 있다. 보다 정확하고 효과적인 웨어레벨링을 위해 구조적 특성을 반영할 필요가 있다.

Ⅲ. 제안하는 알고리듬

본 절에서는 제안된 멀티채널, 멀티웨이 구조에 적용하기 위한 웨어레벨링 기법에 대해 설명한다. 제안된 웨어레벨링 방식은 데이터 분류 단계와 데이터 할당 단계를 통해 블록들 사이의 소거횟수 차이를 감소시킨다. 데이터 분류는 DRAM 버퍼에서 수행되며 해당 데이터의 시간적 지역성과 공간적 지역성 특성을 파악하고 이용하기 위한 처리 과정이다. 쓰기 요청이 도착할 때마다 버퍼 내부에 인접한 논리 주소의 쓰기 요청이 존재하는지 확인하여 해당 쓰기 요청의 공간적 지역성을 파악한다. 인접한 논리 주소는 연속적으로 접근될 가능성이 인접하지 않은 논리 주소에 비해 높으므로, 공간적 지역성 특성을 이용하기 위해 플래시 메모리에 기록할 경우 같은 블록에 기록할 수 있게 같은 쓰기 요청으로 병합한다. 버퍼에 존재하는 논리 주소의 쓰기 요청이 도착한 경우 해당 데이터를 갱신하고, 쓰기 요청과 함께 저장되는 링크드 리스트에 접근된 순서를 반영하여 시간적 지역성을 파악한다. 플래시 메모리와 달리 DRAM은 덮어 쓰는 동작이 가능하여 데이터가 갱신되는 경우 무효 데이터 발생 없이 예전 데이터를 변경된 데이터로 덮어 쓴다. DRAM 버퍼 안에서 갱신된 데이터는 태그를 기록하여 데이터 할당 단계의 참고 자료로 삼는다.

쓰기 요청이 발생할 때마다 SSD 컨트롤러는 도착한 쓰기 요청이 DRAM 버퍼에 저장된 데이터의 논리 주소에 대한 것인지 확인한다. 도착한 쓰기 요청의 논리 주소와 DRAM 버퍼에 존재하는 논리 주소가 일치하지 않으면 대기열 상단에 도착한 쓰기 요청을 추가

시킨다. 데이터의 공간적 지역성 특성을 이용하기 위해, 쓰기 요청을 추가하는 과정에서 대기열에 인접한 주소의 쓰기 요청이 존재하는 경우 추가될 쓰기 요청을 해당 쓰기 요청에 병합시킨다. 일치하는 논리 주소가 존재하는 경우 예전 요청을 갱신하고 해당 주소가 포함된 저장된 쓰기 요청을 대기열 상단으로 이동한다. 그림 4는 제안된 알고리듬에 따라 DRAM 버퍼에 쓰기 요청이 저장되는 과정을 보인다. 그림 4 (a)에서 SSD 컨트롤러에 도착한 쓰기 요청 4의 논리 주소 29는 DRAM 버퍼에 존재하지 않는 주소이다. SSD 컨트롤러는 쓰기 요청 4를 DRAM 버퍼 대기열 상단에 추가한다. 그림 4 (b)에서 쓰기 요청 4의 논리 주소 5는 DRAM 버퍼에 존재하는 주소이므로 쓰기 요청 1의 논리 주소 5가 지정하는 데이터를 갱신하고, 쓰기 요청 1을 대기열 상단으로 이동시킨다. 쓰기 요청 4는 논리 주소 5의 데이터를 갱신하기 위한 요청이므로 버퍼 내부의 해당 데이터를 갱신한 뒤 사라진다. 쓰기 요청 1은 갱신이 이루어졌으므로 핫 데이터로 간주되고 태그가 기록되며, 핫 데이터 태그는 DRAM 버퍼에서 메모리 블록으로 데이터가 옮겨질 때 참조된다.

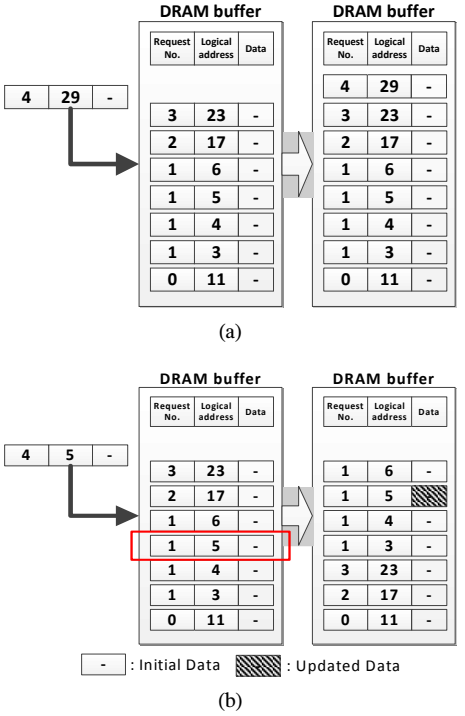

그림 4. DRAM 버퍼의 쓰기 요청 처리 예. (a) 논리 주소가 DRAM 버퍼에 존재하지 않는 경우, (b) DRAM 버퍼에 존재하는 경우.
Fig. 4. Write request process in DRAM buffer. (a) Logical address does not exist, (b) Logical address exists.

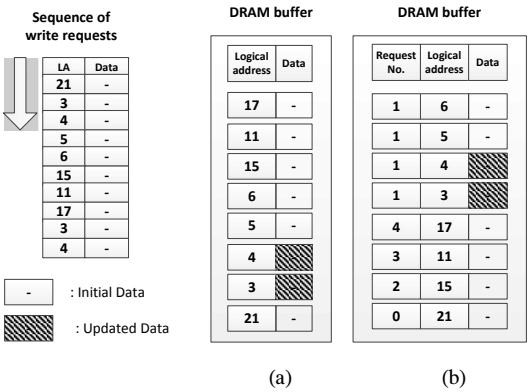

그림 5. 쓰기 요청 처리의 예. (a) 기존의 알고리듬, (b) 제안한 알고리듬.
Fig. 5. Example of processing write requests. (a) Previous algorithm, (b) Proposed algorithm.

그림 5는 기존의 방식과 제안한 방식으로 DRAM 버퍼에서 쓰기 요청을 처리하는 과정을 보인다. 그림 5 (a)는 기존의 DRAM 버퍼 사용 방식으로 쓰기 요청이 지정하는 논리 주소와 그에 상응하는 데이터를 도착한 순서대로 DRAM 버퍼에 갱신하고 추가한다. 그림 5 (b)는 제안한 방식으로 DRAM 버퍼가 쓰기 요청을 처리하는 과정을 보이며 쓰기 요청이 지정하는 논리 주소를 기준으로 공간적 지역성을 파악하고, 최근 접근 주소가 속한 쓰기 요청의 대기열 순서를 바꾸는 것에 의해 시간적 지역성을 고려한다. 제안한 방식은 공간적 지역성 특성을 이용하기 위해 DRAM 버퍼에 쓰기 요청 번호를 기록하고 연속한 논리 주소에 대한 쓰기 요청을 같은 번호로 묶는다. 쓰기 요청 1은 연속한 논리 주소 3~6을 대상으로 하는 쓰기요청으로 일련의 쓰기 요청을 처리하는 과정에서 논리 주소 3과 4의 데이터가 변경되어 대기열 상단으로 함께 이동한다. 논리 주소 5와 6은 제시된 예에서 아직 변경되지 않았지만 공간적 지역성 특성에 의하여 향후 변경될 가능성이 높다.

DRAM 버퍼의 용량이 한정되므로 현재 시점을 기준으로 논리 주소 접근 빈도가 높은 데이터만 유지하고 낮은 데이터는 플래시 메모리에 기록한다. 제안하는 기법은 DRAM 버퍼에서 LRU 유사 방식을 사용하여 플래시 메모리에 기록할 데이터를 결정한다. 버퍼에 존재하는 데이터 논리 주소의 최근 사용 여부를 파악하기 위해 링크드 리스트를 사용하고, 버퍼에 존재하는 논리 주소의 쓰기 요청이 도착하여 데이터가 갱신될 때 해당 쓰기 요청을 리스트 가장 뒤로 이동시킨다. 플래시 메모리에 데이터를 기록하는 경우 리스트 앞에 위치한 데이터부터 기록하여 버퍼에 논리 주

소 접근 빈도가 높은 데이터만 유지한다. 제안하는 알고리듬은 선정된 데이터로 논리 슈퍼블록을 구성하여 플래시 메모리에 기록한다. 선정된 데이터가 과거에 갱신된 적이 있는 데이터인 경우 버퍼 안에서 한 번도 갱신되지 않은 데이터에 비해 갱신될 가능성이 높다. 버퍼에서 데이터를 분류하는 단계에서 갱신이 한 번이라도 이루어진 데이터는 태그를 기록하여 핫 데이터로 분류한다. 태그는 버퍼에서 데이터를 꺼내 플래시 메모리에 저장하는 경우 해당 데이터의 종류를 확인하고 저장될 위치를 결정하는데 사용된다. 기존의 웨어레벨링 기법은 블록 간 소거횟수 차이를 줄이기 위해 소거횟수가 가장 작은 블록부터 데이터를 기록하나, 제안하는 기법이 적용되는 멀티채널 구조에서는 각 채널 별 가장 소거횟수가 작은 블록들 사이에 소거횟수 차이가 존재할 가능성이 크다. 이 차이를 줄이기 위해 기록되는 데이터의 종류를 기준으로 기록할 데이터가 핫 데이터인 경우 상대적으로 블록의 소거횟수가 적은 채널의 블록을 선택하고, 콜드 데이터인 경우 소거횟수가 많은 채널의 블록을 선택하여 블록 간 소거횟수 차이를 감소시킨다. 핫 데이터의 논리 주소는 콜드 데이터의 것에 비해 다시 참조될 가능성이 높아 해당 블록에 저장된 데이터가 무효화 될 가능성이 높다. 가비지 콜렉션의 희생 블록은 무효화 된 데이터가 많은 블록으로 결정되므로, 사용 횟수가 적은 블록에 갱신 가능성이 높은 핫 데이터를 저장하여 다른 블록에 비해 무효 데이터를 많이 발생시켜 희생 블록으로 선택되도록 유도한다.

그림 6은 기존의 방식과 제안한 방식으로 버퍼링된 데이터가 메모리 블록에 할당될 때 처리과정의 차이를 보인다. 그림 6 (a)에서 기존의 방식은 FIFO 방식으로 데이터를 할당하여 다른 데이터에 비해 변경될 확률이 높은 데이터임에도 블록 소거횟수가 많은 블록에 할당되므로 데이터가 변경되어 블록 소거횟수가 증가할 가능성이 높다. 그림 6 (b)는 제안한 방식으로 데이터를 메모리 블록에 할당한 예로 쓰기 요청 1은 공간적 지역성 특성을 이용하기 위해 같은 블록에 저장된다. 소거횟수가 많은 슈퍼블록 1에 변경될 가능성이 적은 데이터를 할당하고 소거횟수가 적은 슈퍼블록 0에 변경될 가능성이 높은 데이터를 저장하여 블록 간 소거횟수 차이를 줄인다. 제안된 알고리듬은 DRAM 버퍼에서 데이터의 지역성을 파악하고 논리 주소 접근 빈도에 따라 데이터를 분류하여, 상대적으로 소거횟수가 많은 블록에 접근 빈도가 낮은 데이터를 할당하는 방식으로 블록 간 소거횟수 차이를 감소시킨다.

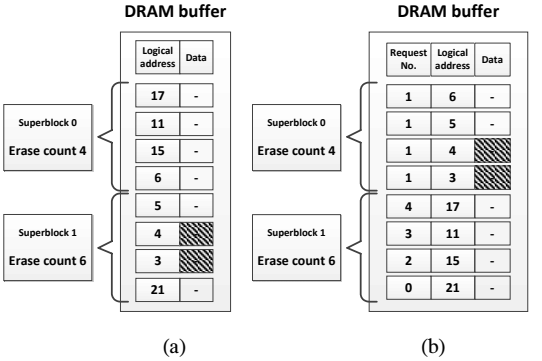

그림 6. 분류된 데이터 할당 예. (a) 기존 알고리듬, (b) 제안한 알고리듬.
Fig. 6. Example of allocating classified data. (a) Previous algorithm, (b) Proposed algorithm.

Ⅳ. 실험 결과

제안한 방법의 효용성을 검증하기 위해 가상의 SSD 환경을 조성하여 시뮬레이션을 수행하였다. 제안된 방법은 8MB의 DRAM 버퍼를 사용하는 4-채널 2-웨이 SSD 시뮬레이터를 C++로 구현하였고 사양은 표 1과 같다.

그림 7에서 시뮬레이터의 구성을 보인다. 실험에 사용된 모델은 소거 동작에 영향을 주지 않는 읽기 동작을 제외하고 쓰기 동작에 대해서 작동한다. 시뮬레이터는 실제 SSD의 구조와 동작을 모델링하여 구현하였고 어플리케이션으로부터 얻은 트레이스의 쓰기 요청을 입력으로 한다. 시뮬레이터는 입력된 쓰기 요청의 논리 주소와 데이터 길이 정보를 버퍼로 전달하고 데이터 분류기는 입력된 정보와 버퍼의 데이터를 비교하여 핫 데이터 발생 여부를 판단하고 대기열에 입력된 쓰기 요청을 추가시킨다. 버퍼가 가득 찬 경우 내보내는 데이터의 핫/콜드 여부에 따라 데이터 할당 모듈은 블록 정보를 참조하여 데이터가 기록될 위치

표 1. 실험에 사용된 NAND 플래시 메모리 파라미터
Table 1. Parameter of NAND flash memory used for simulation

Parameters	Value
Number of channels	4
Number of planes per channel	2
Number of blocks per plane	8,192
Number of pages per block	64
Page size	2KB
Address mapping unit	Superpage

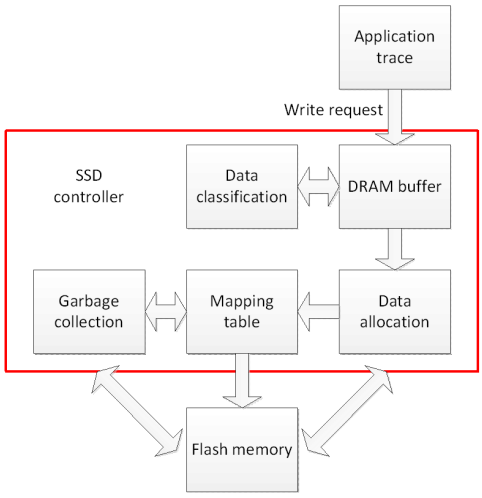

그림 7. SSD 시뮬레이터 구조.
Fig. 7. SSD simulator architecture.

를 결정한다. 트레이스는 WINDOWS XP을 사용하는 PC와 노트북 환경에서 Diskmon 프로그램을 사용하여 어플리케이션이 실행되는 동안 디스크 접근 패턴 중 쓰기 요청을 추출하여 구했다. 미디어 파일 재생 관련 트레이스는 인터넷 스트리밍을 통한 재생과정에서 획득하였고, 웹 브라우저 트레이스는 웹 사이트 관련 임시 파일 저장과 여러 파일을 다운로드 하는 과정에서 발생하는 쓰기 요청을 추출하였다.

기존의 방법은 인접한 블록 사이에 연관 없이 각각의 블록을 기준으로 연구되어 쓰기 동작 이후에 적용되는데 반하여, 제안된 알고리듬은 메모리에 데이터를 쓰는 단계부터 슈퍼블록 내부의 데이터 연관성을 높인다. 문서 편집기와 스트리밍 MP3 데이터 다운로더, 스트리밍 동영상 데이터 다운로더, 웹 브라우저 실행으로 얻은 트레이스에 대해 시뮬레이션을 수행한 결과는 표 2와 같으며, 기존 방식에 비해 평균적으로 9.3% 감소하였다.

표 2. 소거횟수 편차 비교 결과
Table 2. Comparison result of differences in erase counts

Programs	Dual Pool[17]	Proposed	Comparison (%)
Document editor	63	58	-7.9
Streaming MP3 data downloader	66	59	-10.6
Streaming MPEG data downloader	69	65	-5.8
Web browser	77	67	-13.0
Average	68.8	62.3	-9.3

표 3에서 보는 바와 같이 총 소거횟수를 비교한 결과 제안된 알고리듬은 기존 알고리듬과 비교하여 평균 -4.6%의 차이를 보인다. 기존 방법은 블록의 소거횟수 균등화를 위해 소거횟수가 적게 발생한 블록의 데이터와 소거횟수가 많이 발생한 블록의 데이터를 교환하여 이에 따른 추가적인 오버헤드가 발생하고, 이는 소거동작이 증가하는 원인이 된다. 쓰기 단계부터 지역성을 이용하여 슈퍼블록 내부에 접근 빈도가 비슷한 데이터가 저장되게 하는 제안된 알고리듬과는 달리, 기존 알고리듬은 쓰기 단계 이후에 웨어레벨링을 수행하기 때문에 슈퍼블록 내부에 서로 다른 접근 빈도를 가진 데이터가 저장되는 경우가 발생할 확률이 높다. 제안된 알고리듬은 슈퍼블록 내부에 비슷한 접근 빈도를 가진 데이터를 저장하여 해당 블록이 가비지 콜렉션의 희생 블록으로 선정되는 경우에 발생하는 오버헤드를 감소시키고, 가비지 콜렉션의 효율을 높여 총 소거횟수를 감소시킨다.

표 3. 총 소거횟수 비교 결과 (단위: 1,000)
Table 3. Comparison result in total number of erase counts

Programs	Dual Pool[17]	Proposed	Comparison (%)
Document editor	225,362	213,192	-5.4
Streaming MP3 data downloader	210,571	200,463	-4.8
Streaming MPEG data downloader	218,493	205,601	-5.9
Web browser	237,124	231,670	-2.3
Average	222,887	212,731	-4.6

V. 결 론

웨어레벨링은 블록 간의 마모도 차이를 줄여 플래시 메모리의 수명을 연장한다. 본 논문에서는 멀티채널과 멀티웨이 구조에서 데이터 분류 단계와 데이터 할당을 통해 블록간 소모횟수 차이를 감소시키는 방법을 제안하였다. 기존의 알고리듬은 블록들이 개별적으로 매핑되는 환경에 대해 고안되어 여러 블록들이 슈퍼블록을 구성하는 경우에 적용시킬 때 성능 저하를 가져온다. 제안된 알고리듬은 지역성 특성을 감안

하여 데이터를 분류하고 SSD의 구조 특성을 고려하여 소거횟수의 편차를 감소시키는 방향으로 데이터를 저장한다. 기존 알고리듬과 비교하여 블록 소거횟수 차이는 -9.3 %, 총 소거횟수는 -4.6 % 감소시킨다. 추후에는 데이터 분류 단계에서 지역성 특성을 보다 정확하게 파악하는 방법에 대한 추가 연구가 필요하다.

References

[1] S. Shim, H. Kang, S. Jung, and Y. Song, "Page mapping table caching technique for large scale NAND flash memory," in *Proc. Symp. Korean Inst. Commun. Inf. Sci. (KICS)*, pp. 463-464, Yongpyong, Korea, Feb. 2012.

[2] J. Kim, J. Kim, S. Noh, S. Min, and Y. Cho, "A space-efficient flash translation layer for compact flash systems," *IEEE Trans. Consumer Electron.*, vol. 48, no. 2, pp. 366-375, May 2002.

[3] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J. Tauber, "Storage alternatives for mobile computers," in *Proc. Symp. Operating Syst. Design and Implementation*, pp. 25-37, Monterey, CA, Nov. 1994.

[4] S. Lim and K. Park, "An efficient NAND flash file system for flash memory storage," *IEEE Trans. Comput.*, vol. 55, no. 7, pp. 906-912, Jul. 2006.

[5] S. Boyd, A. Horvath, and D. Dornfeld, "Life-cycle assessment of NAND flash memory," *IEEE Trans. Semiconductor Manufacturing,* vol. 24, no. 1, pp. 117-124, Feb. 2011.

[6] D. Park and J. Lee, "Performance of the coupling canceller with the various window size on the multi-level cell NAND flash memory channel," *J. Korean Inst. Commun. Sci.*, vol. 37, no. 8, pp. 706-711, Aug. 2012.

[7] D. Lee and W. Sung, "Adaptive quantization scheme for multi-level cell NAND flash memory," *J. Korean Inst. Commun. Sci.*, vol. 38, no. 6, pp. 540-549, Jun. 2013.

[8] J. Ha and J. O, "Error correction code for NAND flash memory," *J. Korean Inst. Commun. Sci.*, vol. 28, no. 9, pp. 58-68, Aug. 2011.

[9] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys*, vol. 37, no. 2, pp. 138-163, Jun. 2005.

[10] Y. Hu, N. Xiao, and X. Liu, "An elastic error correction code technique for NAND flash-based consumer electronic devices," *IEEE Trans. Consumer Electron.*, vol. 59, no. 1, pp. 1-8, Feb. 2013.

[11] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill, "Bit error rate in NAND flash memories," in *Proc. Reliability Physics Symp.*, pp. 9-19, Phoenix, AZ, Apr. 2008.

[12] C. Park, P. Talawar, D. Won, M. Jung, J. Im, S. Kim, and Y. Choi, "A high performance controller for NAND flash-based solid state disk," in *Proc. 21st IEEE Non-Volatile Semiconductor Memory Workshop*, pp. 17-30, Monterey, CA, Feb. 2006.

[13] S. Park, S. Ha, K. Bang, and E. Chung, "Design and analysis of flash translation layers for multi-channel NAND flash-based storage devices," *IEEE Trans. Consumer Electron.*, vol. 55, no. 3, pp. 1392-1400, Aug. 2009.

[14] J. Kang, J. Kim, C. Park, H. Park, and J. Lee, "A multi-channel architecture for high-performance NAND flash-based storage system," *J. Syst. Architecture*, vol. 53, no. 9, pp. 644-658, Sept. 2007.

[15] M. Jung, E. Wilson, D. Donofrio, J. Shalf, and M. Kandemir, "NANDFlashSim: Intrinsic latency variation aware NAND flash memory system modeling and simulation at microarchitecture level," in *Proc. IEEE 28th Symp. Mass Storage Systems and Technol.*, pp. 1-12, San Diego, CA, Apr. 2012.

[16] M. Chiang, P. Lee, and R. Chang, "Managing flash memory in personal communication devices," in *Proc. Int. Symp. Conumer Electron.*, pp. 177-182, Singapore, Dec. 1997.

[17] L. Chang "On efficient wear leveling for large-scale flash-memory storage systems," in

Proc. ACM Symp. Applied Computing, pp. 1126-1130, Seoul, Korea, Mar. 2007.

김 동 호 (Dong-ho Kim)

2012년 2월 : 서강대학교 전자공학과 졸업

2012년 2월~현재 : 서강대학교 전자공학과 CAD & ES 연구실 석사과정

<관심분야> Embedded System Design, ASIP Design

황 선 영 (Sun-young Hwang)

1976년 2월 : 서울대학교 전자공학과 학사

1978년 2월 : 한국과학원 전기 및 전자공학과 공학석사

1986년 10월 : 미국 Stanford대학 전자공학 박사

1976~1981년 : 삼성반도체(주) 연구원, 팀장

1986~1989년 : Stanford 대학 Center for Intergrated System 연구소 책임연구원 및 Fairchild Semiconductor Palo Alto Research Center 기술 자문

1989~1992년 : 삼성전자(주) 반도체 기술 자문

1989년 3월~현재 : 서강대학교 전자공학과 교수

<관심분야> SoC 설계 및 framework 구성, CAD 시스템, Embedded System 설계, Computer Architecture 및 DSP System Design 등