KIPS Trans. Comp. and Comm. Sys.
Vol.8, No.7 pp.159~166 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2019.8.7.159

Performance Evaluation of HMB-Supported DRAM-Less NVMe SSDs

Kyu Sik Kim† · Tae Seok Kim††

ABSTRACT

Unlike modern Solid-State Drives with DRAM, DRAM-less SSDs do not have DRAM because they are cheap and consume less power. Obviously, they have performance degradation problem due to lack of DRAM in the controller and this problem can be alleviated by utilizing host memory buffer(HMB) feature of NVMe, which allows SSDs to utilize the DRAM of host. In this paper, we show that commercial DRAM-less SSDs surely exhibit lower I/O performance than other SSDs with DRAM, but they can be improved by utilizing the HMB feature. Through various experiments and analysis, we also show that DRAM-less SSDs mainly exploit the DRAM of host as mapping table cache rather than read cache or write buffer to improve I/O performance.

Keywords : DRAM-Less SSD, NVMe, Host Memory Buffer

HMB를 지원하는 DRAM-Less NVMe SSD의 성능 평가

김 규 식† · 김 태 석††

요 약

상용화된 많은 SSD와 달리 DRAM-less SSD는 원가절감, 전력소모량 감소 등의 이유로 DRAM을 가지고 있지 않다. 따라서 DRAM의 부재로 인해 입출력 성능이 저하될 가능성이 존재하며, 이는 호스트의 메모리 일부를 SSD 컨트롤러가 사용할 수 있는 NVMe 인터페이스의 HMB 기능을 통해 개선할 여지가 있다. 본 논문에서는 현재 상용화된 여러 DRAM-less SSD가 DRAM을 가지고 있는 동급 SSD에 비해 실제로 입출력 성능이 떨어지지만 HMB 기능을 사용해 일부 개선하고 있으며, 이는 SSD 컨트롤러가 호스트의 메모리를 매핑테이블 캐시로 주로 사용하고 있기 때문이라는 점을 다양한 실험을 통해 증명한다.

키워드 : DRAM-Less SSD, NVMe, Host Memory Buffer

1. 서 론

일반적으로 Solid State Drive(SSD)는 입출력 성능을 향상시키고 NAND 플래시의 수명을 개선하기 위한 목적으로 컨트롤러에 DRAM을 가지고 있다[1]. 하지만 DRAM은 SSD 동작에 필수적인 요소는 아니므로, 비용 절감, 전력소모 감소 등의 목적으로 이를 제거한 형태의 DRAM-less SSD가 등장하였다[2, 9, 10]. 이는 앞서 기술한 다양한 이점도 있지만, 본래 DRAM이 성능 향상을 목적으로 사용되었으므로 이를 제거한 DRAM-less SSD는 필연적으로 성능이 하락할 수밖에 없다.

NVMe 인터페이스를 채택한 DRAM-less SSD의 경우, host memory buffer(HMB)를 이용하여 성능 하락 문제를 개선할 수 있다[3, 10]. HMB는 SSD가 호스트의 메모리 일부를 사용하는 기능으로, 컨트롤러에 DRAM이 부재하여 성능 저하가 발생하는 DRAM-less SSD에 적합한 기능이다. 실제로, HMB를 지원하는 DRAM-less SSD 제품들이 최근 상용화되어 판매되고 있고, 다양한 운영체제에서도 이 기능을 점차 지원하고 있다[8].

본 연구에서는 현재의 DRAM-less SSD가 DRAM을 사용하는 기존의 SSD 대비 실제로 성능상 결점이 있는지, 그리고 HMB를 통해 DRAM-less SSD의 성능 개선이 가능한지를 확인하였다. 또한 다양한 실험을 통해 DRAM-less SSD들이 HMB를 어떠한 목적으로 사용하는지 실험적으로 분석하였다. 실험 결과, DRAM-less SSD가 기존의 SSD 대비 입출력 성능이 다소 떨어지며, HMB를 대부분 매핑테이블 캐시로 사용함으로써 그 성능을 일부 보완하고 있음을 확인하였다.

※ 이 논문은 2019년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2017R1A2B4008536).
† 준 회 원 : 광운대학교 컴퓨터공학과 석·박사통합과정
†† 정 회 원 : 광운대학교 컴퓨터정보공학부 교수
Manuscript Received : January 17, 2019
Accepted : March 27, 2019
* Corresponding Author : Tae Seok Kim(tskim@kw.ac.kr)

2. 관련 연구

2.1 DRAM-less SSD

SSD 컨트롤러의 DRAM은 주로 성능 향상과 NAND 플래시의 수명 문제를 개선하기 위해 매핑테이블 캐시, 쓰기 버퍼 등의 용도로 사용된다[1, 6]. 즉, DRAM은 SSD 컨트롤러 동작에 필수 요소가 아니며, 일반적으로 이러한 역할로 사용 가능한 최소한의 SRAM이 SSD에 포함되어 있다. 이러한 이유로 SSD 구조에서 DRAM을 제거한 형태의 DRAM-less SSD가 최근 빠른 속도로 보급되고 있다. 이는 비록 성능 문제를 안고 있으나, DRAM을 가진 기존의 SSD 대비 신뢰성 향상, 생산 비용 절감, 전력 소모 감소, 장치 크기 축소 가능 등의 장점을 가지고 있어[2, 9, 10], 향후 DRAM-less SSD가 SSD 시장의 상당 부분을 차지할 것으로 예측된다[9].

2.2 NVM Express (NVMe)

SSD를 타겟으로 만들어진 고성능 호스트 컨트롤러 인터페이스로서 NAND 플래시 기반의 SSD 뿐만 아니라 미래의 비휘발성 메모리 기반 제품 까지도 지원할 목표로 설계되었다. 고속의 입출력 처리 성능을 제공하기 위해 호스트와 SSD 간 입출력 요청 처리를 위한 다수의 submission, completion 큐를 제공하며, host memory buffer(HMB)를 비롯한 다양한 기능을 제공한다[3, 7, 11].

2.3 Host Memory Buffer (HMB)

호스트의 DRAM 일부를 SSD가 사용할 수 있도록 지원하는 NVMe의 기능 중 하나로, NVMe 1.2에서 추가되었다[3]. 해당 메모리 공간을 어떻게 사용하는지는 전적으로 SSD 제조사가 결정한다. SSD와 호스트 운영체제에서 모두 지원하여야 사용할 수 있으며, NVMe SSD가 HMB를 지원할 경우 호스트 운영체제에서 NVMe Admin 명령어 중 하나인 Set Feature 명령어를 SSD에 전달하여 해당 기능을 활성화 또는 비활성화 할 수 있다. 최신 리눅스 커널의 NVMe 디바이스 드라이버는 NVMe SSD가 HMB를 지원하는 경우 장치 초기화 과정에서 HMB 활성화를 시도하도록 구현되어 있다[8]. 이를 사용할 경우, 순차 연산보다는 임의 연산 시 SSD의 입출력 성능이 개선되는 것으로 알려져 있다[10].

2.4 SSD의 내부 정보 분석

SSD에 전달된 입출력 요청의 결과로 SSD 내부 정보를 분석하는 연구는 이전부터 진행되어 왔다. [4]에서는, SSD에 일정한 규칙의 입출력 요청을 보내고, 처리 시간을 다양한 방법으로 분석하여 컨트롤러 내 읽기 캐시의 크기, 쓰기 버퍼의 크기, 클러스터 페이지 및 클러스터 블록의 크기와 같은 SSD 내부 정보를 추출하였다. [5]에서는, 실험적으로 클러스터 페이지와 클러스터 블록 크기를 추출하고, 이를 기반으로 한 리눅스 입출력 스케줄러를 제안하였다.

3. SSD의 컨트롤러 내 DRAM 부재의 영향 분석

컨트롤러 내에 DRAM이 있는 기존의 SSD 대비 DRAM-less SSD가 가지는 입출력 성능 저하 문제를 분석하기 위해 Table 2의 호스트 PC에서 입출력 벤치마크 툴인 fio을 이용하여 Table 1의 6개의 SSD를 대상으로 성능 평가 실험을 진행하였다. SSD-A, SSD-B, SSD-C는 HMB를 지원하는 DRAM-less SSD이며, SSD-D, SSD-E, SSD-F는 DRAM이 있는 기존의 SSD이다. SSD-A, SSD-C, SSD-D는 컨트롤러 DRAM의 유무를 제외한 나머지 하드웨어 구성이 유사하고, SSD-B와 SSD-E도 마찬가지로 DRAM의 유무를 제외하고 유사한 하드웨어를 가지며, SSD-F는 실험한 SSD들 중 가장 좋은 하드웨어 구성을 가진다. 그리고 SSD-A, SSD-B, SSD-C는 HMB 기능을 제공하는데, 커널 디바이스 드라이버에서 지원하는 HMB의 크기를 제한적으로 조정할 수 있다. 본 실험에서는 HMB 크기를 SSD-A와 SSD-C는 480MB, SSD-B는 64MB로 할당하고 실험을 진행하였다. 이는 각 기기에서 지원 가능한 최대 HMB 크기이다.

입출력 요청 부하에 따른 성능 변화를 확인하기 위해 워크로드를 Table 3의 *Light*, *Heavy* 두 종류로 구분하고, 읽기, 쓰기 요청을 순차패턴과 임의패턴으로 각각 요청하여 입출력

Table 1. Tested SSDs

Product	SSD-A	SSD-B	SSD-C	SSD-D	SSD-E	SSD-F
	SP A80	HP EX900	TAMMUZ M730	Kingston A1000	WD Black 3D	Samsung 970 PRO
Interface	PCIe 3.0 x2	PCIe 3.0 x4	PCIe 3.0 x2	PCIe 3.0 x2	PCIe 3.0 x4	PCie 3.0 x4
	NVMe 1.2	NVMe 1.3	NVMe 1.2	NVMe 1.2	NVMe 1.3	NVMe 1.3
Controller DRAM	*DRAM-less*	*DRAM-less*	*DRAM-less*	O, size unknown	512MB	512MB
NAND Flash	3D TLC	3D TLC	3D TLC	3D TLC	3D TLC	3D MLC
Capacity	512GB	500GB	512GB	480GB	500GB	512GB
HMB	*O (8MB~480MB)*	*O (64MB)*	*O (8MB~480MB)*	X	X	X

Table 2. Host PC Environment

Category	Description
Processor	Intel i7-8700 3.2GHz
Main memory	DDR4 16GB
OS	Ubuntu 16.04.4 (Kernel 4.13.10)
Benchmark tool	fio-2.2.10

Table 3. Workloads of Fio

Name	Parameter	Description
Light	active CPU cores	1
	# running threads	1
	Total I/O size	16GB
	Block size	4KB
Heavy	# active CPU cores	12
	# running threads	72
	Total I/O size	100% of each SSD
	Block size	4KB

대역폭을 측정하였다. *Light*는 fio에서 하나의 쓰레드로 적은 양의 입출력 요청함으로써 상대적으로 SSD에 가해지는 부하가 적은 워크로드이며, *Heavy*는 fio에서 다중 쓰레드로 많은 양의 입출력을 요청하여 *Light* 워크로드 대비 입출력 부하가 많은 워크로드이다.

실험 결과는 Fig. 1과 같다. *Light* 워크로드에서는 컨트롤러 DRAM의 부재가 적지 않은 입출력 성능 차이를 보였다. 특히, SSD-A와 SSD-C는 모든 경우에서 SSD-D 대비 더 낮은 성능을 보였다(Fig. 1A-D의 “w/o HMB”). 반면, SSD-B는 SSD-E와 비교하였을 때 큰 성능 차이를 보이지 않았으며, 오히려 순차 읽기에서는 더 좋은 성능을 보였다(Fig. 1A)의 “w/o HMB”).

Heavy 워크로드에서도 DRAM-less SSD의 성능이 상대적으로 떨어지는 것을 확인할 수 있었다. 주로 읽기 연산에서 그 경향이 심화되며, 특히 임의 읽기 연산에서는 더 두드러졌다(Fig. 1E, Fig. 1G의 “w/o HMB”).

실험에 사용된 SSD들의 내부 하드웨어 설계가 명확히 알려져 있지 않고, 내부 펌웨어가 SSD 성능 개선을 위해 수행하는 다양한 연산도 역시 공개되어 있지 않다. 따라서 본 장의 실험을 통해 관찰되는 DRAM-less SSD가 HMB 도움 없이도 기존 SSD 대비 더 나은 성능을 보이는 것과 같은 일부 실험 결과에 대해 명확한 결론을 내리긴 어렵다. 다만, 대체로 DRAM-less SSD는 기존의 SSD 대비 확실히 더 낮은 입출력 성능을 보이며, 특히 쓰기 연산보다 읽기 연산에서 그 차이가 커 읽기 성능이 중요한 응용에서 DRAM-less SSD의 사용은 적합하지 않는 것으로 파악된다.

4. HMB의 역할 분석

본 절에서는 현재 상용화된 DRAM-less SSD에서 HMB를 어떻게 사용하는지, 이를 통해 SSD의 입출력 성능이 얼마나 개선되는지 분석한 결과를 제시한다. 현재, SSD가 HMB 영역을 어떻게 사용하는지는 온전히 SSD 제조사의 몫이며[3], 대부분의 경우 이 공간을 어떻게 활용하는지 공식적으로 언급하고 있지 않으므로 이를 다양한 실험을 통해 간접적으로 확인하였다. “4.1”은 3절과 동일한 fio로, “4.2”, “4.3”, “4.4”는 마이크로 벤치마크 툴을 직접 작성하여 성능을 평가하였다.

4.1 Is HMB used to Improve I/O Performance?

먼저 HMB가 입출력 성능 개선 목적으로 사용되는지 확인하기 위해, 3절과 동일한 실험환경에서 HMB 활성화를 통해 DRAM-less SSD의 입출력 처리 성능이 얼마나 개선되는지 분석하였다(Fig. 1)의 “w/ HMB”).

Light 워크로드에서 SSD-A와 SSD-C의 경우, 순차·임의 읽기 연산 성능이 개선되었다(Fig. 1A, 1C의 “w/ HMB”). 주목할 만한 점은 HMB를 사용하여 임의 읽기와 순차 읽기의 성능이 거의 같아졌다는 점이다. SSD-B도 마찬가지로 임의 읽기 성능이 상당히 개선되었지만(Fig. 1C의 “w/ HMB”), 순차 읽기 성능에는 이르지 못했다.

Heavy 워크로드에서는 모든 DRAM-less SSD에서 임의 읽기의 성능 향상을 확인할 수 있었으며(Fig. 1H의 “w/ HMB”), 이 또한 순차 쓰기 성능 수준까지 개선되었다.

본 실험을 통해 HMB를 지원하는 DRAM-less SSD에서

HMB 영역을 사용하면 입출력 성능이 실제로 향상되며, 특히 임의 읽기 성능이 상당히 개선됨을 확인하였다. 다만, 본 실험만으로는 HMB를 어떤 용도로 사용하여 입출력 성능이 개선되었는지 알 수 없었다. 따라서 컨트롤러에 DRAM을 가지고 있는 기존의 SSD에서 DRAM 공간을 읽기 캐시(read cache), 쓰기 버퍼(write buffer), 매핑테이블 캐시(mapping table cache)[1] 용도로 활용하고 있음을 근거해 DRAM-less SSD에서는 HMB로 어떻게 사용하고 있는지 분석하였다.

4.2 Is HMB used as Read Cache?

[4]에서의 연구를 참고해 HMB를 읽기 캐시로 사용하는지 다음과 같이 확인하였다.

1) 요청하는 입출력이 운영체제의 각종 캐시 및 버퍼를 거치지 않도록 SSD 장치를 연다(Fig. 2A의 “1)”).
2) 이후 과정에서 사용하지 않는 위치의 데이터를 HMB 크기(C_{HMB})만큼 읽는다. (Fig. 2A의 “2)”)
3) 특정 위치로부터 데이터를 n KB만큼 읽을 때 걸린 시간(Fig. 3의 “NAND”)을 측정한다(Fig. 2A의 “3)”). 이는 읽기 캐시 존재 여부와 무관하게 NAND 플래시로부터 데이터를 읽어오는 시간이다. 읽기 캐시가 존재한다면, “2)”에서 사용되지 않는 데이터가 채워졌을 것이기 때문이다.
4) “3)”을 반복하고, 이 때 걸린 시간(Fig. 3의 “Cache”)을 측정한다(Fig. 2A의 “4)”). 읽기 캐시가 존재한다면, 본 단계에서 전체 혹은 일부 데이터를 해당 캐시로부터 읽어올 것이다. 그렇지 않다면, 이 과정에서도 모든 데이터를 NAND 플래시로부터 읽어올 것이다.
5) n 값을 1씩 증가시켜 가며, 요청 크기가 2MB가 될 때까지 “2)”, “3)”, “4)” 단계를 반복한다.

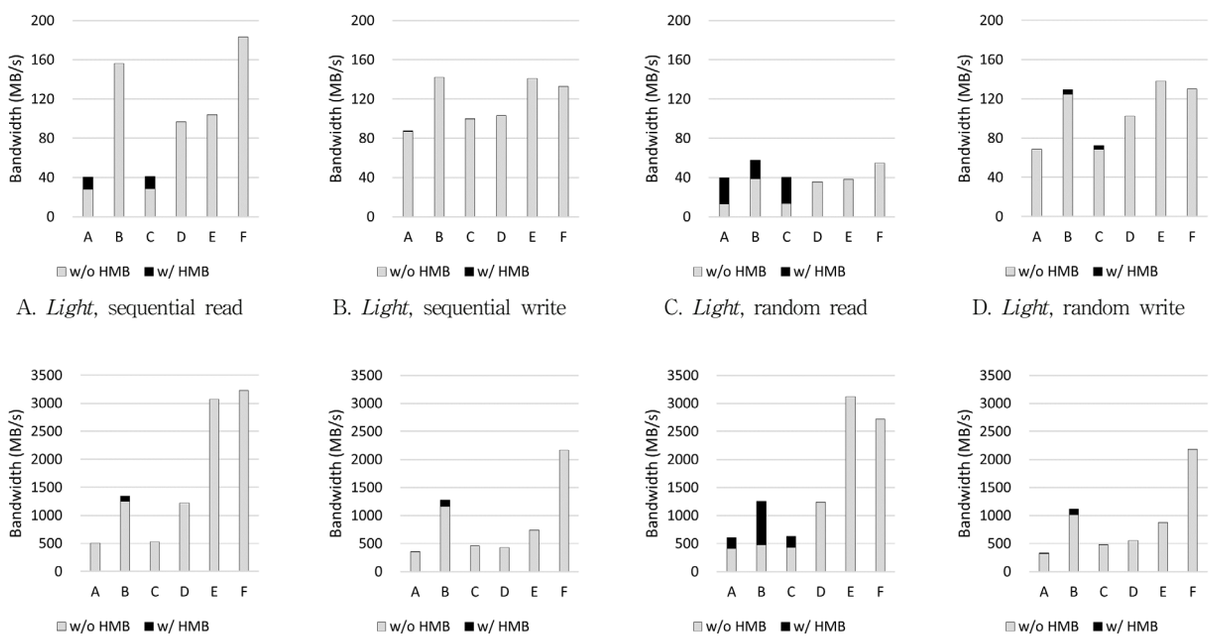


Fig. 1. Results of fio by Workloads and I/O Types

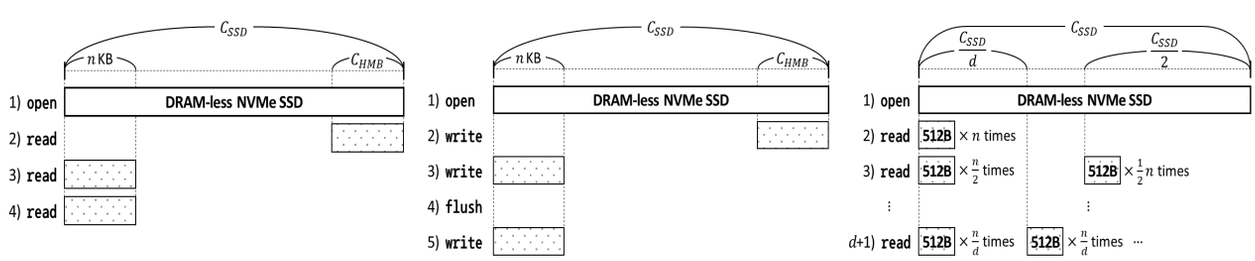
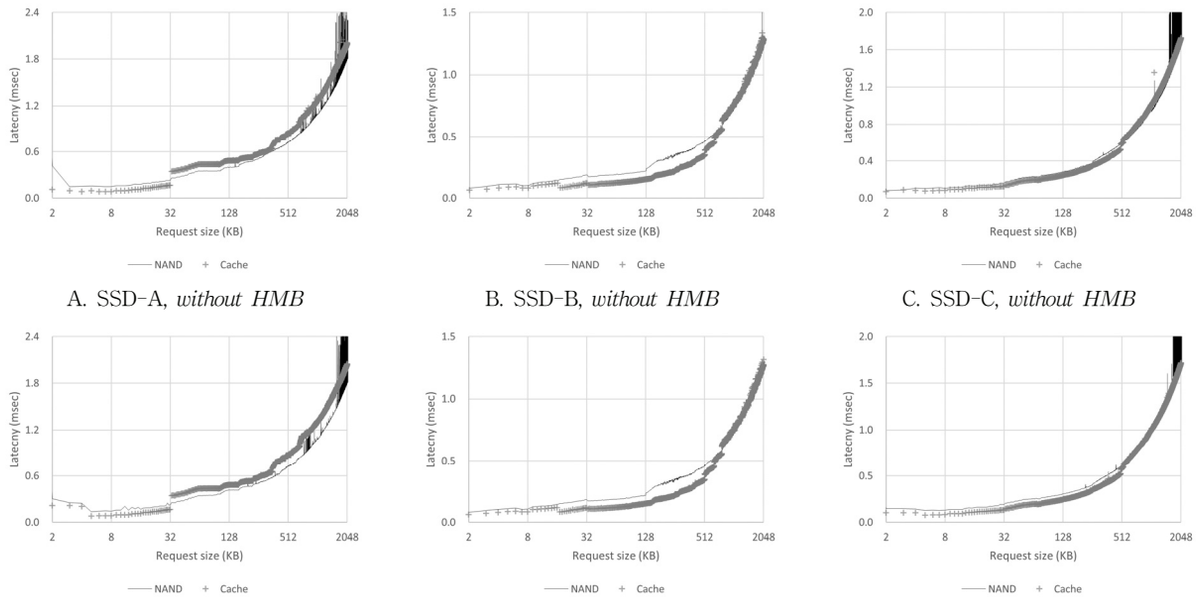


Fig. 2. Roles of HMB



A. SSD-A, *without HMB* B. SSD-B, *without HMB* C. SSD-C, *without HMB*

D. SSD-A, *with HMB* E. SSD-B, *with HMB* F. SSD-C, *with HMB*

Fig. 3. Existence of Read Cache in HMB

HMB를 활성화할 경우 SSD-A는 읽기 요청 크기가 32KB일 때(Fig. 3D), SSD-B는 768KB(Fig. 3E), SSD-C는 512KB (Fig. 3F) 이후로 4)에서처럼 동일한 데이터를 반복해서 읽는 “Cache”가 3)과 같이 처음 데이터를 읽는 “NAND” 대비 같거나 더 크다. 즉, 해당 지점 이전까지는 읽기 캐시에서 데이터를 읽어오므로 “Cache”가 “NAND” 대비 더 나은 성능을 보였지만, 이후로는 읽기 캐시에 없는 일부 데이터를 NAND 플래시로부터 읽어오게 되므로 “Cache”의 성능이 점차 “NAND”와 유사해진다. 여기서 주목할 만한 점은 HMB를 사용하지 않을 때에도 동일한 현상을 보이며(Fig. 3A-3C), HMB 사용 여부와 무관하게 전반적으로 성능이 동일하다. 결론적으로, 실험에 사용된 DRAM-less SSD들은 읽기 캐시가 존재하나, 그 역할을 HMB가 아닌 SLC 캐시와 같은 컨트롤러 내 다른 매체가 수행하고 있는 것으로 추정된다.

4.3 Is HMB used as Write Buffer?

HMB를 쓰기 버퍼로 사용하는지 확인하기 위해 기존 연구[4]를 참고해 다음과 같이 실험을 진행하였다.

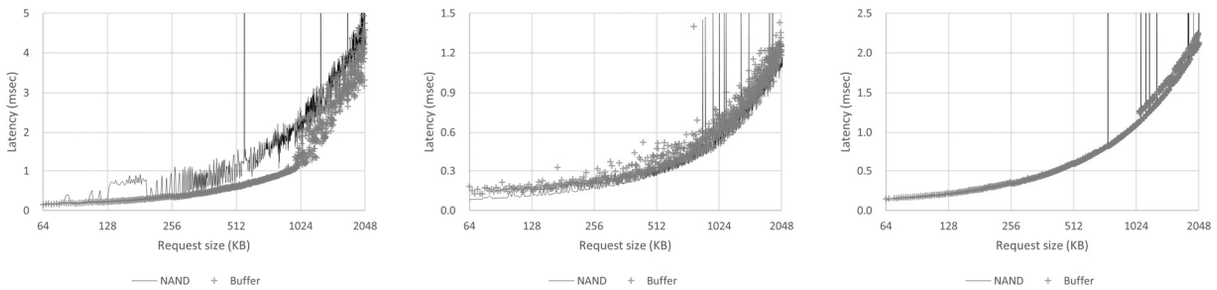
1) 요청하는 입출력이 운영체제의 각종 캐시 및 버퍼를 거치지 않도록 SSD 장치를 연다(Fig. 2B의 “1)”).
2) 쓰기 버퍼를 가득 채우기 위해 이후 과정에서 사용되지 않는 위치에 데이터를 C_{HMB} 만큼 쓴다(Fig. 2B의 “2)”).
3) “2)”와 겹치지 않는 영역에 n KB의 데이터를 기록할 때 걸린 시간(Fig. 4의 “NAND”)을 측정한다(Fig. 2B의 “3)”). 쓰기 버퍼가 존재하는 경우 “2)”에서 버퍼를 가득 채웠으므로 쓰기 버퍼 내 데이터를 NAND 플래시에 반영하며, 버퍼 크기 이상의 데이터도 마찬가지로 NAND 플래시에 반영한다. 즉, 본 단계에서 걸린 시간은 NAND 플래시에 데이터를 기록하는 시간이다.
4) NVMe의 Flush 명령어를 통해 쓰기 버퍼 내용을 NAND 플래시에 반영하고(Fig. 2B의 “4)”), “3)”의 단계와 동일한 크기의 쓰기 요청에 걸린 시간(Fig. 4의 “Buffer”)을 측정한다(Fig. 2B의 “5)”). 쓰기 버퍼가 존재한다면, 요청한 데이터의 전체 혹은 일부분이 비어있는 버퍼에 기록되어 “3)” 대비 더 나은 성능을 보일 것이며, 그렇지 않다면 두 결과가 큰 차이가 없을 것이다.
5) n 값을 1씩 증가시켜 가며, 요청 크기가 2MB가 될 때까지 “2)”, “3)”, “4)” 단계를 반복한다.

Fig. 4D에서 볼 수 있듯이, SSD-A는 쓰기 요청 크기가 964KB 보다 클 때, 4)의 과정에서 측정되는 시간인 “Buffer”의 값이 급격히 산개되며 커지고 있다. 즉, 해당 지점 이전까지는 쓰기 요청이 쓰기 버퍼에만 기록되었기에 “Buffer”가 “NAND” 대비 더 좋은 그리고 일정한 성능을 보였으나, 그 이후로 쓰기 버퍼 크기 이상의 데이터를 NAND 플래시에도 반영하면서 성능이 저하된다. 또한, 이 결과가 HMB의 사용 여부와 무관하게 동일하므로, SSD-A는 HMB가 아닌 다른 매체를 쓰기 버퍼로 사용하고 있음을 알 수 있다.

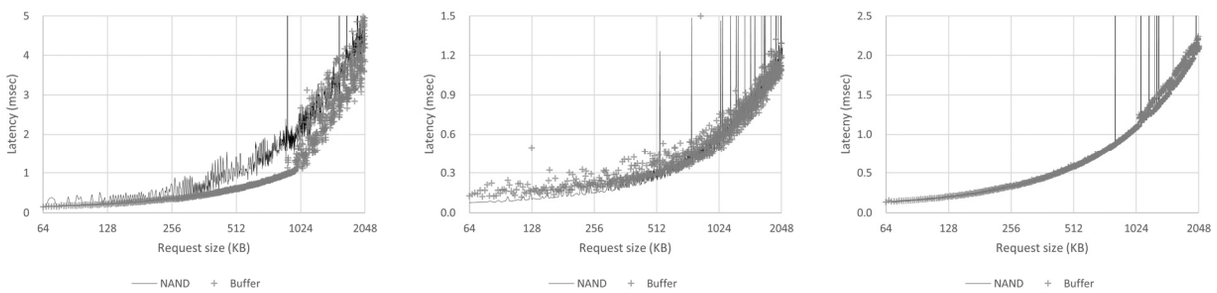
SSD-B와 SSD-C에서는 SSD-A와 같은 성능 변화 경향을 볼 수 없었으므로 SSD에 쓰기 버퍼가 존재하는지는 확신할 수 없으나, HMB 활성화에 따라 경향이 달라지지 않았으므로 HMB를 쓰기 버퍼로 사용하지 않는다는 사실은 확인할 수 있다.

4.4 Is HMB used as Mapping Table Cache?

최근의 SSD는 주로 페이지 수준 주소 매핑을 사용하며, 페이지 매핑테이블의 크기는 일반적으로 전체 SSD 용량의 0.1% 정도를 차지한다[6]. SSD의 용량이 커짐에 따라 매핑테



A. SSD-A, *without HMB*　　B. SSD-B, *without HMB*　　C. SSD-C, *without HMB*



D. SSD-A, *with HMB*　　E. SSD-B, *with HMB*　　F. SSD-C, *with HMB*

Fig. 4. Existence of Write Buffer in HMB

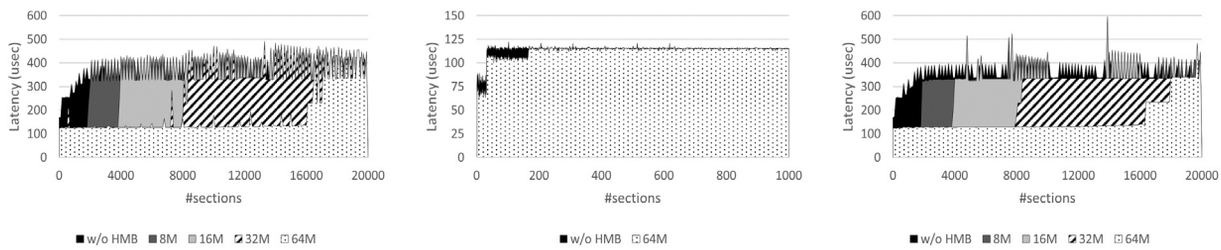
이블의 크기가 커지고, 따라서 이를 컨트롤러 DRAM에 전부 적재할 수 없게 되어 NAND 플래시에 있는 원본 매핑 데이터 중 사용할 일부를 DRAM에 캐싱하여 사용한다. 호스트가 요청한 입출력을 처리할 때마다 매핑테이블 내용을 통해 논리 주소를 물리 주소로 변환해야 하므로, 이 매핑 데이터를 캐싱하는 것은 SSD 성능에 큰 영향을 미친다.

우리는 HMB를 이러한 매핑테이블 캐시로 사용하는지 알아보기 위해, 다음과 같은 실험을 진행하였다.

1) 요청하는 입출력이 운영체제의 각종 캐시 및 버퍼를 거치지 않도록 SSD 장치를 연다(Fig. 2C의 “1)”).
2) 전체 SSD 용량에 대한 매핑테이블을 전부 구성하기 위해 SSD의 모든 영역의 데이터를 지운 후, 낮은 논리 주소부터 순차적으로 다시 기록한다.
3) SSD 전체 용량(C_{SSD})을 d개의 구간으로 분할하고(Fig. 5)의 “#sections”), 각 분할 구간의 시작 위치, 즉 0, $\frac{C}{d}$, $2\times\frac{C}{d}$, ..., $(d-1)\times\frac{C}{d}$ 순으로 읽고(Fig. 2C의 “2)”, “3)”, ..., “$d+1$)”), 이를 $\frac{n}{d}$회 반복한다. 읽을 때마다 SSD에 요청 가능한 최소 읽기 단위인 512바이트만큼 요청하는데, 이는 읽기 연산 처리가 미치는 영향을 최소화하기 위함이다. “2)”에서 순차적으로 데이터를 기록했으므로, 각 논리 주소, 즉 분할 구간 시작 부분에 대한 매핑 정보가 서로 다른 캐시 엔트리에 포함될 것으로 기대된다. 즉, 본 단계에서 NAND 플래시에 있는 매핑 테이블 엔트리 다수가 캐싱될 것이다.
4) 분할 구간의 개수인 d를 증가시키며 “3)”을 반복 수행하고, 읽기 요청 마다 걸린 시간을 측정하여 산술 평균을 계산한다. d 값이 커질수록 캐시 엔트리 수가 증가할 것이고, 그 수가 캐시 엔트리 수 한도에 다다르면 캐시 미스로 인한 처리 시간 증가를 가져올 것으로 기대한다.

세 종류의 DRAM-less SSD에 대해 HMB 크기를 조절하며 실험한 결과는 Fig. 5와 같다. SSD-A와 SSD-C는 HMB가 8MB, 16MB, 32MB, 64MB 할당되었을 때 구간 수가 약 2,000개, 4,000개, 8,000개, 16,000개인 지점에서 평균 처리 시간이 2배가량 증가하는 경향을 보였다(Fig. 5A, Fig. 5C). 또한, HMB를 사용하지 않을 때는 구간 수가 200개인 지점에서 평균 처리 시간이 크게 증가하였다. 결론적으로, 두 SSD는 HMB가 활성화되어 있을 때는 이를 매핑테이블을 캐시하는 용도로 사용하며, 비활성화되어 있을 때는 컨트롤러 내부 매체를 통하여 매핑 데이터를 캐싱하고 있다. 또한 HMB 크기와 평균 처리 시간이 크게 변화하는 구간 수의 관계가 비례하는 것으로 보아 HMB 영역의 특정 비율만큼 항상 매핑테이블 캐시용으로 사용된다는 것도 알 수 있다. 추가로, HMB 전체를 매핑테이블 캐시로 사용한다고 가정하면, 캐시 테이블 엔트리는 4KB로 추정할 수 있다.

SSD-B는 나머지 두 SSD에서 보이는 명확한 경향은 확인할 수 없었으나(Fig. 5B), 구간 수가 166 이하일 때 HMB를 사용하여 평균 처리 시간이 개선되었으므로, 해당 SSD도 HMB를 매핑테이블 캐시로 사용하고 있음을 알 수 있다.



A. SSD-A B. SSD-B C. SSD-C

Fig. 5. Existence of Mapping Table Cache in HMB

5. 결론 및 향후 연구 과제

본 연구에서는 컨트롤러에 DRAM을 가지고 있지 않은 DRAM-less SSD가 DRAM을 가지고 있는 기존의 SSD 대비 읽기 연산의 성능이 저하되는 것을 실험으로 확인하였다. 또한 상용화된 DRAM-less NVMe SSD 제품들이 이러한 성능 저하 문제를 NVMe가 제공하는 HMB 기능으로 완화하고 있고, 특히 HMB를 매핑테이블 캐시로 활용하여 개선하고 있음을 확인하였다.

본 연구 결과를 바탕으로 다양한 응용에서 DRAM-less SSD와 HMB를 사용하는 것이 응용 성능에 미치는 영향을 분석하고자 한다. 또한 HMB를 적극적으로 이용하여 SSD 기반 시스템의 I/O 성능을 개선하기 위한 연구를 진행하고자 한다. HMB를 사용하면 SSD가 가지고 있지 않던 DRAM을 갖게 되거나, 혹은 가지고 있던 DRAM 크기가 확장되는 효과를 얻을 수 있다. 따라서 SSD가 DRAM을 활용하는 방법이 HMB를 활용하는 SSD에서도 여전히 유효한지 검토가 필요하다. 이와 더불어, HMB 공간을 SSD와 호스트가 공유하여 I/O 성능을 개선하는 다양한 방법을 연구하고자 한다.

References

[1] I. H. Shin and J. D. Kim, “Performance analysis of buffer management policy considering internal parallelism of solid state drives,” *IEICE Electronics Express*, 2015.

[2] M. Wu, “DRAM-less SSD – The New Trend for Embedded System,” *Flash Memory Summit*, 2015.

[3] NVM Express Base Specification Revision 1.3c [Internet], http://nvmexpress.org/wp-content/uploads/NVM-Express-1_3c-2018.05.24-Ratified.pdf

[4] J. H. Kim, D. W. Jung, J. S. Kim, and J. H. Huh, “A methodology for extracting performance parameters in solid state disks (SSDs),” in *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, London, 2009.

[5] B. K. Ko, Y. J. Kim, and T. S. Kim, “Performance Improvement of I/O Subsystems Exploiting the Characteristics of Solid State Drives,” in *International Conference on Computational Science and Its Applications*, Santander, 2011.

[6] H. J. Kim, D. K. Shin, Y. H. Jeong, and K. H. Kim, “SHRD: Improving Spatial Locality in Flash Storage Accesses by Sequentializing in Host and Randomizing in Device,” in *15th USENIX Conference on File and Storage Technologies*, Santa Clara, 2017.

[7] NVMe Overview [Internet], https://nvmexpress.org/wp-content/uploads/NVMe_Overview.pdf

[8] NVMe device driver in Kernel 4.13.10 [Internet], https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/nvme/host/pci.c?h=v4.13.10#n2179

[9] S. Huang, “DRAM-Less SSD Facilitates HDD Replacement,” *Flash Memory Summit*, 2015.

[10] S. Yang, “Improving the Design of DRAM-Less PCIe SSD,” *Flash Memory Summit*, 2017.

[11] J. Zhang, M. Y. Kwon, D. H. Gouk, S. J. Koh, C. L. Lee, M. Alian, M. J. Chun, M. T. Kandemir, N. S. Kim, J. H. Kim, and M. S. Jung, “FlashShare: Punching Through Server Storage Stack from Kernel to Firmware for Ultra-Low Latency SSDs,” *in 13th USENIX Symposium on Operating Systems Design and Implementation*, 2018.

김 규 식

https://orcid.org/0000-0002-5470-9205

e-mail : kks@kw.ac.kr

2014년 광운대학교 컴퓨터공학과(공학사)

2010년～현 재 광운대학교 컴퓨터공학과 석・박사통합과정

관심분야 : Operating Systems, Storage Systems, Embedded Systems

김 태 석

https://orcid.org/0000-0002-4200-2384

e-mail : tskim@kw.ac.kr

2000년 서울대학교 전산과학과(공학사)

2002년 서울대학교 컴퓨터공학부(공학석사)

2007년 서울대학교 컴퓨터공학부(공학박사)

2008년 삼성전자 책임연구원

2008년~현 재 광운대학교 컴퓨터정보공학부 교수

관심분야 : Operating Systems, Storage Systems, Multimedia Systems, Embedded Systems