Journal of Digital Contents Society Vol. 14 No. 1 Mar. 2013(pp. 51-57)
http://dx.doi.org/10.9728/dcs.2013.14.1.51

마모도 평준화를 위한 File Clustering 알고리즘

이태화* 차재혁**

요 약

플래시 메모리 기반의 저장 장치는 고성능, 저전력, 내구성과 경량 등의 특징을 가지고 있어 기존에 사용되고 있던 저장장치를 빠르게 대체하고 있다. 플래시 메모리 기반의 저장 장치는 기존 저장장치인 블록 저장 장치로 가상화하기 위한 계층인 FTL (Flash Translation Layer) 을 가지고 있다. 가비지 컬렉션(Garbage Collection)은 FTL의 주요한 기능으로서 플래시 메모리의 수명과 성능에 큰 영향을 끼친다. 플래시 메모리의 수명은 가비지 컬렉션에 의해 발생되는 지우기의 횟수와 마모도의 영향을 받는다. 본 논문에서는 마모도 평준화 개선을 위해 File 정보를 알 수 있는 환경에서 File Clustering 알고리즘을 제시한다. File Clustering은 같은 File에서의 요청이 또다시 같이 호출 될 것을 기대하여 같은 File로부터 온 요청을 같은 블록에 할당하는 알고리즘이다. 이를 위해 FTL의 기능 중 페이지 할당 정책을 제안하였고, 최소한의 마모도 평준화를 보장하기 위해 MIN-MAX GAP을 사용하였다. 본 논문에서 제안하는 알고리즘을 검증하기 위해 TPC 벤치마크를 이용하였고 이를 통해 마모도 평준화 하지 않은 분산보다 690%이상 값이 개선되었고, 기존에 연구되던 Hot/Cold보다도 좋은 분산을 갖는 것을 보였다.

키워드 : 플래시 메모리, 가비지 컬렉션, 마모도 평준화

A File Clustering Algorithm for Wear-leveling

Taehwa Lee*, Jaehyuk Cha**

Abstract

Storage device based on Flash Memory have many attractive features such as high performance, low power consumption, shock resistance, and low weight, so they replace HDDs to a certain extent. An Storage device based on Flash Memory has FTL(Flash Translation Layer) which emulate block storage devices like HDDs. A garbage collection, one of major functions of FTL, effects highly on the performance and the lifetime of devices. However, there is no de facto standard for new garbage collection algorithms. To solve this problem, we propose File Clustering Algorithm. File Clustering Algorithm respect to update page from same file at the same time. So, these are clustered to same block. For this mechanism, We propose Page Allocation Policy in FTL and use MIN-MAX GAP to guarantee wear leveling. To verify the algorithm in this paper, we use TPC Benchmark. So, The performance evaluation reveals that the proposed algorithm has comparable result with the existing algorithms(No wear leveling, Hot/Cold) and shows approximately 690% improvement in terms of the wear leveling.

Keywords : Flash Memory, FTL, Garbage Collection, Wear leveling

1. 서론

최근 낸드 플래시 메모리(NAND Flash Memory) 기반 SSD(Solid State Drives)는 기존 저장 장치인 하드디스크를 대체하는 저장 장치

※ 교신저자(Corresponding Author): Jaehyuk Cha
접수일:2013년 01월 18일, 수정일:2013년 03월 13일
완료일:2013년 03월 22일
* 한양대학교 컴퓨터소프트웨어학과 박사과정
email: alghost@hanyang.ac.kr
** 한양대학교 컴퓨터공학 교수
Tel: +82-2-2220-1158, Fax: +82-2-2220-4459
email: chajh@hanyang.ac.kr
▣본 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2011-0029181).

로 주목받고 있다[1]. 이는 플래시 메모리가가 빠른 속도, 저전력, 내구성과 같은 장점을 갖고 있기 때문이다. 하지만 SSD의 내부 저장 소자인 플래시 메모리는 덮어쓰기가 불가능하여 지우기 연산이 존재한다. 또한 이 지우기 연산은 페이지들의 집합인 블록단위로 이루어져있다. 그렇기 때문에 SSD는 하드 디스크의 덮어쓰기를 지원하기 위해 블록을 지운 후에 유효 페이지의 데이터를 덮어쓰기 하는데 이 때 블록 내에 다른 유효 페이지가 있다면 이 페이지 또한 다시 써야 한다. 그러나 지우기 연산은 읽기/쓰기에 비해 속도가 느리고 각 블록은 제한된 지우기 횟수를 가지고 있고 제한된 지우기 횟수를 초과할 경우 배드 블록이 되어 사용할 수 없는 문제가 있다. 또한 배드 블록이 일정 수준을 초과하면 플래시 메모리를 사용할 수 없게 된다. 이러한 이유로 SSD는 지우기 후 쓰기를 보완하는 방법으로 out-of-place 쓰기를 사용하여 덮어쓰기 문제를 해결하고 있다. 이러한 out-of-place 쓰기에 따른 자료구조 관리 및 알고리즘과 플래시 메모리의 고유 특징에 의한 문제를 해결하기 위해 SSD는 FTL(Flash Translation Layer)라는 소프트웨어 모듈을 두고 있다.[2]

SSD의 성능은 FTL의 기능에 영향을 많이 받기 때문에 FTL에 대해 많은 연구가 이루어졌다. 그 중 가비지 컬렉션(garbage collection)에 대한 연구도 활발히 진행되고 있다. 가비지 컬렉션은 플래시 메모리의 out-of-place 쓰기 작업으로 인해 생긴 유효하지 않은 페이지를 수집하여 빈 블록을 확보하는 작업으로서, 플래시 메모리의 블록에 대한 지우기 연산과 블록에 존재하는 유효한 페이지들의 복사 과정을 수반한다. 가비지 컬렉션과 같은 FTL이 갖는 주요 기능의 알고리즘들은 플래시 메모리의 지우기 발생 횟수, 쓰기 증폭률(write amplification factor),마모 평준화(wear-leveling)등에 영향을 끼친다. 본 논문에서는 가비지 컬렉션이 영향을 끼치는 요소 중 마모 평준화를 개선하기 위한 알고리즘을 제시한다. [3][4][5][6][7][8][9].

본 논문의 구성은 다음과 같다. 2장에서는 플래시 메모리내 모듈 FTL의 주요 기능 중 마모도 평준화에 대해 설명하고, 마모도를 평준화하기 위해 기존에 연구된 알고리즘을 소개한다. 3장에서는 본 논문에서 제시하는 메커니즘에 대해 설명을 한다. 4장에서는 실험 결과 및 분석을 설명하고, 마지막으로 5장에서 최종적인 결론을 맺고 향후 연구 과제를 설명한다.

2. 관련연구

이 장에서는 플래시 메모리와 플래시 메모리가 블록 장치로 사용되기 위해 가지고 있는 FTL의 내부 기능 중 본 논문에서 다루고자 하는 마모도 평준화에 대해 설명을 하고 이를 나타내기 위한 성능 평가 척도를 제시한다.

2.1 플래시 메모리

플래시 메모리는 SSD와 eMMC등과 같은 저장 장치로 활용되어 기존에 사용되던 하드디스크의 대안으로 사용되고 있다. 이는 플래시 메모리가 빠른 속도, 휴대성, 내구성 등과 같은 장점을 갖기 때문이다. 하지만 플래시 메모리의 경우 약점이 존재하는데 이는 다음과 같이 몇 가지로 정리할 수 있다.

첫 째로 기존에 사용되던 하드디스크는 읽기/쓰기 단위가 섹터였다면 플래시 메모리는 페이지 단위이다. 이는 기존 단위보다 훨씬 큰 단위로 플래시 메모리의 물리적인 최소 단위이다.

둘째로 하드디스크는 재 쓰기가 가능하였던 반면 플래시 메모리는 재 쓰기가 불가능하다. 플래시 메모리는 페이지에 대한 재 쓰기가 불가능하여 지우기 연산을 추가로 제공하고 있다. 하지만 지우기 연산은 페이지의 집합인 블록 단위로만 이루어지기 때문에 많은 단점이 생긴다. 덮어쓰고자 하는 페이지가 속한 블록을 지워야 하기 때문에 블록 안에 존재하는 모든 페이지를 임시 버퍼로 복사 한 후 블록을 지우고 다시 써야 한다.

셋째로 플래시 메모리를 이루고 있는 블록은 제한된 지우기 횟수를 갖는다. 이는 플래시 메모리 타입에 따라 다른데 SLC의 경우 100,000회, MLC의 경우 10,000회, TLC의 경우 1,000회에 이른다. 제한된 지우기 횟수를 넘게 되면 해당 블록은 배드 블록이 되어 사용할 수 없게 되며, 배드 블록이 일정 수준 이상 생기면 플래시 메모리를 사용 할 수 없게 된다.

플래시 메모리는 이와 같은 단점을 갖고 있기

때문에 하드디스크와 동일하게 사용 되면 잦은 지우기에 의해 저장 장치로 사용할 수 없게 된다. 그렇기 때문에 플래시 메모리를 활용한 저장 장치들은 FTL을 두어 이러한 문제를 해결하고 있다. FTL은 위와 같은 문제를 해결하기 위해 호스트로부터 오는 요청을 논리 주소로 사용하여 주소 체계를 분리하고 이를 매핑한다. 재 쓰기 요청 시 다른 물리 주소에 쓰고 매핑 정보만 업데이트함으로써 재 쓰기 문제를 해결할 수 있다. 하지만 재 쓰기에 의해 생기는 유효하지 않은 페이지를 수집해야 하는 문제가 생긴다. 그래서 FTL은 이를 수집하여 빈 블록을 얻는 가비지 컬렉션 기능이 있다. 이를 통해 재 쓰기 시 발생하는 지우기 횟수를 줄이고 궁극적으로 수명을 늘리고 성능을 개선하였다.

이와 같이 플래시 메모리 기반의 저장 장치는 FTL을 가지고 있으며 각 저장 장치마다 다양한 매핑과 가비지 컬렉션 알고리즘을 갖는다.

2.2 마모도 평준화 (Wear-Leveling)

플래시 메모리는 배드 블록이 많아지면 사용할 수 없는 단점을 갖고 있다. 마모도 평준화란 이러한 단점을 보완하기 위한 것으로 지우기 연산이 모든 블록에 균등하게 발생하게 하는 것이다. 이는 FTL의 기능 자체가 아닌 FTL에 존재하는 기능을 개선하여 유도하는 것으로 페이지 할당 정책, 희생 블록 선정 정책 등과 같은 기본 기능을 마모도가 평준화 될 수 있도록 개선한다. 마모도 평준화를 위해 FTL은 추가적인 지우기 횟수를 발생시킬 수 있다. 마모도 평준화는 모든 블록이 균등한 지우기 횟수를 갖는 것이기 때문에 지우기 연산을 수행하는 가비지 컬렉션과도 밀접하다. 가비지 컬렉션은 일반적으로 블록을 지울 때 블록 내 페이지를 복사하는데 이 때 발생하는 오버헤드를 줄이기 위해 유효 페이지가 가장 적은 유효 페이지를 갖는 블록을 희생 블록으로 선정한다. 하지만 자주 재 쓰기가 되는 페이지가 있는 블록은 계속해서 유효 페이지가 적기 때문에 해당 블록만 지우기 횟수가 높아져 결국 배드 블록이 될 수 있다.

이렇듯 사용 패턴에 따라 논리 페이지는 다양한 특성을 갖기 때문에 이러한 특성을 활용하여 할당하고 페이지가 분배된 블록을 이용하여 다양한 기법을 적용 할 수 있다. 일반적으로 Hot/Cold와 같이 자주 재 쓰기 되는 페이지와 자주 쓰이지 않는 페이지로 분류하여 이를 활용한 기법이 많이 연구되었다. 이는 Dual-Pool과 FeGC와 같은 연구로 구분 되는데, Dual-Pool은 논리 페이지의 빈도수를 고려하여 Hot/Cold를 정의 및 Clustering 한다. 또한 빈도수에 의해 고려된 상태 정의를 이용하여 블록 영역(pool)간의 교환을 하는 Dirty Swap이라는 기법을 통해 마모도를 평준화 하였다. FeGC는 논리 페이지의 업데이트 시점을 고려하여 Hot/Cold를 정의하였고 업데이트 시점과 같은 Recency와 Valid 페이지 개수와 같은 빈도수를 함께 고려하여 희생 블록을 선정한다. 또한 페이지를 Hot/Cold로 Clustering하고 블록 할당은 각 상태에 맞게 Hot은 지우기 횟수가 낮은 블록, Cold는 지우기 횟수가 높은 블록에 할당한다.

2.3 성능 평가 척도

일반적으로 하드디스크와 같은 디스크의 성능 실험에서는 IO 대역폭(IO bandwidth) 초당 IO 회수(IOPS: IO per seconds) 등이 중요한 성능 평가 척도로 활용되고 있다. 플래시 메모리의 경우에 앞서 말한 두 가지 또한 중요하지만 수명을 고려하는 척도 또한 중요하다[12]. 앞서 언급한바와 같이 배드 블록이 생김에 따라 신뢰할 수 있는 저장장치로 사용할 수 없기 때문에 플래시 메모리의 수명은 성능을 평가하는 중요한 요소이다[3]. 이를 위한 평가 척도로는 지우기 횟수와 블록별 지우기 횟수의 분산이 있다. 같은 횟수의 쓰기 요청이라도 FTL의 내부 알고리즘에 따라 이들 지우기 횟수 및 분산은 확연이 달라진다. 본 논문에서도 지우기 횟수와 분산을 성능 평가 척도로 활용한다.

3. File Clustering 알고리즘

이 장에서는 마모도 평준화 개선을 위해 File Clustering 알고리즘을 제시한다. 이 장에서 제안하는 알고리즘은 파일 정보를 포함한 트레이스를 활용하여 같은 파일로부터의 요청을 같은 블록에 Clustering 하는 알고리즘으로 같은 파일로 부터의 요청은 같이 재 쓰기 될 확률이 높다는 전제조건을 갖는다.

이 장은 페이지 할당 정책, 빈 블록 할당 정책, 희생 블록 선정 정책에 대해 설명한다.

3.1 페이지 할당 정책

File Clustering 알고리즘은 모든 LPN(Logical Page Number, 페이지)이 어떤 File로부터 요청이 왔는지 File 정보를 알고 같은 File의 요청을 같은 블록에 저장한다. 이를 위해 File ID(File의 고유ID)와 블록을 사상하는 테이블이 존재하게 되고 Write 요청 시 LPN의 File ID를 확인하여 File ID와 사상되는 블록이 있을 경우 그 블록에 할당하고, 없을 경우 새로운 블록을 할당받아 사상테이블을 업데이트한다. 이는 같은 File에 대한 업데이트가 동시에 발생할 확률이 높다는 전제를 갖는다.

3.2 빈 블록 할당 정책

File Clustering 알고리즘은 기존의 Hot/Cold와 다르게 Clustering하는 정보가 특성을 갖지 않기 때문에 각 File ID별로 다른 정책을 적용할 수가 없다. 그렇기 때문에 본 알고리즘에서는 지우기 횟수가 가장 낮은 블록을 할당한다.

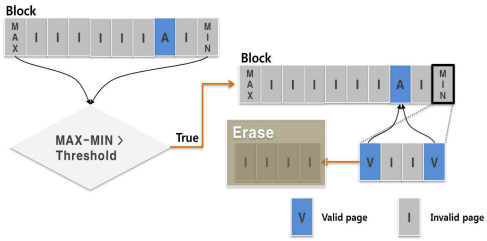
3.3 희생 블록 선정 정책

희생 블록 선정 정책 또한 각 File이 특성을 갖지 않기 때문에 널리 사용되고 있는 Greedy 기법[10]을 사용한다. Greedy 기법은 가비지 컬렉션 수행 시 페이지 복사를 최소화하기 위해 블록 내 유효 페이지가 가장 적은 블록을 희생 블록으로 선정하는 기법을 말한다.

3.4 MIN-MAX GAP 기반 희생 블록 선정 기법

마모도 평준화는 Clustering 기법에 의해 연구되었다. 하지만 File Clustering 알고리즘의 경우 각 File의 요청 패턴에 따라 마모도가 불균형을 이룰 수 있다. 이를 해결하기 위해 FTL의 기능과 별개로 MIN-MAX GAP 기반 희생 블록 선정이 있다. 본 알고리즘은 이를 채용하여 사용한다.

(그림 1) MIN-MAX GAP 기반 희생 블록 선정 기법



(Figure 1) Victim Block Selection Technique based on MIN-MAX GAP

본 기법은 지우기 횟수의 최대 최소의 차이가 Threshold를 넘지 못하도록 하여 최소한의 마모도 평준화를 보장한다. 플래시 메모리에 있는 블록들이 갖는 지우기 횟수의 최대와 최소의 차이가 Threshold를 넘으면 최소의 지우기 횟수를 갖는 블록을 자주 참조되지 않는 File의 페이지이며 그렇기 때문에 희생 블록으로 선정되지 못하는 상황이 발생하여 마모도 불균형을 야기한다. 이를 강제로 희생 블록으로 선정되도록 하여 선정된 후 바로 가비지 컬렉션에 의해 지우기를 수행한다. 이를 통해서 SSD는 Threshold값에 의해 마모도 평준화가 보장된다.

4. 실험

이 장에서는 본 논문이 제안한 File Clustering의 성능을 검증하기 위한 실험과 실험을 위해 File ID를 가져온 환경에 대해 설명하였다.

4.1 트레이스

<표 1> 트레이스 추출 환경

Attributes	Values
OS	Solaris 10
Transactions	10000
Thread	10
Database	PostgreSQL
Tool for extraction	dtrace

<Table 1> Environment to extract the trace

본 논문의 기법의 성능을 확인하기 위해 TPC-B 벤치마크를 사용하였다. TPC는 Transaction Processing Performance Council로 OLTP 및 스트레스 벤치마크 환경이다. 그중 TPC-B는 OLTP가 아닌 데이터베이스의 스트레스를 테스트하기 위한 벤치마크이다. <표 1>와 같은 환경에서 TPC-B를 수행하여 트레이스를 추출하였다.

또한 Solaris와 dtrace를 활용한 이유는 온라인 트랜잭션의 트레이스를 추출하는 과정에 있어서 File 정보를 추출 할 수 있는 툴이기 때문이다.

4.2 실험 환경

<표 2> 실험 환경

Attributes	Values
OS	Ubuntu 10.04
Page Size	8K
# of Blocks	625
# of Pages per Block	64

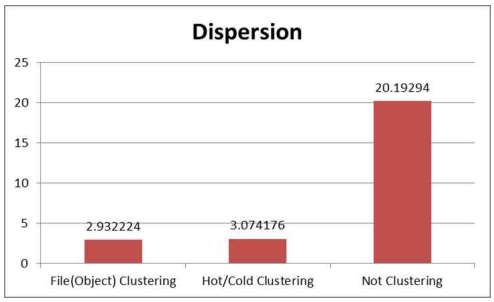
<Table 2> Environment for experiment

실험 환경은 <표 2>에서 보는 바와 같이 리눅스 시스템에서 이루어졌다. Page 크기는 8K로 계산하여 총 40000K의 용량을 갖는 플래시 메모리를 유저 레벨의 코드로 시뮬레이션 하였다. 마모도 평준화를 판단하기 위해서는 잦은 지우기 횟수를 가져야하기 때문에 용량을 약 40M로 작은 플래시 메모리를 가정하였다.

4.3 실험 및 평가

실험결과는 앞서 언급한 성능 평가 척도를 실험하여 그래프로 제시한다.

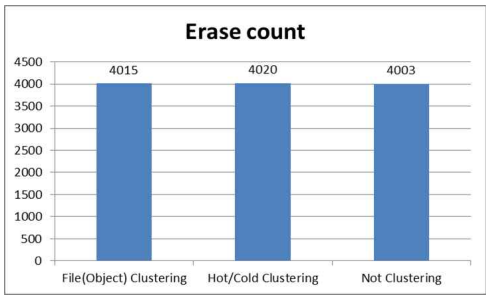
(그림 2) TPC-B 실험 결과 (분산)



(Figure 2) Experiment result for TPC-B (dispersion)

위 그래프의 결과에서 알 수 있듯이 TPC-B 환경에서 Clustering을 하지 않았을 경우보다 약 690%이상 성능이 좋아졌다. 또한 기존의 마모도 평준화 기법과 같은 빈도수 기반의 Clustering중 하나인 Hot/Cold Clustering 보다도 성능이 개선되었다. Hot/Cold Clustering은 잦은 빈도를 갖는 Hot페이지와 드문 빈도를 갖는 Cold페이지를 나누어 Clustering하는 알고리즘으로 빈도수에 기반하여 페이지의 특성을 추측하여 같은 블록에 모으는 기법을 말한다.

(그림 3) TPC-B 실험 결과 (지우기 횟수)



(Figure 3) Experiment result for TPC-B (Erase Count)

총 지우기 횟수의 경우 Clustering을 하지 않았을 때와 Clustering을 했을 때의 차이가 12~17회 정도로 매우 적은 값의 차이를 볼 수 있다. 또한 Hot/Cold Clustering보다 File Clustering의 경우 지우기 횟수가 더 적어진 것을 볼 수 있다.

(그림 3), (그림 4)의 그래프에서 알 수 있듯이 File Clustering은 기존에 연구된 빈도수 기반

의 Hot/Cold Clustering보다 분산이 더 개선되어 마모도 평준화 개선에 더 유리하다는 것을 알 수 있다. 또한 마모도 평준화 개선 시 발생될 수 있는 지우기 횟수 증가와 같은 오버헤드도 기존 연구보다 더 적은 것으로 나타났다.

5. 결론

본 논문에서는 플래시 메모리의 수명을 연장하기 위한 방법으로 File Clustering 알고리즘을 통해서 마모도 평준화를 개선하였다. FTL의 주요 기능인 빈 페이지 할당, 빈 블록 할당, 희생 블록 선정 등과 같은 기본 기능과 MIN-MAX GAP 기반 희생 블록 선정과 같은 기능을 추가함으로써 마모도를 평준화 하였다. 이처럼 기존 연구와 같이 빈도수에 따른 마모도 평준화 기법이 아닌 File과 같은 다른 척도를 활용한 다양한 연구가 필요할 것으로 생각된다.

References

[1] S. K. Lee, S. L. Min, Y. K. Cho, "Current trends on flash memory technology," Journal of KIISE, vol.24, no.12, pp.99-106, Dec. 2006. (in Korean)

[2] Tae-Sun Chung, Dong-Joo Park, Dong-Ho Lee, Sang-Won Lee, and Ha-Joo Song, "System Software for Flash Memory: A Survey", 2004.

[3] Jeong Su Park, Sang Lyul Min, "Flash Memory Wear-Leveling using Regulation Pools", Journal of Korean Institute of Information Scientists and Engineers (KIISE): Computing Practices and Letters, Vol.16 no12 (2010.12)

[4] Soung Hwan Lee, Tae Hoon Lee, Ki Dong Chung, "Adaptive Garbage Collection Policy based on Analysis of Page Ratio for Flash Memory", Journal of Korean Institute of Information Scientists and Engineers(KIISE) : Computer System and Theory Vol.36 no.5 (2009.10)

[5] L.-P. Chang, "On efficient wear-leveling for largescale flash-memory storage systems," Proc. of the 2007 ACM symposium on Applied computing, pp.1126-1130, 2007.

[6] Y.-H. Chang, J.-W. Hsieh, T.-W. Kuo, "Improving flash wear-leveling by proactively moving static data," IEEE Transactions on Computers, vol.59, no.1, pp.53-65, Jan. 2010.

[7] Jen-Wei Hsieh, Li-Pin Chang, Tei-Wei Kuo, "Efficient On-line Identification of Hot Data for Flash-Memory Management" ACM symposium on applied computing SAC 05, pp. 838, 2005.

[8] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," ACM Transactions on Embedded Computing Systems (TECS), vol.6, no.3, Jul. 2007.

[9] J. Kim, J.M. Kim, S.H. Noh, S.L. Min, and Y.Cho, "A Space-Efficient Flash Transaction on Consumer Electronics, Vol. 48, No.2, pp.366-375, 2002.

[10] A. Kawaguchi, S. Nishioka, and H. Motoda, "A flash-memory based File System." Proceedings of the USENIX Technical Conference, 1995.

[11] Sivan Toledo, "Algorithms and Data Structures for Flash Memories" ACM Computing Surveys.

[12] J.-U. Kang, H. Jo, J.-S. Kim, J. Lee, "A superblock-based flash translation layer for NAND flash memory," Proc. of the 6th ACM & IEEE International Conference on Embedded Software, pp.161-170, 2006.

[13] Xiao-Yu Hu, Robert Haas and Eleftheriou Evangelos, "Write amplification analysis in flash-based solid state drives", SYSTOR '09, The Israeli Experimental Systems Conference Articale No.10, 2009.

[14] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis and Roman Pletka, "Container Marking: Combining Data Placement, Garbage Collection and Wear Levelling for Flash", MASCOTS, 2011.

[15] Ioannis Koltsidas,Stratis D. Viglas, "Data management over flash memory", SIGMOD '11

[16] Ohhoon Kwon, Kern Koh, Jaewoo Lee, Hyokyung Bahn, “FeGC: An efficient garbage collection scheme for flash memory based storage systems”, The Journal of Systems and Software 84(2011) 1507-1523.

[17] Ho-Young Jung, Taehwa Lee, Jaehyuk Cha “An Offline FTL Algorithm to Verify the Endurance of Flash SSD”, Journal of Digital Contents Society. Vol.13 no14 p75-81.

차 재 혁

1987년 : 서울대학교 계산통계학과 (이학사)
1991년 : 서울대학교 컴퓨터공학과 (공학석사)
1997년 :서울대학교 컴퓨터공학과 (공학박사)
1998년~2001년 : 한양대학교 컴퓨터교육과 조교수
2002년~현 재 : 한양대학교 컴퓨터공학부 교수
관심분야 : 데이터베이스, 파일시스템, 플래시 메모리, 이러닝, 콘텐츠변환 등

이 태 화

2011년 : 안양대학교 컴퓨터학과 (공학사)
2011년~2013년 : 한양대학교 대학원 정보통신공학과 (석사과정)
2013년~현 재 : 한양대학교 대학원 컴퓨터 소프트웨어학과 (박사과정)
관심분야 : 데이터베이스, 파일시스템, 플래시 메모리, NVRAM 등