논문 2018-13-13

NAND 플래시 변환 계층에서 전력 소모에 영향을 미치는 페이지 캐싱 전략의 비교・분석

(A Comparative Analysis on Page Caching Strategies Affecting Energy Consumption in the NAND Flash Translation Layer)

이 형 봉*, 정 태 윤
(Hyung-Bong Lee, Tae-Yun Chung)

Abstract : SSDs that are not allowed in-place update within the allocated page cause another allocation of a new page that will replace the previous page at the moment data modification occurs. This intrinsic characteristic of SSDs requires many changes to the existing HDD-based IO theory. In this paper, we conduct a performance comparison of FTL caching strategy in perspective of cache hashing (Global vs. grouped) and caching algorithm (LRU vs. NUR) through a simulation. Experimental results show that in terms of energy consumption for flash operation the grouped management of cache is not suitable and NUR algorithm is superior to LRU algorithm. In particular, we found that the cache hit ratio of LRU algorithm is about 10% point higher than that of NUR algorithm while the energy consumption of LRU algorithm is about 32% high.

Keywords : SSD, NAND Flash, FTL, Buffer cache, LRU, NUR

Ⅰ. 서 론

SSD (Solid State Disk)는 원래 고체 상태의 반도체 디스크라는 의미로서, RAM을 디스크 용도로 활용한다면 이 또한 SSD라 말할 수 있다. 그러나 보통 디스크라 하면 전원이 없는 상태에서도 기록 내용을 보존하는 ROM 성격의 저장 매체를 의미하므로 일반적으로 SSD는 플래시 메모리 기반 반도체 디스크를, 그 중에서도 NAND 플래시 메모리 기반 반도체 디스크를 지칭한다 [1]. NAND 플래시 메모리 (이하 플래시라 칭함)의 비트 소자는 그림 1과 같이 크게 콘트롤 게이트, 부유 게이트 (Floating gate), 그리고 트랜지스터 등 세 부분으로 구성되는데, 부유 게이트에 전자를 채워 트래지스터 회로를 차단함으로써 비트 0을, 반대로 전자를 비워 트랜지스터 회로를 개방함으로써 비트 1을 식별한다. 그림 1에서 G (Gate)에 높은 전압을 인가하면 P 반도체의 전자가 차단막을 뚫고 부유 게이트로 이동하여 갇히고, B (Back gate)에 높은 전압을 인가하면 반대로 부유 게이트 전자가 P 반도체로 이동하여 부유 게이트를 비운다. 부유 게이트 내 전자의 유・무 상태는 G에 낮은 전압을 인가하여 S (Source)에서 D (Drain)로의 전류 e~가 흐르는지에 따라 결정된다. 이 때, 부유 게이트에 전자 채우기, 전자 지우기, 전자 존재 여부 체크하기 등 세 가지 연산을 위한 전력소모는 서로 다르다 [2].

전자를 채우는 일과 비우는 일은 각각 페이지와 블록으로 불리는 일정 크기 단위 (블록 크기 = 페이지 크기 * n)별로 가능하고, 한 번 채워진 전자는 수 년간 유지될 수 있다. 전자를 채우거나 비우는 작업이 일정 크기 단위별로 이루어지는 이유는 집적도를 높이기 위해 각 소자들을 독립시키지 않고 그룹으로 묶어 연결했기 때문이다. 즉, 특정 소자 그룹에 접근하기 위해서는 해당되는 모든 소자들의

*Corresponding Author (tychung@gwnu.ac.kr)
Received: Apr. 30 2018, Revised: May 24 2018, Accepted: June 4 2018.
H.B. Lee, T.Y. Chung: Gangneung-Wonju National University
※ 본 연구는 산업통상자원부의 “지역산업거점기관지원사업”의 지원을 받아 수행된 결과임 (R0006229, 2018).

 2018 June 13(3) 109-116
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2018.13.3.109

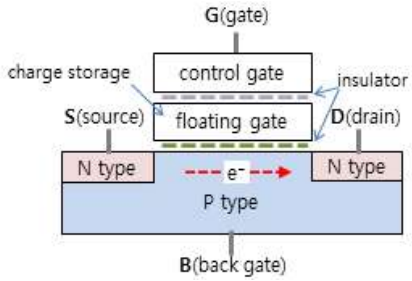

그림 1. NAND 플래시 메모리 비트 소자
Fig. 1 Bit cell of NAND flash memory

활성 라인 (WL: Word Line)이 선택되었을 때만 가능하도록 설계하고, 따라서 읽기 또한 정해진 그룹별로만 가능하다. 이를 위해서는 소자들의 활성 라인 중 하나라도 선택되지 않으면 그룹 또한 선택되지 않도록 NAND 논리로 연결해야 한다 [2].

이 논문에서는 플래시 고유 특성에 따라 SSD 내부에 제공되는 캐시 운용 방법들을 전력 소모 관점에서 비교・분석하여 가장 적합한 캐시 전략을 권고한다. 이를 위해 2 장에서 SSD의 문제점 및 해결 방안을 고찰하고, 3 장에서 FTL 캐시 실험 계획을 설계하며, 4 장에서 실험 결과를 분석한다. 그리고 마지막 5 장의 결론으로 맺는다.

II. SSD의 문제점 및 해결 방안 고찰

1. SSD의 특징 및 문제점

HDD 대비 SSD의 최대 장점은 접근 속도가 빠르다는 데 있다. 표 1은 RAM 접근 시간을 기준으로 했을 때 SSD와 HDD의 접근 시간이 개략적으로 몇 배인가를 요약해보이고 있는데 [3-5], SSD가 HDD보다 약 500배 이상 빠르다. 표 1에서 SLC (Single Level Cell), MLC (Multiple Level Cell, TLC (Triple Level Cell)는 부유 게이트 내 전자양에 따라 각각 1 비트, 2 비트, 3 비트를 식별할 수 있음을 의미한다. 그 외에도 가볍고 부피가 작으며 충격에 강하다는 장점도 있다. 이러한 장점을 기반으로 SSD를 HDD에 대한 캐시로 활용하려는 연구가 활발하다. [6]은 SSD의 랜덤 쓰기 단점을 순차 쓰기로 변환하여 SSD의 빠른 접근 속도를 유지함으로써 랜덤 쓰기 작업이 비교적 많은 IoT 게이트웨이를 위한 가상화 저장 장치의 HDD에 대한 캐시 적용 방안을 제시하였고, [7]은 EXT3 등 일반적인 파일 시스템에서 HDD의 블럭을 캐시 용도인 SSD 블럭으로의 효과적인 대응 방안을 다루고 있다.

표 1. 저장 매체별 접근 시간 비교
Table 1. Comparison of access time by storage media

Opera-tion	RAM (ref.)	SSD (*times*)			HDD (*times*)
		SLC	MLC	TLC	
read	0.06 (*us*)	500	1,000	2,000	100,000
write		5,000	20,000	30,000	100,000
erase	-	30,000	60,000	100,000	-
seek	-	-	-	-	150,000

그러나 SSD는 현재 저장된 데이터를 그 위치에 즉시 다시 쓸 수 없다는 큰 단점이 있다. 서론에서 언급한 바와 같이 플래시 메모리는 일정한 크기 단위별로 접근하기 때문에 어떤 비트 소자는 전자를 채우고, 다른 어떤 비트 소자는 전자를 비우는 등의 서로 다른 일을 한 번의 조작으로 동시에 할 수 없다. 따라서 페이지 별 데이터 쓰기 연산은 초기 상태를 바탕으로 수정이 필요한 비트 소자 모두에 대하여 전자 채우기나 지우기 등 둘 중 한 가지를 선택해서 일괄 적용해야 한다. 이를 위해 플래시 쓰기 연산은 초기 상태를 '1' (부유 게이트가 비어있는 상태)로 간주하고, 데이터가 '1'이 아닌 비트 소자들에게만 전자를 한꺼번에 채워 '0'으로 설정한다. 만약 초기 상태가 아닌 페이지라면 '0'으로 전제되었던 부분에 '1'이 그대로 존재하여 원하는 데이터가 되지 못한다. 이런 이유로 현재 어느 페이지에 저장된 데이터에 변경이 일어나면 그 페이지에 곧바로 갱신된 데이터를 다시 쓰는 이른바 덮어쓰기를 할 수 없다는 제한이 따르고, 이는 플래시의 효과적인 활용을 매우 복잡하게 만든다 [1-7].

2. FTL (Flash Translation Layer)

HDD, SSD 등 2차 대용량 저장 장치를 사용하는 운영체제나 어플리케이션은 디스크 저장 공간의 물리적 페이지에 논리적 페이지 번호를 부여하고 할당한 페이지 번호 쌍을 해당 파일 시스템 정보에 관리한다. HDD의 경우 논리적 페이지 번호와 디스크 내 물리적 페이지 번호의 대응이 헤더 개수, 실린더 개수, 트랙 당 페이지 (섹터) 개수를 적용한 간단한 해싱 함수로 1:1 대응되고 이 대응은 디스크와 파일 시스템이 공유하며 변하지 않는다. 그러나 SSD의 경우 데이터 수정이 있을 때마다 새로운 물리 페이지를 할당해야 하므로 SSD 내부의 FTL (플래시 변환 계층)에서 논리적 페이지 번호에 대한 물리적 페이지 대응을 위한 실시간 맵 테이블 관리가 필요하다. 이를 위한 가장 기본적인 방법은 페이지 단위의 맵 테이블 운영인데, 이 경우 맵 테이블

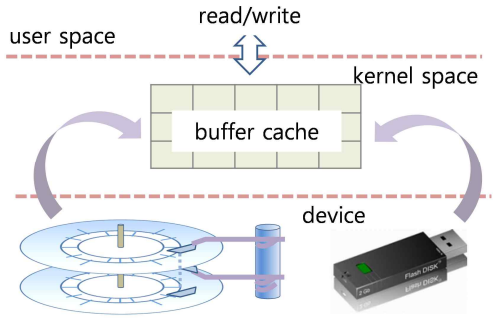

그림 2. 운영체제의 디스크 버퍼 캐시
Fig. 2 Disk buffer cache of operating systems

을 위한 공간이 크게 소요되는 문제가 대두된다. 삼성 SSD 850 PRO 256GB [8]의 경우 총 32M개의 맵 엔트리가 필요하고, 엔트리 당 4 바이트 (25 비트 페이지 번호 범위)를 차지하므로 총 128 MB가 요구된다. 이를 완화시키기 위해 [9]는 매핑 테이블 자체에 대한 효율적인 캐시 적용 방안을 제안한다. 페이지 맵 자체를 줄이는 방법으로 여러 페이지의 묶음인 블록 단위 매핑 테이블을 고려할 수 있는데 이는 블록 매핑 후 어느 하나의 대응 페이지가 수정되면 블록 전체를 이동시켜야 하는 단점이 있어 블록 매핑과 페이지 매핑을 혼용하는 하이브리드 혹은 로그 기반 매핑 방법들이 제안되었다 [10, 11]. FTL의 두 번째 기능은 가비지 (Garbage) 수집이다. SSD의 덮어쓰기 불가능으로 인해 이전의 데이터를 저장했던 페이지들은 효과적인 방법으로 수집·검색된 후 초기화 (지우기)를 거쳐 재할당되어야 한다. 가비지 수집은 SSD의 쓰기 횟수 (Wear level) 제한과 결부되어 쓰기 횟수를 SSD 전체적으로 고르게 분포하도록 유도할 수 있는 정책이 필요하다 [12, 13].

FTL의 세 번째 기능으로 디스크 내부에서의 페이지 버퍼 캐싱을 둘 수 있다. 이 영역에서의 연구는 주로 운영체제 파일시스템 수준에서 SSD 특성을 반영하는 캐싱 기법에 치우쳐 있는데, 저장 데이터를 변경이 잦은 동적 (Hot) 데이터와 그렇지 않은 정적 (Cold) 데이터 등 두 가지로 유형으로 분류하여 가급적 유형별로 동일 플래시 블록에 저장함으로써 궁극적으로 사용 페이지와 가비지 페이지가 섞이지 않도록 유도한다. 이 작업의 성과는 정적 데이터와 동적 데이터의 분류 정확성에 따라 좌우되므로 그 분류 방법론에 관한 연구가 주류를 이룬다 [14]. FTL 내에서의 플래시 페이지 캐싱 전략에 대한 연구로 데이터의 크기와 확률에 따라 캐시를

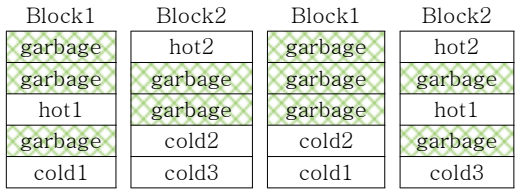

그림 3. 플래시 페이지 저장 패턴
Fig. 3 Pattern of writing flash pages

우회하여 곧바로 저장하거나 캐시를 거치도록 하는 방법이 있는데 [15], 그 근거가 충분하지 못하고 캐시 자체에 대한 운영전략은 다루지 않는다. 이 논문에서는 SSD의 NCQ (Native Command Queue)를 통해 전달된 입·출력 페이지에 대한 몇 가지 캐싱 전략을 비교·분석하여 에너지 소모 관점에서 가장 우수한 방법을 선별한다.

Ⅲ. FTL 페이지 캐싱 전략

1. 디스크 버퍼 캐시

버퍼 (Buffer)의 일반적인 개념은 그림 2와 같이 입·출력 단위나 속도의 차이를 극복하기 위해 데이터를 일시적으로 보관하는 장소이고, 캐시 (Cache)는 속도가 느린 저장장치에 대한 빠른 접근을 지원하기 위해 그 복사본을 저장해 두는 고속의 장소이다 [16]. 디스크 버퍼 캐시는 일반적으로 운영체제에 의해 HDD 등 2차 저장 장치를 위해 관리되고, 캐시가 부족할 경우 FIFO (Firts In First Out), LRU (Least Recentlly Used), NUR (Not Used Recently), MFU (Most Frequently Used) 등의 알고리즘에 의해 희생 캐시를 선택한다.

2. FTL 페이지 캐시의 평가 지표

일반적인 운영체제 수준에서 논의되는 HDD 캐시 교체 알고리즘의 성능은 캐시 그 자체와 관련된 평가 지표로 충분하다. 대표적인 평가 지표로 적중률 (Hit ratio)을 들 수 있는데, 일정 기간 동안 가장 높은 적중률을 보인 알고리즘을 우수하게 평가한다. 그 이유는 덮어쓰기를 당연시 하는 HDD의 경우 캐시 아래에서의 변수 요인이 거의 없기 때문이다. 그런데, SSD의 경우 캐시 자체도 중요하지만 플래시에 저장될 때마다 새로운 페이지가 할당되어 가비지가 생성되므로 캐시 교체 알고리즘에 따라 달라지는 페이지 저장 패턴에 의해 2차적으로 가비지 수집에 영향을 미칠 수 있다. 이를테면 그림 3의 왼쪽과 오른쪽 두 저장 패턴에서 hot1, hot2

표 2. NAND 플래시 에너지 소모
Table 2. NAND flash energy consumption

Read (ref.)	Write (times)	Erase (times)
0.5㎶ (1)	7.5㎶ (14)	40㎶ (80)

표 3. 실험 알고리즘 분류
Table 3. Classification of experimental algorithm

Algorithm type		Algorithm name
LRU	global	LB
	grouped	LG
NUR	global	NB
	grouped	NG

페이지에 의해 관련 블록이 가비지 수집 대상이 된다고 하면, 왼쪽 저장 패턴에서는 Block1, Block2 모두 수집 대상이 되어 cold1 ~ cold3 페이지에 대한 복사가 필요하지만, 오른쪽 저장 패턴에서는 Block2만 수집 대상이 되어 cold3 페이지만 복사하면 된다. 즉, 왼쪽 저장 패턴이 오른쪽 저장 패턴보다 가비지 수집 비용이 더 크다. 따라서 단순 적중률은 낮지만 그림 3의 오른쪽 패턴 저장 빈도가 높아 전체적으로 더 나은 캐싱 전략이 있는지를 관찰해볼 필요가 있다. 이를 위해 읽기·쓰기·지우기 연산 각각에 표 2의 에너지 소모 가중치 [2]를 부여하고, 동일한 파일 처리 어플리케이션에 대한 페이지 캐싱 전략들의 플래시 페이지 입·출력 총량을 소모 에너지로 변환하여 성능 평가 지표로 활용한다.

3. FTL 페이지 캐싱 전략 분류

이 연구에서는 대부분의 FTL 관련 실험에서 묵시적으로 사용되는 LRU와 캐시 규모가 큰 운영체제 페이지 교체에 사용되는 NUR 알고리즘 중심으로 실험한다.

■ 희생 캐시 선택 알고리즘

LRU는 가장 오랫동안 참조되지 않은 페이지 캐시를 희생 캐시로 선택하는 알고리즘으로, 이중 연결 리스트에 의한 스택 개념으로 구현한다. 즉, 참조된 페이지 캐시를 해당 리스트의 꼬리로 이동시켜, 결국 머리에는 가장 오래된 페이지 캐시가 위치하도록 한다. NUR은 참조 카운트와 불결 (Dirty) 비트를 두고 청결 (Clean)하고 참조 카운트가 가장 적은 것을 최 우선 희생 캐시로 선택하는 알고리즘으로 배열 및 선형 탐색으로 구현한다.

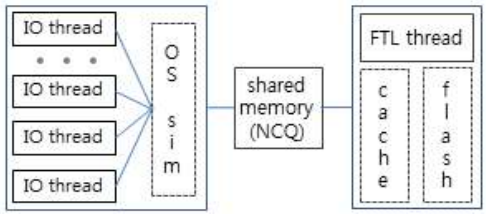

그림 4. 구현 시뮬레이터 구조
Fig. 4 Architecture of implemented simulator

■ 전역 캐시와 그룹 캐시

캐시는 TLB (Translation Look-aside Buffer)와 같은 하드웨어 지원을 받지 않는 한 선형 탐색이 불가피하다. 따라서 캐시를 페이지 번호 등에 의한 간단한 해시 그룹으로 관리함으로써 검색 부담을 줄일 수 있다. 그러나 희생 캐시 선택의 효과를 높이기 위해 캐시를 그룹핑하지 않는 전역 캐시로 운영할 수도 있다.

■ 실험 캐싱 알고리즘 분류

실험을 위해 LRU, NUR 각각에 대하여 전역 캐시와 그룹 캐시 두 가지를 적용하여 표 3과 같이 명명한다.

Ⅳ. 실험 및 결과 분석

1. 실험 방법 및 환경

■ 시뮬레이터

SSD를 위한 시뮬레이터는 FlashSim [17]을 참조하여 그림 4의 구조로 구현하고 운영체제 파일 입·출력 API로 open(), read(), write(), lseek(), close()를 두었다.

■ FTL (플래시, 가비지 수집, 캐시)

실험 환경 모델 플래시로 캐시 RAM이 256MB인 삼성 SSD 850 PRO 256GB [12]을 목표로 하되 다만 페이지 할당 용량을 2GB 이내로 한정한다. 캐시 RAM 256MB는 섹터 (페이지) 맵 테이블에 128MB (=32K pages * 4 bytes), 나머지 128MB는 캐시에 할당한다. 이에 따라 캐시는 총 16,384개 (=128MB/8KB)이고 이는 플래시 2GB의 전체 페이지 262,144 (=2GB/8KB)개의 약 6%에 해당된다. 가비지는 그림 3의 페이지 저장 패턴에 의한 특성이 가급적 여과 없이 부각될 수 있도록 대상 블록을 랜덤하게 선정하여 수집한다.

■ 입·출력 워크로드

입·출력 워크로드 (이하 WL 표기 혼용)를 아래 같이 다섯 유형으로 분류하고, 이들을 표 4와 같이

표 4. 파일 입·출력 어플리케이션 유형
Table 4. Type file I/O application

Type	Work load and number of thread
AE1	WL1~WL2: 15 threads for each WL
AE2	WL3~WL5: 10 threads for each WL
AE3	WL1~WL5: 6 threads for each WL

표 5. 실험 환경
Table 5. Experimental environment

Items	Specification or model
CPU	ZeonW3860, 6core
Memory	32GB
OS	UNIX (MacOS, Darwin 14.5.0)
Language	C and POSIX thread

세 가지로 조합한 어플리케이션 환경 (이하 AE 표기 혼용, 파일 용량이 플래시 용량 2GB 중 약 1.7GB (80~90%) 차지)에 대한 실험 결과를 분석하였다.

- WL1 : 1,000~9,000 바이트를 10,000번 순차 입력하는 과정을 5번 반복
- WL2 : 1,000~9,000 바이트를 10,000번 순차 출력하는 과정을 5번 반복
- WL3 : 1,000~9,000 바이트를 10,000번 랜덤 입력하는 과정을 5번 반복
- WL4 : 1,000~9,000 바이트를 10,000번 랜덤 출력하는 과정을 5번 반복
- WL5 : 1,000~9,000 바이트를 10,000번 랜덤 입력 후 동일 위치에 출력하는 과정을 5번 반복

■ 실험 시스템 환경

시뮬레이터 구현 및 실험은 표 5와 같이 PC 플랫폼 상의 UNIX 환경에서 POSIX 스레드로 이루어졌다.

2. 실험 결과 및 분석

■ 캐시 관리 관점 (전역 캐시, 그룹 캐시)

표 6에 캐시 관리 방법별 캐시 적중률 추이를 보였는데, 뚜렷하게 우수한 방법론은 드러나지 않고 워크 로드에 따라 다소 차이가 있다. 여기서 HR은 읽기 적중 회수, HW는 쓰기 적중 회수, FR은 플래시 읽기 회수, FW는 플래시 쓰기 회수, H (%)는 적중률을 각각 의미한다.

■ LRU와 NUR 알고리즘 관점

그림 5에 LRU와 NUR 간의 적중률 추이를 보였다.

표 6. 캐시 관리 방법별 적중률
Table 6. Hit ratio by cache management method

LRU		HR	HW	FR	FW	H(%)
AE1	LB	1,527,956	1,201,890	886,129	186,916	71.7
	LG	1,483,862	1,192,982	930,223	216,873	70.0
AE2	LB	1,384,146	1,147,505	926,746	82,757	71.4
	LG	1,298,390	1,121,273	1,012,502	213,532	66.3
AE3	LB	1,454,567	1,190,408	899,071	96,037	72.6
	LG	1,365,083	1,169,488	988,555	213,620	67.8

NUR		HR	HW	FR	FW	H%
AE1	NB	1,118,605	997,515	1,295,480	80,723	60.5
	NG	1,483,314	1,154,984	930,771	162,101	70.7
AE2	NB	1,076,275	1,012,997	1,234,617	82,752	61.3
	NG	1,043,598	1,039,716	1,267,294	225,689	58.2
AE3	NB	1,131,502	987,544	1,222,136	82,041	61.9
	NG	1,221,452	1,108,713	1,132,186	204,904	63.5

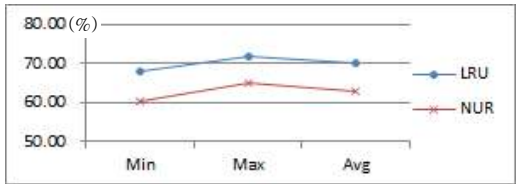

그림 5. LRU와 NUR 알고리즘의 적중률 비교
Fig. 5 Hit ratio comparison between LRU and NUR algorithms

여기서 Min은 AE1, AE2, AE3 각 분야에서 최저 적중률들의 평균, Max는 최대 적중률들의 평균, Avg는 AE 구분없는 적중률 평균을 의미한다. 그림 5로부터 LRU 알고리즘이 약 10% 포인트 더 우수함을 알 수 있다.

■ LRU와 NUR 알고리즘 관점

그림 5에 LRU와 NUR 간의 적중률 추이를 보였다. 여기서 Min은 AE1, AE2, AE3 각 분야에서 최저 적중률들의 평균, Max는 최대 적중률들의 평균, Avg는 AE 구분없는 적중률 평균을 의미한다. 그림 5로부터 LRU 알고리즘이 약 10% 포인트 더 우수함을 알 수 있다.

■ 총 에너지 소모 관점

표 7은 가비지 수집을 위해 수행된 플래시 읽기 회수 (GR), 플래시 쓰기 회수 (GW), 플래시 지우기 회수 (GE), 그리고 표 6의 FR 및 FW와 가비지 수집 비용 GR, GW, GE를 모두 포함한 총 에너지 소모 추이 (TE (J))를 보이고 있다. 여기서 총 소모 에너지 TE (J)에는 표 2의 연산별 에너지 소모를 적용했으며 단위는 주울 (J)이다. 표 7의 일차적 시사점은

표 7. 캐시 관리 방법별 에너지 소모
Table 7. Energy consumption by cache management method

LRU		GR	GW	GE	TE(J)
AE1	LB	3,332,592	340,641	1,751	4.29
	LG	94,132,586	2,287,661	9,473	57.93
AE2	LB	25,873,870	2,247,188	8,791	23.07
	LG	639,702,544	15,911,286	62,678	387.36
AE3	LB	19,095,801	1,697,856	6,697	17.44
	LG	451,562,749	11,991,270	47,365	276.99

NUR		GR	GW	GE	TE(J)
AE1	NB	2,457,080	612,352	2,397	4.74
	NG	47,515,671	1,332,108	5,526	30.42
AE2	NB	11,212,915	1,953,925	7,646	14.68
	NG	817,948,704	18,763,298	73,865	488.52
AE3	LB	6,639,158	1,397,106	5,468	10.07
	NG	519,419,981	13,758,788	54,235	318.30

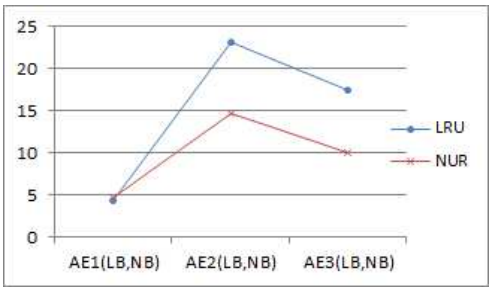

그림 6. LRU와 NUR 알고리즘의 에너지 소모 비교
Fig. 6 Energy consumption comparison between LRU and NUR algorithms

LRU나 NUR 알고리즘과 무관하게 그룹 캐시 관리 방법 (LG, NG)은 권고되지 않는다는 점이다. 아울러 표 7의 궁극적 시사점은 그림 6에서 보는 바와 같이 총 에너지 소모 관점에서 NUR 알고리즘이 LRU의 약 68%에 불과해 그 우수성이 뚜렷하다는 점인데, 이는 적중률 관점에서는 NUR 알고리즘이 더 낮았다는 점에서 매우 흥미로운 결과이다.

V. 결 론

SSD는 제자리 업데이트 즉, 덮어쓰기가 불가능하다는 원천적 한계로 인하여, 그동안 익숙해져 있었던 HDD 기반 페이지 스와핑, 파일 시스템, 그리고 캐시 정책 수립 등에 많은 변화를 요구하여 현재 분야별 연구가 활발하게 진행 중이다. 이 연구에서는 대표적인 캐싱 전략인 LRU와 NUR 알고리즘을 FTL 캐싱에 적용했을 때의 성능을 시뮬레이션 기법으로 평가하고 비교・분석하였다. 분석 결과 캐시 적중률은 LRU가 약 10% 포인트 우수한 반면 가비지 수집 비용을 포함한 플래시 연산 총 에너지 소모는 오히려 NUR이 LRU의 약 68% 수준에 머물러 32% 더 우수하다는 흥미로운 결과를 확인하였다. 이는, 적중률 및 가비지 수집 비용 모두를 만족하는 보다 정교한 캐싱 전략의 필요성에 대한 시사점이기도 하다.

References

[1] https://en.wikipedia.org/wiki/Flash_memory
[2] V. Mohan, T. Bunker, L. Grupp, S. Gurumurthi, M.R. Stan, S. Swanson, “Modeling Power Consumption of NAND Flash Memories Using FlashPower,” IEEE Transactions on Computer-Aided Design of Integrated Circuit and Stsrems, Vol. 32, No. 7, pp. 1031-1044, 2013.
[3] http://www.storagereview.com/corsair_vengeance_ddr3_ram_disk_review
[4] https://www.anandtech.com/show/6337/samsung-ssd-840-250gb-review
[5] http://www.storagereview.com/wd_black_4tb_desktop_hard_drive_review_wd4003fzex
[6] D. Lee, Y.I. Eom, “SSD Caching For Improving Performance of Virtualized IoT Gateway,” Journal of KIISE, Vol. 42, No. 8, pp. 954-960, 2015 (in Korean).
[7] C. Han, J Ryu, D Lee, K. Kang, H. Shin, “File-System-Level SSD Caching for Improving Application Launch Time,” Journal of KIISE, Vol. 42, No. 6, pp. 691-698, 2015 (in Korean).
[8] Samsung Electronics, "Samsung SSD 850 Pro Data Sheet Rev. 3," 2017, Available: http://downloadcenter.samsung.com/content/UM/201711/20171115103115156/Samsung_SSD_850_PRO_Data_Sheet_Rev_3.pdf
[9] H. Choi, Y. Kim, “An Efficient Cache Management Scheme of Flash Translation Layer for Large Size Flash Memory Drives,” Journal of the Korea Society of Computer and Information, Vol. 20, No. 11, pp. 31-38, 2015 (in Korean).

[10] S.Y. Kim, S.I. Jung, "A Log-based Flash Translation Layer for Large NAND Flash Memory," Proceedings of IEEE 8th International Conference on Advanced Communication Technology, Vol. 3, pp. 1641-1644, 2006.

[11] T.S. Chung, D.J. Park, S. Park, D.H. Lee, S.W. Lee, H.J. Song, "A Survey of Flash Translation Layer," Journal of Systems Archicture, Vol. 55, No. 5, pp. 332-343, 2009.

[12] S.H. Kim, J.W. Kwak, "Garbage Collection Technique using Erasure Interval for NAND Flash Memory-based Storage Systems," International Journal of Applied Engineering Research, Vol. 11, No. 7, pp.5188-5194, 2016.

[13] S.H. Hwang, J.W. Kwak, "Garbage Collection Technique for Reduction of Migration Overhead and Lifetime Prolongment of NAND Flash Memory," IEMEK J. Embed. Sys. Appl., Vol. 11, No. 2, pp. 125-134, 2016 (in Korean).

[14] J. Liu, S. Chen, T. Wu, H. Zhang, "A Novel Hot Data Identification Mechanism for NAND Flash Memory," IEEE Transactions on Consumer Electronics, Vol. 61, No. 4, pp. 463-469, 2015.

[15] D. Kang, S. Baek, J. Choi, "Probability Based Cache Management for SSD", Proceedings of KIISE Winter Conference, pp. 75-77, 2014 (in Korean).

[16] H.B. Lee, T.Y. Chung, "In-depth Analysis and Performance Improvement of a Flash Disk-based Matrix Transposition Algorithm," IEMEK J. Embed. Sys. Appl., Vol. 12, No. 6, pp. 377-384, 2017 (in Korean).

[17] Y. Kim, B. Taurus, A. Gupta, B. Urgaonkar, "FlashSim: A Simulator for NAND Flash-based Solid-State Drives," Proceedings of the First International Conference on Advances in System Simulation (SIMUL), pp. 125-131, 2009.

Hyung-Bong Lee (이 형 봉)

He received the B.S. and M.S. degrees in Computer Science from Seoul National University, Seoul, Korea, in 1984 and 1986 respectively. He received his Ph.D. degree in Computer Science from Kangwon National University, Chuncheon, Korea, in 2002. From 1986 to 1994, he was a senior engineer in Computer R&D Division of LG Electronics. From 1995 to 1998, he was with DEC (Digital Equipment Corporation) as a UNIX consultant. From 1999 to 2003, he was an Associate Professor at Honam University, Gwangju, Korea. Since 2004, he has been a Professor in the Department of Computer Science & Engineering at Gangneung-Wonju National University, Wonju, Korea. His current research interests include embedded systems, wireless sensor networks, and data mining algorithms.

Email: hblee@gwnu.ac.kr

Tae-Yun Chung (정 태 윤)

He received the B.S., M.S., and Ph.D. degrees in the School of Electrical & Computer Engineering at Yonsei University, Seoul, Korea in 1987, 1989, and 2000 respectively. From 1989 to 1996, he was a Research Engineer of Samsung Advanced Institute of Technology. From 1996 to 2001, he was with Samsung Electronics as a Senior Research Engineer. From 2000 to 2001, he was a vice chair of International DVD-Forum. Since 2001, he has been with the Department of Electronics Engineering at Gangneung-Wonju National University, Gangneung, Korea, and is currently a Professor. Since 2004, he has been with GEMS-CRC (Gangwon Embedded Software Cooperative Research Center, Gangneung, Korea) as the Chef. His major fields are image signal processing, digital video encoding, multimedia, copy protection , and his current research interests are embedded system, sensor network, video encoding.

Email: tychung@gwnu.ac.kr