KIPS Trans. Comp. and Comm. Sys.
Vol.12, No.1 pp.1~8
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2023.12.1.1

Analysis and Advice on Cache Algorithms of SSD FTL

Hyung Bong Lee† · Tae Yun Chung††

ABSTRACT

It is impossible to overwrite on an already allocated page in SSDs, so whenever a write operation occurs a page replacement with a clean page is required. To resolve this problem, SSDs have an internal flash translation layer called FTL that maps logical pages managed by a file system of operating system to currently allocated physical pages. SSD pages discarded due to write operations must be recycled through initialization, but since the number of initialization times is limited the FTL provides a caching function to reduce the number of writes in addition to the page mapping function, which is a core function. In this study, we focus on the FTL cache methodologies reducing the number of page writes and analyze the related algorithms, and propose a write-only cache strategy. As a result of experimenting with the write-only cache using a simulator, it showed an improvement of up to 29%.

Keywords : SSD, FTL, LRU, MLC(Multi Level Cell), Write-only Cache

SSD FTL 캐시 알고리즘 분석 및 제언

이 형 봉† · 정 태 윤††

요 약

SSD는 이미 할당된 페이지에 대한 제자리 덮어쓰기가 불가능하므로 쓰기 연산이 있을 때마다 새로운 클린 페이지로의 대체가 필요하다. 이 문제를 지원하기 위해 SSD는 운영체제의 파일시스템에서 관리하는 논리 페이지를 현재 할당된 물리 페이지로 매핑하는 플래시 변환 계층인 FTL을 내부에 둔다. 쓰기 연산으로 버려진 SSD 페이지는 초기화 작업을 거쳐 재활용되어야 하는데, 그 횟수에 제한이 있기 때문에 FTL은 기본인 페이지 매핑 기능 외에 쓰기 횟수를 줄일 수 있는 캐시 기능을 제공한다. 이 연구에서는 쓰기 횟수를 줄이기 위한 FTL의 캐시 방법론에 집중하여 관련된 알고리즘들을 분석하고, 쓰기 전용 캐시 전략을 제안한다. 시뮬레이터를 사용하여 쓰기 전용 캐시를 실험한 결과 최대 29%의 개선 효과를 보였다.

키워드 : SSD, 플래시 변환 계층, LRU, 다계층셀, 쓰기 전용 캐시

1. 서 론

SSD(Solid State Drive)는 고체 상태인 반도체로 제작한 보조 저장 장치로서 HDD(Hard Disk Drive)를 대체하는 SSD는 대부분 NAND 플래시 메모리를 기반으로 한다[1]. SSD는 HDD에 비해 충격에 강하고, 입·출력 처리 용량이 크며, 접근 속도가 빠르며, 소모 전력이 낮다는 등의 여러 가지 장점에 힘입어 주변 저장 장치로서의 점유 비율을 높이고 있다[2-4]. 그러나 SSD는 플래시 메모리의 쓰기 전 초기화 특성에 의해 사용 중인 페이지에 대한 즉각적인 제자리 덮어쓰기가 불가능하다는 큰 단점을 갖고 있는데, 이는 이미 할당된 페이지에 대한 쓰기 작업을 위해서는 새로운 클린 페이지를 할당하여 대체해야 함을 의미한다[4]. 이처럼 할당된 디스크 페이지의 번호가 바뀌면 운영체제의 파일시스템에서 관리하는 페이지 번호의 수정이 요구되는데 이는 기존의 저장장치 HDD를 SSD로 간단하게 대체하려는 이른바 Plug & Play 취지에 어긋나 SSD 활용에 장애 요인이 된다. 이를 지원하기 위해 SSD는 자체적으로 플래시 변환 계층 FTL(Flash Translation Layer)을 두어 운영체제 관점에서의 논리 페이지 번호를 SSD의 물리 페이지 번호로 매핑하도록 한다[5]. FTL의 하드웨어 환경은 ARM 급 임베디드 프로세서와 512MB~4GB 정도의 DRAM으로 구성되어 SSD 비용 상승의 요인이다[6].

쓰기 작업에 의해 버려진 플래시 페이지는 가비지 수집 및 지우기 작업에 의해 재활용되어야 하는데 지우기 작업이 페

※ 이 논문은 2022년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원 지원을 받아 연구되었음(P0015356, 상용화를 위한 개인 맞춤형 재활 헬스케어 디바이스 연동 시스템 개발).
† 종신회원 : 강릉원주대학교 컴퓨터공학과 교수
†† 비 회 원 : 강릉원주대학교 전자공학과 교수
Manuscript Received : July 19, 2022
First Revision : August 29, 2022
Accepted : October 11, 2022
* Corresponding Author : Tae Yun Chung(tychung@gwnu.ac.kr)

이지들의 묶음인 블록 단위로만 가능하기 때문에 지우기 대상 블록에 포함된 유효 페이지들의 복사·이전을 위한 또 다른 쓰기 작업이 요구되어 쓰기 팽창 현상이 일어나고, 이를 완화하기 위해 FTL은 핵심적인 페이지 번호 매핑 기능 외에 플래시 페이지 쓰기 횟수를 최대한 줄일 수 있는 캐시 기능을 제공한다[7].

이 연구에서는 플래시 물리 페이지에 대한 쓰기 횟수 감소를 위한 FTL 캐시 알고리즘들을 분석하고, 그 알고리즘들의 불필요한 복잡성을 회피하면서도 쓰기 횟수 관점에서의 성능을 개선할 수 있는 쓰기 전용 캐시 활용을 제안한다. 이를 위하여 2장에서 FTL 캐시 알고리즘 연구를 살펴보고, 3장에서 쓰기 전용 캐시를 정의하고, 4장에서 쓰기 전용 캐시의 성능을 분석하며, 5장의 결론으로 이 논문을 맺는다.

2. FTL 캐시 알고리즘 관련 연구

2.1 일반적인 캐시 및 캐시 알고리즘

일반적인 캐시는 상대적으로 대량이고 느린 저장장치 앞에 위치하여 먼저 참조하는 소량의 빠른 저장장치를 말하며, 캐시에 없는 원래의 저장장치 내용 참조 시 그 내용을 캐시에 복사해놓고 이후에 참조함으로써 접근 시간 성능 향상을 주 목적으로 한다. CPU와 주기억장치 사이의 캐시와 주기억장치와 HDD 등 보조기억장치 사이의 버퍼 캐시 등이 대표적인 예이다.

캐시 운영에서의 주요 논점은 새로운 캐시 삽입을 위해 희생 페이지를 선정하는 페이지 교체 알고리즘에 있는데 적중률을 높이기 위해 대부분 LRU(Least Recently Used)를 근간으로 한다[8].

2.2 FTL 캐시 알고리즘

FTL 캐시는 Fig. 1과 같이 SSD에 내부에 존재하여 플래시 페이지에 대한 쓰기 횟수 감소에 기여한다[9]. 플래시 페이지에 대한 쓰기 억제를 위한 직관적인 방법은 특정 논리 페이지에 대한 쓰기 연산을 최대한 오랫동안 FTL 캐시에서 지원하고, 페이지 교체나 시스템 종료 시 등 마감 시점에 이르러 물리 페이지를 할당하여 업데이트하는 write-back 캐시 쓰기 정책[8]을 도입하는 것이다. 이 경우 특정 논리 페이지에 대한 얼마 동안의 쓰기·읽기 작업 후 폐기(TRIM)가 이루어진다면 물리 페이지에 대한 쓰기는 아예 발생하지 않을 수도 있다. 이처럼 쓰기에 의한 더티 페이지가 캐시에 가급적 오래 머물기 위해서는 페이지 교체 시 클린 페이지를 우선으로 선정할 필요가 있다.

1) CFLRU(Clear First LRU)

CFLRU[9]는 읽기와 쓰기 구분 없이 관리되는 LRU 리스트에서 상위 몇 개 범위에 클린 페이지가 존재하면 LRU 순위가 더티 페이지보다 낮더라도 그 클린 페이지를 교체 페이지로 선정한다. 읽기와 쓰기에 대한 시간적 교체 비용 가중치 비를 두고 특정 워크 로드의 입·출력에 대한 총 교체 비용을 지표로 하는 성능 평가 결과 LRU보다 우수하다. 이 방안은 더티 페이지가 캐시에 너무 오랫동안 머물러 전체적인 캐시 적중률을 저하한다는 비판을 받는다[10].

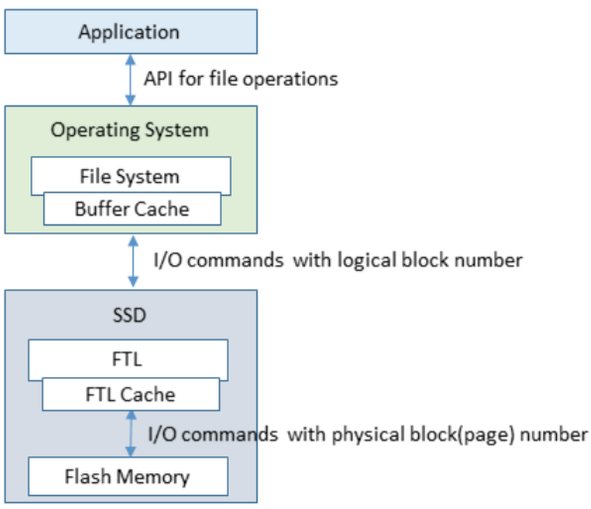


Fig. 1. Position of the FTL Cache

2) LRU-WSR(LRU Writes Sequence Reordering)

LRU-WSR[11]은 교체 대상인 LRU 페이지가 클린이면 그대로 교체하되, 더티 페이지라면 참조 빈도를 고려하여 빈도가 낮은 페이지를 먼저 선택하여 플래시에 쓰고 교체하는 방안을 시도한다. 이를 위해 더티 페이지에 대하여 이전의 LRU 페이지 교체 대상 체크 이후 참조가 있었다면 콜드(cold) 상태로 표시하여 다시 MRU 페이지로 보내어 다음 LRU 페이지 체크 시까지 여전히 콜드 상태라면 교체 대상으로 선정한다. 즉, 쓰기에 의한 초기의 더티 페이지가 콜드가 아닌 상태로 LRU 페이지로 이동하면 곧바로 교체하지 않고 콜드 표시와 함께 MRU 페이지로 보내고, 다음번 LRU 페이지로 이동하는 동안 참조가 있으면 콜드 표시를 제거하여 교체를 예방한다. 캐시 적중률과 쓰기 횟수를 지표로 도입한 성능 평가 결과는 쓰기 비율이 높은 워크 로드에서 CF-LRU보다 우수하다. 이 알고리즘은 클린 페이지에 대해서도 콜드 개념 적용의 가능성을 보인다.

3) CCF-LRU(Cold Clean First LRU)

CCF-LRU[12]는 LRU-WSR의 더티 페이지에 대한 콜드 상태와 콜드의 반대인 핫(hot) 상태를 도입하여 클린 페이지와 더티 페이지 모두에 적용함으로써 페이지 상태를 총 네 가지로 분류하여 콜드-클린 페이지를 최우선 교체 순위로 선정

하고 나머지는 LRU-WSR 알고리즘을 적용한다. 이를 위해 LRU 리스트를 콜드-클린 리스트와 나머지 혼합 리스트 등 두 개로 관리하고 콜드 상태는 LRU-WSR과 동일한 방법으로 판별한다. 다만, 클린 페이지는 콜드-클린 리스트와 혼합 리스트 사이를 오간다. 쓰기 비율이 다양한 워크 로드에 대한 성능 평가에서 캐시 적중률과 쓰기 횟수 모두 CFLRU와 CF-WSR보다 우수하다. 이 알고리즘은 클린 페이지 중에서도 클린-콜드 페이지를 먼저 교체함으로써 캐시 효율을 좀 더 개선한다는 점에 의의가 있으나, 클린 페이지 수가 극히 적을 경우에는 캐시 전체적인 효율성 측면에서 최근 들어온 클린 페이지의 이른 방출을 막지는 못한다.

4) AD-LRU(Adaptive LRU)

AD-LRU[13]는 콜드 LRU 리스트와 핫 LRU 리스트 등 두 개의 리스트를 두고 첫 참조 페이지는 콜드 리스트에서 출발하고, 참조가 일어나면 핫 리스트로 이동한다. 교체 페이지는 우선 콜드 리스트에서 선택하되 콜드 리스트 크기가 임계치를 밑돌면 핫 리스트에서 선택하게 함으로써 CCF-LRU의 단점을 보완한다. 각 리스트에서의 교체 페이지는 LRU 순위상 가장 높은 클린 페이지로 선정하되, 클린 페이지가 존재하지 않으면 더티 페이지에 대하여 LRU-WSR의 2차 기회 방식을 적용하고, 페이지 교체 및 새로운 페이지 삽입이 일어날 때마다 두 리스트 크기의 적응 변화가 이루어진다. 읽기 위주, 쓰기 위주, 랜덤 등의 워크 로드에 대한 성능 평가에서 캐시 적중률 및 쓰기 횟수가 CFLRU, LRU-WSR, CCF-LRU보다 대등하거나 높다.

5) DPW-LRU(Dynamic Page Weight LRU)

DPW-LRU[14]는 AD-LRU의 콜드 리스트와 핫 리스트를 ER(Exchange Region)과 WR(Working Region)로 각각 명명하고, 읽기 첫 참조는 ER에, 쓰기 첫 참조는 WR에 삽입하며, ER의 재참조 페이지는 WR로 이동한다. 여기서 ER은 방출 대상임을, WR은 방출 대상이 아님을 각각 의미한다. ER 삽입 시 ER의 크기가 임계치를 초과하면 순수한 LRU 알고리즘에 의해 희생 페이지를 선택하여 방출한다. WR 삽입 시 WR의 크기가 임계치를 초과하면 LRU 쪽 일부 페이지들을 대상으로 교체 비용인 DPW를 측정하여 비용이 가장 낮은 희생 페이지를 ER로 이동시켜 삽입한다.

특정 페이지의 교체 비용 DPW는 모든 입·출력 순서상에서의 참조 간격 즉 지역성을 의미하는 TLI(Temporal Locality Interval), 방출로 인해 예상되는 입·출력 지연을 의미하는 EC(Eviction Cost), 참조의 최근성을 의미하는 RE(Recency) 등 세 가지 지표를 사용한다. 읽기·쓰기 비율이 다양한 워크 로드에 대한 성능 평가에서 캐시 적중률, 쓰기 횟수, 입·출력 지연 측면에서 CFLRU, LRU-WSR, CCF-LRU, AD-LRU보다 우수하다.

3. FTL 쓰기 전용 캐시

3.1 플래시 메모리의 읽기·쓰기 지연 시간 특성

SSD의 플래시 메모리는 Table 1에서 보는 바와 같이 읽기, 쓰기, 지우기 연산에 대한 지연 시간이 모두 다르다[4, 16, 17]. 이처럼 플래시의 쓰기 지연 시간이 읽기 지연 시간보다 길므로 초기 FTL 캐시 알고리즘들은 가급적 교체 시 쓰기가 불필요한 클린 페이지를 우선 선택함으로써 지연 시간을 줄이자는데 주안점을 둔다.

Table 1. Read/Write Latency Characteristics of SSDs

Type	Page size	Block size	Read latency	Write latency	Erase latency
SLC	16KB	8MB	65us	380us	500us
MLC_1	16KB	4MB	85us	400us	8,500us
MLC_2	16KB	16MB	78us	1,300us	500us
Ultra-low latency	4KB	256KB	12us	16us	1,500us

3.2 쓰기 전용 캐시(Write-only Cache)

DPW-LRU 등 최근의 FTL 캐시 알고리즘들은 교체 대상 페이지 범위를 설정하고, 이 범위 내에 클린 페이지가 존재하더라도 교체 비용 가중치가 더 낮은 더티 페이지가 있다면 그 더티 페이지를 교체한다. 이처럼 무조건적인 클린 페이지 선택이 아닌 지표에 의한 교체 비용 가중치를 적용하는 이유는 캐시 적중률을 고려하기 때문이다. 이를테면 쓰기 지연 시간이 읽기 지연 시간의 10배 이상인 환경에서 최근 참조 빈도가 1인 더티 페이지와 참조 빈도가 10인 클린 페이지 중 클린 페이지를 선택하여 교체한다면 쓰기 횟수는 감소하겠지만 캐시 적중률은 낮아지고 미래의 참조에 의한 읽기 지연으로 결국 총 지연 시간이 늘어날 수 있다. 즉, FTL 캐시 알고리즘이 지향하는 바는 쓰기 횟수 감소, 적중률 향상, 지연 시간 감소라는 세 가지 복합 목표에 있는데, 이들 세 가지 목표 사이의 상호 배타성 혹은 상호 의존성을 결정론적 관계식으로 표현할 수 없기 때문에 다양한 지표 발굴을 통한 최적화 문제로 접근한다[14, 15].

그런데 Table 2에서 보는 바와 같이 SSD가 HDD에 비해 물리적 지연 시간이 1/50 정도로 짧다는 사실에 주목하면[18], 시간적 성능을 더 향상하기 위한 캐시 적중률 고려의 중요성은 낮아진다. 이런 경우 FTL 캐시의 주목적을 SSD의 최대 단점인 쓰기 및 지우기 횟수 제한에 의한 수명 단축을 완화하기 위한 쓰기 횟수 감소에 두는 것이 타당하다. 캐시에서 가장 지배적인 쓰기 횟수 감소 방안은 캐시 전체를 더티 페이지용으로만 활용하고 클린 페이지용으로는 할당하지 않는 것인데, 이는 곧 캐시에 없는 읽기 페이지는 물리 페이지 읽기 후

Table 2. Read/Write Latency Characteristics of HDDs

RPM	Sectors per track	Average seek time	Average rotational latency	Data transmission latency	Read/Write latency
4,200	63 (512B)	4,000~ 15,000us	7,140us	226us	11,366~22,366us
5,400			5,560us	176us	9,736~20,736us
7,200			4,170us	132us	8,302~19,302us
10,000			3,000us	95us	7,095~18,095us
15,000			2,000us	63us	6,063~17063us

```
ftl_read(page_no, buffer) { // general read-write cache
    if (cache_R_hit(page_no, buffer))
        return;
    read_physical_flash(page_no, buffer);
    cache_insert(page_no, buffer);
}
ftl_read_wo(page_no, buffer) { // write-only cache
    if (cache_R_hit(page_no, buffer))
        return;
    read_physical_flash(page_no, buffer);
}
ftl_write(page_no, buffer) {
    if (cache_W_hit(page_no, buffer))
        return;
    cache_insert(page_no, buffer);
}
```

Fig. 2. FTL Read/Write Procedures using Write-only Cache

캐시에 삽입하지 않음을 의미한다. 이처럼 더티 페이지만을 관리하는 경우를 쓰기 전용 캐시라 정의한다. Fig. 2에 FTL 쓰기 전용 캐시의 프로시저 골격을 보였고, 그 내용은 아래와 같다.

- ftl_read() : 일반적인 캐시 운용상 읽기 프로시저로서 캐시 히트 시 복사하여 돌려주고(cache_R_hit()), 아니면 플래시 물리 페이지를 읽어(read_physical_flash()) 캐시에 삽입한 후(cache_insert()), 읽은 내용을 돌려준다.
- ftl_read_wo() : 쓰기 전용 캐시 운용상 읽기 프로시저로서 캐시 히트 시 복사하여 돌려주고(cache_R_hit()), 아니면 플래시 물리 페이지를 읽은(read_physical_flash()) 내용을 돌려주고, 캐시에는 삽입하지 않는다.
- ftl_write() : 쓰기 프로시저는 일반 캐시나 쓰기 전용 캐시에 관계 없이 캐시 히트 시 해당 캐시에 복사하고, 아니면 새로운 캐시를 할당하여 복사하며, 물리 플래시 페이지쓰기는 하지 않는다.

3.3 쓰기 전제 읽기(Read for Write)

SSD, HDD 등 보조기억장치의 읽기·쓰기는 페이지 혹은 섹터 단위로 이루어지기 때문에 페이지 내 일부분만 수정하려면 일단 기존 페이지를 읽어야 하는데 이는 곧 쓰기를 전제로 하는 읽기를 의미한다. 쓰기 전용 캐시라도 이와 같은 쓰기를 전제로 하여 읽은 페이지는 캐시에 삽입하여 이후 쓰기에 대한 적중을 기대할 수 있겠으나 이 경우 읽은 페이지의 캐시 내 존속 시간을 연장하여 캐시 효율에 악영향을 미칠 뿐 다른 이점이 없다.

4. 쓰기 전용 캐시 실험 및 평가

4.1 실험 환경 및 시뮬레이터

실험을 위한 시뮬레이터를 Fig. 3과 같이 운영체제 입·출력 및 FTL 에뮬레이터 프레임을 갖추고[1], 여기에 LRU, CFLRU, DWP-LRU 등 실험 캐시 알고리즘을 구현하여 그 각각에 일반적인 RW(Read-Write) 캐시와 쓰기 전용인 WO (Write-Only) 캐시의 적용 결과를 비교한다. 이때 플래시 용량은 20GB로, FTL 캐시는 8KB 페이지 16K 개로 각각 설정한다. Fig. 3의 OS IO 스텁은 가상 파일의 저장 상태 관리를 위해 Fig. 4의 자료 구조를 도입하여 SSD에 대한 LBA(Logical Block Address)를 다음과 같이 관리하고, Fig. 5는 그 구현 코드 일부이다.

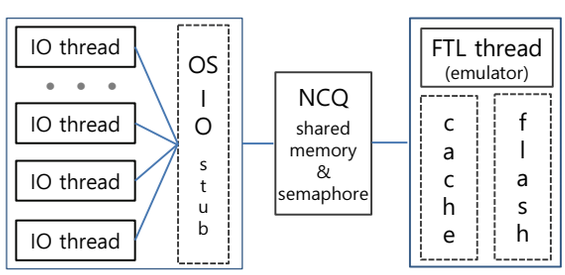


Fig. 3. Block Diagram of the Simulator

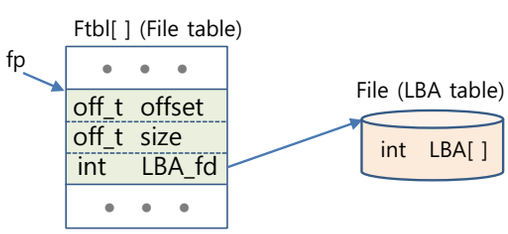


Fig. 4. Data Structure for the OS IO Stub

```
off_t os_read(int  fd, void *buf, off_t size) {
  :
  fp = Ftbl + fd;
  f_lba = fp->offset / F_PGSIZE;          // first LBA(page)
  l_off = fp->offsetf % F_PGSIZE;         // offset in a page
  npage = ((l_off + size) + F_PGSIZE  - 1) / F_PGSIZE;
  l_tbl = (int *)malloc(npage * sizeof(int)); // for LBA table
  lseek(fp->LBA_fd, f_lba * sizeof(int), SEEK_SET);
  read(fp->LBA_fd, l_tbl, npage * sizeof(int))
  for (i = 0, bp = buf; i < npage; i++) {
    ioq_put_os(IOQ_READ, l_tbl[i]);       // call FTL
    csize = size > F_PGSIZE-l_off ? F_PGSIZE-l_off : size;
    l_off = 0; bp += csize; size -= csize; fp->offset += csize;
  }
  free(l_tbl);
  :
}
```

Fig. 5. A Sample Code of the OS Stub for the Simulator

- 입력을 위해 os_read(int fd, void *buf, off_t size), 출력을 위해 os_write(int fd, void *buf, off_t size) 프로시저를 제공한다.
- 파일 생성 시 파일 테이블을 할당하여 입·출력 위치, 파일 크기, 그리고 LBA 테이블을 유지하고, LBA 테이블은 메모리가 아닌 파일에 둔다.
- 입·출력 시 해당 LBA 및 LBA 내 입·출력 위치는LBA [fp->offset/F_PGSIZE], fp->offset%F_PGSIZE 계산식으로 각각 구한다.
- FTL에는 입·출력 대상인 LBA만 전달하고, 실제 데이터는 주거나 받지 않는다.

4.2 실험 데이터 및 워크로드

실험을 위한 입·출력 데이터 및 워크로드는 [10]에서 사용한 내용을 따라 설계한다. 입·출력 데이터로는 [14, 15] 등 SSD 연구에 자주 사용되는 Table 3의 금융 현장 트레이스 데이터[19]를 사용하고, 이들 데이터를 입·출력하는 워크로드 스레드는 Table 4와 같이 구성한다.

Table 3. Field Trace Data for the Experiment

Name	Max LBA	Number of I/O
Financial1	1,215,633	5,334,984
Financial2	1,215,222	3,699,194

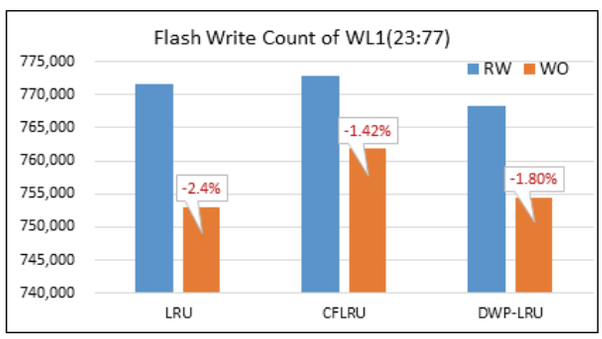
Table 4. Definition of Workloads for the Experiment

Name	Combination of Data	I/O Ratio(%)
WL1	Financial1	23:77
WL2	Financial1 + Financial2	63:37
WL3	Financial2	82:18

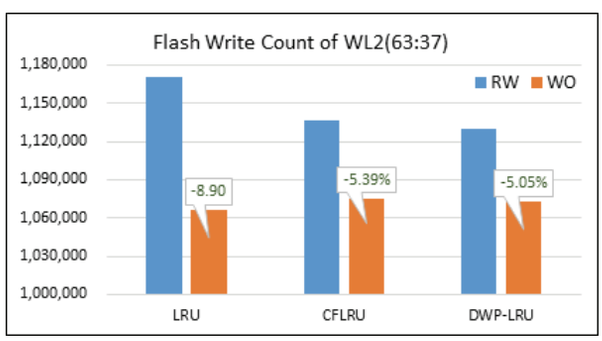
4.3 실험 결과 및 분석

1) 쓰기 횟수 관점

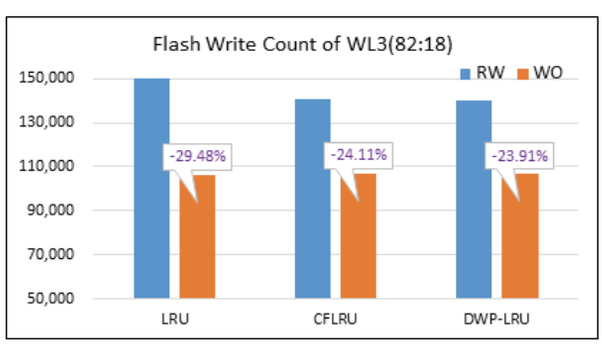
Fig. 6은 입·출력 비율이 서로 다른 각 워크로드별로 LRU, CFLRU, DPW-LRU 등 세 알고리즘에 적용한 일반 캐시인 “RW” 대비 쓰기 전용 캐시인 “WO”의 효과이다. 이 그림으로부터 아래 사항들을 유추할 수 있다.



(a) Write Count Reduction of the WL1



(b) Write Count Reduction of the WL2



(c) Write Count Reduction of WL3

Fig. 6. Effectiveness of the Write-only Cache

- 읽기 비율이 높은 워크로드일수록 쓰기 횟수 감소 효과가 크다. 이는 높은 비율을 차지했던 읽기용 캐시를 쓰기용으로 전용함으로써 쓰기에 의한 페이지 교체 횟수가 감소하기 때문이다.
- 반대로 쓰기 비율이 높은 워크로드에서는 상대적으로 읽기용 캐시가 적을 것이므로 이를 쓰기용으로 전용했을 때의 효과도 적게 나타난다.

Table 5. Comparison of the Hit Ratio and Total Latency(LRU)

Metrics \ Workload / Cache		WL1		WL3	
		RW	WO	RW	WO
Read hit ratio(%)		91	87	93	69
Flash read count		603,970	902,434	295,726	1,406,799
Flash read latency(ms)	SLC	39,258	58,658	19,222	91,441
	MLC_2	47,109	80,389	23,066	109,730
Write hit ratio(%)		96	87	95	87
Flash write count		771,568	753,075	150,416	106,066
Flash write latency(ms)	SLC	293,195	286,168	57,158	40,306
	MLC_2	1,003,038	978,997	195,540	137,885
Total latency(ms)	SLC	332,453	344,826	76,380	131,747
	MLC_2	1,050,147	1,059,386	218,606	247,615

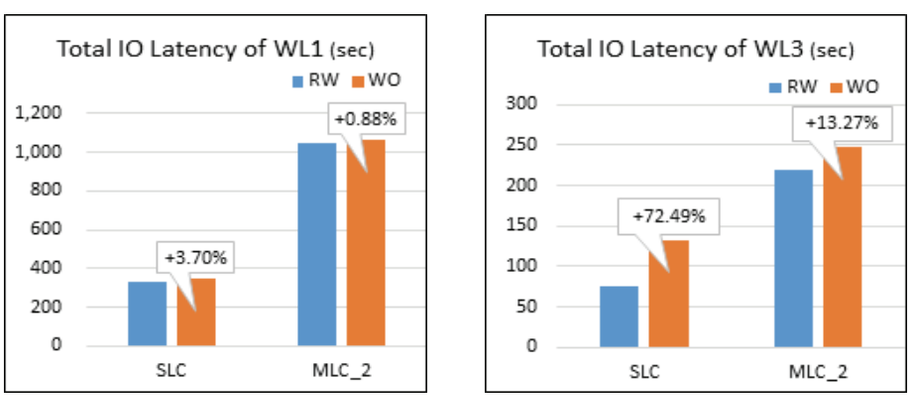


Fig. 7. Latency Burden of the Write-only Cache

- 쓰기 전용 캐시를 적용할 경우 알고리즘 자체의 진화는 도움이 되지 않고 원래의 LRU가 가장 우수하다. 즉, 쓰기 감소 효과 측면에서는 모든 경우에 있어서 복잡한 알고리즘 개선에 의한 효과보다 쓰기 전용 캐시에 의한 효과가 더 높다.

2) 입·출력 지연 시간 관점

플래시 쓰기 횟수 감소는 읽기 횟수의 증가를 동반하므로 전체적인 지연 시간 부담 추이를 살펴볼 필요가 있다. Table 5와 Fig. 7은 Table 1의 SLC와 MLC_2 SSD에 LRU 알고리즘으로 입·출력 비율의 차이가 뚜렷한 워크로드 WL1과 WL3를 가했을 때 일반 캐시와 쓰기 전용 캐시의 의 총 지연 시간 추이를 보인다. 이 표와 그림으로부터 아래 사항들을 유추할 수 있다.

- 쓰기 비율이 높은 WL1의 경우 지연 시간이 긴 쓰기의 횟수 감소가 지연 시간이 짧고 횟수가 증가한 읽기 지연 시간을 상쇄하는 비율이 쓰기 비율이 낮은 WL3보다 높다.
- 읽기 지연 시간과 쓰기 지연 시간의 상대적 차이가 큰 MLC_2의 경우 쓰기 횟수 감소에 의한 총 지연 시간 증가가 읽기 지연 시간과 쓰기 지연 시간의 상대적 차이가 작은 SLC보다 적다.
- 쓰기 비율이 높고, 읽기 지연 시간과 쓰기 지연 시간의 상대적 차이가 큰 WL1/MLC_2의 경우 쓰기 전용 캐시에 의한 총 지연 시간 증가가 0.88%에 그친다.

5. 결 론

HDD 대비 SSD의 최대 장점은 접근 속도가 빠르다는 것이고, 최대 단점은 제자리 덮어쓰기가 불가능하며 이로 인해 요구되는 지우기의 횟수가 제한되어 있다는 점이다. 따라서 SSD 내부의 FTL 캐시는 적중률 제고에 의한 시간적 성능 향상보다는 쓰기 횟수 감소에 더 중점을 둔다. 이 논문에서는 약간의 접근 지연 시간 부담을 전제하고 쓰기 횟수를 더 줄일 수 있는 쓰기 전용 캐시 기반 LRU 방안을 실험 및 분석하

였고. 실험 계획의 모든 경우에서 알고리즘 자체의 진화보다는 쓰기 전용 캐시에 의한 쓰기 횟수 감소 효과가 더 큰 것으로 확인되었다. 쓰기 전용 캐시의 쓰기 횟수 감소 효과는 읽기 비중이 높은 워코로드에서 높고, 접근 지연 시간의 부담은 쓰기 빈도가 높은 워크로드에서 쓰기 지연 시간이 긴 SSD에서 낮은 것으로 나타났다. 따라서, 전체적인 접근 지연 시간이 크게 강조되지 않는 읽기 위주 환경이나 쓰기 위주 워크로드 및 쓰기 지연 시간이 긴 대용량 MLC SSD 환경에서는 메모리 소요량이 적고 간단한 기본 LRU 알고리즘에 쓰기 전용 캐시를 적용하는 것이 바람직하다. 향후 읽기/쓰기 비율의 변화에 따른 쓰기 전용 캐시의 적응적 활동/정지 방안을 도입한다면 쓰기 전용 캐시의 활용 범위는 더 넓어질 것으로 기대된다.

References

[1] H. B. Lee and T. Y. Chung, "A comparative analysis on page caching strategies affecting energy consumption in the NAND flash translation layer," *IEMEK Journal of Embedded Systems and Applications*, Vol.13, No.3, pp.109-115, 2018.

[2] G. Zhu, J. H. Han, and Y. S. Son, "A preliminary study: Towards parallel garbage collection for NAND flash-based SSDs," *IEEE Access*, Vol.8, pp.223574-223587, 2020.

[3] T. S. Chung, D. J. Park, S. W. Park, D. H. Lee, S. W. Lee, and H. J. Song, "A survey of Flash Translation Layer," *Journal of Systems Architecture*, Vol.55, No.5-6, pp.332-343, 2009.

[4] C. Matsui, A. Arakawa, C. Sun, and K. Takeuchi, "Write order-based garbage collection scheme for an LBA scrambler integrated SSD," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, Vol.25, No.2, pp.510-519, 2017.

[5] Y. Deng and J. Zhou, "Architectures and optimization methods of flash memory based storage systems," *Journal of Systems Architecture*, Vol.57, No.2, pp.214-227, 2011.

[6] Samsung SSD_870_EVO [internet], https://semiconductor.samsung.com/resources/datasheet/Samsung_SSD_870_EVO_Data_Sheet_ Rev1.1.pdf, 2021.

[7] A. Eisenman et al., "Flashield: A hybrid key-value cache that controls flash write amplification," in *Proceedings of 16th USENIX Symposium on Networked Systems Design and Implementation(NSDI '19)*, pp.65-78, 2019.

[8] J. H. Kim, *Computer Architecture-5th edition*, Life and Power Press, pp. 284-288, 2019.

[9] S. Y. Park, D. W. Jung, J. U. Kang, J. S. Kim, and J. W. Lee, "CFLRU: A replacement algorithm for flash memory," in *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems(CASES'06)*, pp.234-241, 2006.

[10] H. B. Lee and T. Y. Chung, "Analysis and improvement of the DPW-LRU cache replacement algorithm for flash translation layer," *IEMEK Journal of Embedded Systems and Applications,* Vol.15, No.6, pp.289-297, 2020.

[11] H. Y. Jung, H. K. Shim, S. M. Park, S. Y. Kang, and J. H. Cha, "LRU-WSR: Integration of LRU and writes sequence reordering for flash memory," *IEEE Transactions on Consumer Electronics*, Vol.54, No.3, pp.1215-1223, 2008.

[12] Z. Li, P. Jin, X. Su, K. Cui, and L. Yue, "CCF-LRU: A new buffer replacement algorithm for flash memory," *IEEE Transactions on Consumer Electronics*, Vol.55, No.3, pp.1351-1359, 2009.

[13] P. Jin, Y. Ou, T. Härder, and Z. Li, "AD-LRU: An efficient buffer replacement algorithm for ash-based databases," *Data Knowledge Engineering*, Vol.72, pp.83-10, 2012.

[14] Y. Yuan, J. Zhang, G. Han, G. Jia, L. Yan, and W. Li, "DPW-LRU: An efficient buffer management policy based on dynamic page weight for flash memory in cyber-physical system," *IEEE Access (Special Section on Distributed Computing Infrastructure for Cyber-Physical Systems)*, Vol.7, pp.58810-58821, 2019.

[15] W. H. Lee and J. H. Kwak, "2WPR: Disk buffer replacement algorithm based on the probability of reference to reduce the number of writes in flash memory," *Journal of the Korea Society of Computer and Information*, Vol.25, No.2, pp. 1-10, 2020.

[16] S. Wu, C. Du, H. Li, H. Jiang, Z. Shen, and B. Mao, "CAGC: A content-aware garbage collection scheme for ultra-low latency flash-based SSDs," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium(IPDPS 2021)*, pp.162-171, 2021.

[17] S. Li, W. Tong, J. Liu, B. Wu, and Y. Feng, "Accelerating garbage collection for 3D MLC flash memory with SLC blocks," in *Proceedings of the International Conference on Computer-Aided Design(ICCAD 2019)*, pp.162-171, 2021.

[18] Wikipedia, Hard disk drive performance characteristics [Internet], https://en.wikipedia.org/wiki/Hard_disk_drive_ performance_characteristics.

[19] UMass Trace Repository [Internet], http://traces.cs.umass.edu/index.php/Storage.

이 형 봉

https://orcid.org/0000-0003-4944-5265
e-mail : hblee@gwnu.ac.kr
1984년 서울대학교 계산통계학과(학사)
1986년 서울대학교 계산통계학과(석사)
2002년 강원대학교 컴퓨터과학과(박사)
1986년 ~ 1993년 LG전자 컴퓨터연구소 선임연구원
1994년 ~ 1999년 한국디지털(주) 책임컨설턴트
2004년 ~ 현 재 강릉원주대학교 컴퓨터공학과 교수
관심분야 : 무선 통신(Wireless Networks), 센서 네트워크(Sensor Networks), 임베디드 시스템(Embedded Systems), 사물 인터넷(IoT)

정 태 윤

https://orcid.org/0000-0002-3445-468X
e-mail : tychung@gwnu.ac.kr
1987년 연세대학교 전자컴퓨터공학부(학사)
1989년 연세대학교 전자컴퓨터공학부(석사)
2000년 연세대학교 전자컴퓨터공학부(박사)
1989년 ~ 2001년 삼성전자 고등기술원 연구원
2001년 ~ 2001년 국제 DVD_Forum 부의장
2001년 ~ 현 재 강릉원주대학교 전자공학과 교수
2004년 ~ 현 재 강원 ICT 융합연구원 원장
2022년 ~ 현 재 강릉원주대학교 교학부총장
관심분야 : 센서 네트워크(Sensor Networks), 임베디드 시스템(Embedded Systems), 이미지 신호처리(Image Signal Processing), 디지털 비디오 인코딩(Digital Video Encoding)