ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 50, No. 11, pp. 924-930, 2023. 11
https://doi.org/10.5626/JOK.2023.50.11.924

모바일 저장장치 성능 향상을 위한 통합 호스트-저장장치 주소변환 테이블 캐시 관리기법

(Integrated Host-SSD Mapping Table Cache Management Techniques for Improving Performance of a Mobile Storage Device)

김 윤 아 † 최 인 혁 † 이 성 진 †† 김 지 홍 †††
(Yoona Kim) (Inhyuk Choi) (Sungjin Lee) (Jihong Kim)

요 약 최근 저장장치의 사이즈가 커지며 플래시 기반 저장장치의 주소변환 테이블 관리에 필요한 메모리 공간의 요구량 또한 점차 커지는 추세이다. 모바일 저장장치로 사용되는 UFS의 경우 하드웨어 및 가격적인 제약으로 인해 UFS 내장 SRAM 메모리 공간을 늘리기에 많은 어려움이 존재하여 늘어난 주소변환 테이블 관리에 성능적 문제가 발생하게 된다. 이를 보완하기 위해, 호스트 DRAM 메모리를 사용해 주소변환 테이블의 일부를 적재할 수 있는 HPB 기법이 제안되었다. 본 논문에서는 호스트 메모리와 UFS의 SRAM이 통합적으로 관리되지 않아 주어진 자원을 낭비하는 문제를 발견하였고 두 캐시 계층의 특성을 고려한 통합 주소변환 테이블 관리기법을 제안한다. 본 기법을 통하여 낭비되는 캐시 자원을 최소화하고 저장장치 지연시간을 감소시키며 불필요한 저장장치 수명 저하를 방지할 수 있다. 모바일 응용 트레이스 기반으로 실험한 결과, 기존 관리기법 대비 캐시 적중률은 5% 향상하였고, 낭비되었던 캐시 공간 자원을 95% 감소하였으며 주소변환 테이블 업데이트로 발생되는 가비지 컬렉션 횟수가 43% 감소하였다.

키워드: 모바일 저장장치 시스템, 논리-물리 주소변환, 호스트 퍼포먼스 부스터, 논리-물리 캐시 관리

Abstract As the size of a storage device gradually increases, the demand for on-device memory capacity required for managing the address mapping translation of a NAND flash-based storage device increases. The on-device memory capacity of a mobile storage device, Universal Flash Storage (UFS), does not increase due to H/W and cost constraints, making it challenging to manage the increased address translation table. To resolve the problem, Host Performance Booster (HPB), which borrows host-side DRAM memory to load portions of the address translation table was introduced. In this paper, we demonstrate that the HPB-enabled system does not work in an integrated manner with the device-side SRAM, therefore wasting the given memory resource. We propose integrated mapping

· 이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다.
· 이 논문은 삼성전자의 지원(IO201207-07809-01)을 받아 수행된 결과임
· 이 논문은 2022 한국소프트웨어종합학술대회에서의 '모바일 저장장치 성능 향상을 위한 두 캐시 계층 통합 주소 변환 테이블 관리기법'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 서울대학교 컴퓨터공학부 학생
yoonakim@davinci.snu.ac.kr
ihchoi@davinci.snu.ac.kr
†† 종신회원 : 대구경북과학기술원 전기전자컴퓨터공학과 교수
sunjin.lee@dgist.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수(Seoul Nat'l Univ.)
jihong@davinci.snu.ac.kr
(Corresponding author임)

논문접수 : 2023년 3월 13일
(Received 13 March 2023)
논문수정 : 2023년 7월 17일
(Revised 17 July 2023)
심사완료 : 2023년 7월 24일
(Accepted 24 July 2023)

정보과학회논문지 제50권 제11호(2023. 11)

table management techniques that consider the distinctive features of each cache layer. By adopting these techniques, we aim to minimize wasted cache resources, reduce storage latency, and prevent unnecessary degradation of the storage lifetime. Based on the evaluation results, the cache hit ratio is improved by 5% while the wasted memory resource is reduced by 95%, and the number of device-side garbage collections is reduced by 43% compared to the baseline scheme.

Keywords: mobile storage system, L2P address translation, HPB, L2P cache management

1. 서 론

모바일 기기를 사용하는 사용자의 데이터 및 응용들의 사이즈가 점차 커지며 저장장치 용량의 요구량이 점차 커지는 추세이다. 이러한 사용자 요구량을 만족하기 위해 모바일 기기에 탑재되는 플래시 기반 저장장치인 UFS의 용량이 TB 단위까지 커지게 되었다[1]. UFS의 사이즈가 커짐에 따라 플래시 기반 저장장치 관리에 필수적으로 필요한 논리-물리 주소변환 테이블의 사이즈 또한 저장장치 용량에 비례하며 커지게 된다.

플래시 기반 저장장치는 물리적 특성상 호스트가 관리하는 논리주소와 실제 플래시 메모리에서 저장된 위치를 나타내는 물리주소는 항상 일치할 수 없다. 이는 덮어쓰기가 허용되지 않는 플래시 기반 저장장치의 특성 때문이다. 호스트가 이미 쓰인 데이터에 대해 수정을 요청한 경우, 기존 데이터가 작성되어 있는 물리 페이지를 무효화하고 쓰기 가능한 상태에 있는 새로운 물리 페이지를 할당 받아 쓰기 요청을 처리해야 한다.

이러한 특성에 따라 모든 플래시 기반 저장장치는 논리 주소에 대응되는 실제 플래시 메모리 상 물리적 페이지의 위치를 나타내는 논리-물리(L2P) 주소변환 테이블을 유지한다. 논리-물리 주소변환 테이블의 전체 사이즈는 저장장치 용량의 0.1% 정도이며 전체 테이블은 항상 플래시 메모리에 저장되어 있다. 예를 들어, UFS 용량이 1 TB일 경우 전체 주소변환 테이블의 사이즈는 1 GB의 용량을 차지하게 된다.

주소변환 과정은 긴 지연시간을 유발할 수 있다. 저장장치에 전달된 모든 읽기/쓰기 요청들을 처리하기 위해 플래시 변환 계층은(Flash Translation Layer, FTL) 대응되는 주소변환 테이블 엔트리를 참조해야 한다. 빠른 응답시간 안에 호스트 요청을 처리하기 위해 저장장치에 내장된 SRAM 메모리에 테이블을 적재하여 주소변환 과정을 비교적 빠르게 처리할 수 있다.

논리-물리 주소변환 테이블의 사이즈가 커질수록 더 많은 저장장치 내부 SRAM 메모리 자원이 요구된다. 하지만, 모바일 기기는 여러 하드웨어 및 가격적인 제약으로 인해 요구량에 맞추어 UFS 내장 메모리 용량을 확장하기에는 많은 어려움이 있다. 저장장치 용량이 TB 단위로 커지는 추세이지만 여전히 UFS에 내장된 SRAM의 용량은 대략 1-2 MB 정도이다. 플래시 변환 계층은 DFTL[2] 기법을 통해 호스트 요청이 많은 논리-물리 주소변환 테이블의 일부 엔트리만 SRAM에 적재하여 모든 요청을 처리한다.

전체 주소변환 테이블의 사이즈가 커짐에 따라 주소변환으로 인해 저장장치 응답시간이 더욱 길어지는 현상이 발생한다[3,4]. 이는 전체 테이블 사이즈 대비 SRAM이 적재할 수 있는 엔트리의 수가 점차 줄어들어 캐시 실패(Cache miss) 경우가 많아지기 때문이다. 예를 들어, 호스트가 전달한 읽기 요청에 대응되는 엔트리가 빠른 접근이 가능한 SRAM에 적재되어 있지 않을 경우(캐시 실패) 비교적 느린 플래시 메모리에서 대응하는 논리-물리 주소 엔트리를 읽어와야 한다. 이 경우 캐시 적중(Cache hit) 대비 읽기 요청 처리 시간이 두 배가 되어 저장장치 응답시간이 매우 느려진다.

제한적인 메모리 자원으로 인해 주소변환 과정이 저장장치 응답시간에 영향을 미치는 경우를 방지하고자 호스트 메모리 일부 공간을 저장장치 주소변환 테이블 캐시로 사용하도록 하는 Host Performance Booster (HPB), Host Memory Buffer (HMB) 그리고 Unified Memory Extension (UME)와 같은 기법들이 소개된 바 있다[5-9]. 위 기법들은 한정적인 저장장치 내부 메모리 공간으로 인해 늘어나는 논리-물리 주소변환에 대한 캐시 실패율(Cache miss ratio)을 효과적으로 줄여 저장장치 응답시간을 줄이는 효과를 가져올 수 있다. 최근 안드로이드 기반 모바일 기기에서는 주소변환 과정으로 병목이 되는 저장장치 응답시간을 해결하고자 HPB 기법을 적용하는 추세이다[10].

본 논문에서는 현재 HPB 기법은[10] 호스트 메모리 그리고 저장장치 메모리 캐시 계층이 통합적으로 관리되지 않아 주어진 메모리 자원이 낭비하는 문제점을 발견하였다. 모바일 시스템의 호스트 DRAM 메모리는 저장장치의 SRAM보다 공간적으로 비교적 여유롭지만, 여전히 사용자 응용이 충분하게 사용하기에는 제한적인 자원으로 알려져 있다[4,11,12]. 따라서 단순 저장장치 성능 향상을 위해 HPB를 적용하는 시스템 환경에서는 주어진 메모리 자원을 낭비되지 않도록 하는 것이 높은 사용자 경험을 지원하는 것이 중요한 요소이다.

본 논문에서는 HPB가 적용된 시스템 상에서 두 독립적 주소변환 테이블 캐시 계층인 호스트 메모리와 저장장치 SRAM에 적재된 엔트리들을 각 특성에 따라 통합

적으로 관리하는 기법을 제안한다. 읽기/쓰기 요청에 따라 각 계층별로 엔트리를 관리하는 기법이 상이하며 이러한 특성을 캐시 관리에 적용하여 주어진 메모리 자원 낭비를 최소화함을 목표로 한다. 실제 모바일 기기에서 추출한 모바일 응용 트레이스를 기반으로 실험한 결과 기존 관리기법 대비 캐시 적중률이 5% 향상하고 낭비되었던 메모리 공간 자원의 95%가 감소한다. 또한, 주소변환 테이블 업데이트로 인한 저장장치 가비지 컬렉션을 43% 감소시켜 저장장치 수명 저하를 방지한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 필요한 배경지식을 설명한다. 3장에서는 호스트 메모리와 저장장치 SRAM 캐시 계층이 통합적으로 관리되지 못해 나타나는 문제를 분석한다. 이를 해결하기 위해 4장에서는 두 캐시 계층의 특성에 따라 동작하는 통합 관리기법을 설명한다. 5장에서는 제안 기법을 적용하였을 때의 성능 및 효과를 평가하고 6장에서 결론을 맺는다.

2. 배경지식

본 장에서는 본 연구에서 기반으로 하는 시스템에 대한 배경지식에 대해 간단한 예시를 통하여 설명한다.

2.1 기존 UFS 읽기 동작 방식

그림 1은 논리-물리 주소변환 테이블을 저장장치 SRAM상에만 적재하는 DFTL의 데이터 읽기 동작 과정을 나타낸다. 호스트 시스템은 읽기 요청을 수행하기 위해 논리주소만을 사용하여 UFS에 요청을 전달한다 ❶. 먼저, UFS 저장장치 시스템의 플래시 변환 계층은 전달받은 논리주소에 해당하는 물리주소가 SRAM에 적재되어 있는지 확인한다 ❷. 만약 캐시 실패일 경우 전체 테이블이 저장된 플래시 메모리에서 읽기 요청을 통해 대응되는 물리주소를 읽어 SRAM에 적재한다 ❸. 플래시 메모리에서 읽어온 물리주소 혹은 캐시 적중을 통해 SRAM에 적재되어 있던 물리주소를 사용하여 플래시 메모리에 접근하여 호스트가 요청한 데이터 읽기를 수행하고, ❹, 호스트에게 데이터를 전달한다 ❺.

SRAM 메모리의 주소변환 테이블 엔트리에 대한 캐시 실패 혹은 적중 여부에 따라 호스트 읽기 요청에 대한 저장장치 응답시간은 상이하게 된다. 그림 2는 SRAM 메모리에서 캐시 적중 혹은 실패 상황에 따른 저장장치 읽기 요청 처리 시간을 간략하게 나타낸다. SRAM 캐시 적중의 경우 호스트가 요청한 데이터에 대한 플래시 메모리 읽기 요청만 수행하면 된다. SRAM 캐시 실패의 경우 해당하는 주소변환 테이블 엔트리를 플래시 메모리 에서 읽은 다음 호스트가 요청한 데이터 읽기를 수행할 수 있으므로 기존 읽기 응답시간의 두 배가 걸리게 된다.

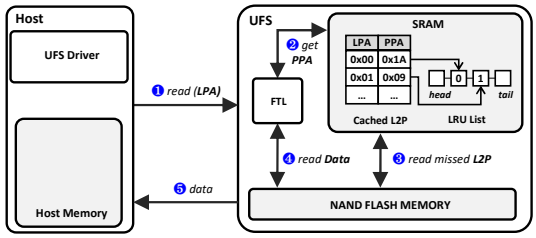


그림 1 DFTL 시스템 읽기 요청 처리 방식
Fig. 1 Read Operation Path with DFTL

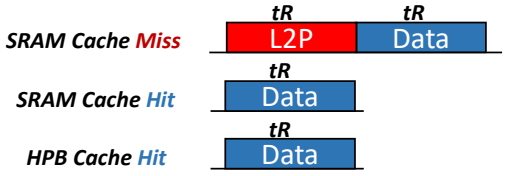


그림 2 저장장치 읽기 요청 처리 시간
Fig. 2 Storage Latency of a Read Operation

2.2 HPB적용 시스템 읽기 동작 방식

그림 3은 HPB를 시스템에 적용한 경우의 호스트 읽기 수행 과정을 나타낸다. HPB를 적용한 시스템은 호스트 DRAM 메모리 일부를 HPB 메모리로 할당받아 저장장치 주소변환 테이블의 일부를 적재한다. 읽기 요청을 저장장치에 전달하기에 앞서 HPB 메모리에 해당하는 주소변환 엔트리가 적재되어 있는지 확인한다. 해당 엔트리가 적재된 경우를 HPB 캐시 적중이라 한다 ①. 따라서 저장장치에 읽기 요청을 전달할 때 논리주소 뿐만 아니라 물리주소와 함께 해당 읽기 요청을 전달할 수 있다 ②. 논리주소와 함께 물리주소를 전달받은 UFS는 기존 기법과 다르게 바로 플래시 메모리 읽기 요청을 수행하고 ③, 호스트에게 읽은 데이터를 전달하게 된다 ④.

HPB 캐시 적중인 경우에는 해당하는 논리-물리 주소 엔트리를 비교적 느린 플래시 메모리에서 읽을 필요가 없다. 따라서 그림 2에서 표현된 바와 같이 기존 DFTL

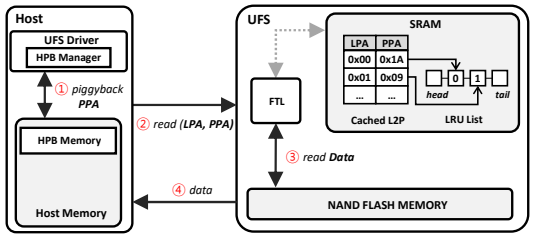


그림 3 HPB 시스템 읽기 요청 처리 방식
Fig. 3 Read Operation Path with HPB

의 SRAM 캐시 적중의 경우와 거의 동일한 읽기 요청 응답시간 안에 처리할 수 있다.

3. 기존 HPB 관리기법의 공간 활용 분석

기존 HPB 관리기법을 적용한 시스템이 얼마나 잘 동작하는지 이해하기 위해, UFS의 SRAM만 활용하는 단일 캐시 시스템(Single-Cache System)과 HPB가 결합된 통합 캐시 시스템(Two Level-Cache Integrated System) 환경을 시뮬레이터에 구현하고 실제 모바일 기기에서 추출한 트레이스를 기반으로 캐시 적중률을 평가하였다. 사용한 UFS의 용량은 128 GB이고, SRAM 사이즈는 기본 값으로 512 KB로 설정하였다. 기존 관리기법이 통합적으로 잘 동작하는지 평가하기 위해, 두 시스템의 전체 캐시 사이즈를 동일하게 설정하였다. 물리적으로는 불가능하지만 단일 캐시 시스템의 SRAM 사이즈는 비교하는 통합 캐시 시스템이 사용하는 캐시 사이즈와 동일하게 설정하였다. 그림 4는 실험 결과로, x축은 전체 주소변환 테이블의 해당 비율만큼 시스템 캐시 사이즈 설정 값을 나타내고, y축은 각 케이스의 캐시 적중률이다.

그림 4를 통해, 통합 캐시 시스템의 캐시 적중률은 단일 캐시 시스템보다 최대 12% 낮은 것을 확인할 수 있다. 즉, 통합 캐시 시스템은 늘어나는 캐시 공간을 효율적으로 활용하고 있지 못함을 파악할 수 있다. 이러한 현상은 두 캐시 계층의 관리기법 분석을 통해 각 계층을 통합적으로 관리하는 기법의 부재로 인해 늘어난 캐시 메모리 공간이 비효율적으로 관리되고 있기 때문임을 파악하였다. 즉, 디바이스 드라이버에 위치하는 HPB 관리기법과[10] UFS 내부의 DFTL기법은 서로 다른 계층을 전혀 고려하지 않으며 동작하고 있기 때문이다.

그림 5는 HPB 메모리 공간에 발생하는 공간적 비효율이 발생하는 예시를 나타낸다. HPB는 많은 읽기 요청이 있는 논리주소에 대해 선별적으로 호스트 메모리에 적재한다. 만약 적재된 논리주소에 대해 쓰기 요청이 오면 해당 엔트리는 무효화되어 더티 플래그가 1로 세팅된다. HPB 메모리에서의 더티 플래그는 유효한 물리주소를 나타내는지를 표시하는데 사용된다. 덮어쓰기가 불가능한 저장장치의 특성상 기존 물리주소는 무효화되고 새로운 물리주소로 주소변환 테이블의 엔트리가 변경되어, HPB에 적재된 기존 엔트리는 더 이상 유효하지 않기 때문이다. 필요한 경우 UFS로부터 변경된 엔트리를 다시 HPB 메모리로 불러와 더티 플래그를 다시 0으로 변경한다. HPB 메모리는 이전 엔트리 값을 가지고 있거나 업데이트된 데이터를 UFS로부터 전달받아 SRAM과 공존 엔트리를 유지하기 때문에 공간적인 비효율이 발생하게 된다.

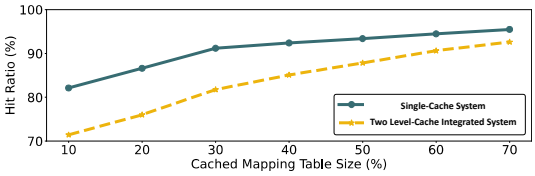


그림 4 통합 캐시 시스템의 공간 효율 분석 결과
Fig. 4 Space Utilization Analysis Results of the HPB-Integrated System

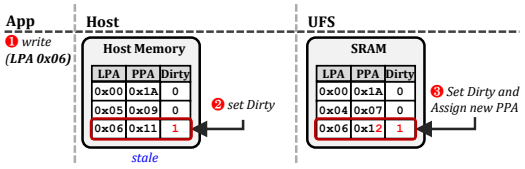


그림 5 기존 시스템의 HPB 공간 비효율
Fig. 5 Inefficient Space Usage of the HPB Memory

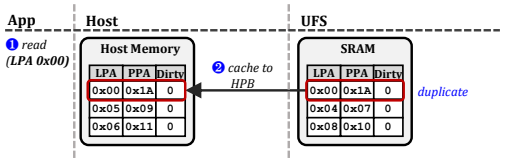


그림 6 기존 시스템의 SRAM 공간 비효율
Fig. 6 Inefficient Space Usage of the Device-side SRAM

반면, UFS의 SRAM은 요청된 모든 읽기 그리고 쓰기 요청들에 해당하는 논리-물리 주소 엔트리를 적재해야한다. 쓰기 요청에 해당하는 엔트리는 새로운 물리주소로 업데이트되어 SRAM에 더티 플래그 비트 1과 함께 적재되어 최대한 나중에 쫓겨나게 된다. SRAM에서의 더티 플래그는 플래시 메모리에 저장되어있는 엔트리 값과 SRAM에 적재된 값이 다른 경우를 나타낸다. 더티 엔트리를 최대한 늦게 쫓아내는 이유는, 업데이트된 엔트리 값을 플래시에 저장된 주소변환 테이블에 플래시 쓰기 요청을 통해 수정하기 때문이다. SRAM에 더티 엔트리를 오래 유지하게 된다면 최대한 주소변환 테이블 수정으로 인한 플래시 쓰기를 줄일 수 있기 때문에 플래시 수명 그리고 성능 측면에서 효율적이다. 따라서 SRAM에서는 업데이트된 엔트리를 최대한 오래 유지하기 때문에 HPB 메모리에 동일한 엔트리를 공존하도록 하는 것은 부족한 캐시 공간을 비효율적으로 활용하게 된다.

그림 6은 UFS의 SRAM 메모리 공간에 발생하는 공간적 비효율에 대해 설명한다. UFS에서는 전달된 읽기 요청을 처리하기 위해 해당하는 엔트리를 SRAM 에 적

재한다. 만약, HPB 정책에 의해 HPB 메모리로 해당 엔트리가 적재될 경우 HPB 메모리와 UFS의 SRAM에 동일한 엔트리가 공존하게 되는 문제가 발생한다. 따라서 서로 다른 특성을 가지는 두 캐시 계층은 각 특성에 맞게 통합적으로 관리되어 주어진 공간을 최대한 활용하는 동시에 높은 캐시 적중률을 제공할 수 있도록 활용되어야 한다.

4. 두 캐시 계층 통합적 주소변환 테이블 관리 기법

본 장에서는 두 캐시 계층의 특성을 고려한 통합 관리기법을 제시한다. 본 관리기법을 통하여 두 독립적인 캐시 계층에 적재된 엔트리 공존 현상을 최소화하여 주어진 공간 자원을 최대한으로 활용하도록 한다.

4.1 HPB의 호스트 메모리 캐시 관리기법

HPB를 활용하였을 때 가장 효과적인 경우는 호스트 메모리에 적재된 엔트리들에 대해 읽기 요청이 오는 경우이다. 반면에, 적재된 엔트리에 쓰기 요청이 수행되어 해당 값이 무효화된 경우 새로 업데이트된 물리주소의 엔트리를 다시 읽어오기 전까지는 시스템 성능에 도움이 되지 않으며 공간적 낭비이다. 따라서 HPB 메모리에는 쓰기 요청에 의해 업데이트되어 더티 플래그를 유지하며 유효하지 않은 엔트리들을 적재하는데 낭비되는 현상을 방지해야 한다.

무효화된 엔트리의 업데이트된 값을 UFS에서 전달받게 되면, 추후 읽기 요청에 대해 HPB 캐시 적중이 될 수 있지만 이는 동기화 오버헤드를 유발할 수 있다. 그 이유는 UFS에서 최신 항목을 가져오더라도 많은 항목들은 다시 쓰기 요청이 수행되어 또 무효화될 가능성이 높기 때문이다. 또한, 업데이트된 엔트리는 이미 UFS SRAM에 존재하기 때문에 공간적으로 비효율적이다.

그림 7은 제안하는 기법의 동작 방식을 보여준다. 이러한 현상을 방지하기 위해 쓰기 요청에 의해 새로운 물리주소로 업데이트되는 엔트리는 호스트 메모리에서 즉시 제거하는 기법을 제안한다. 모바일 시스템의 호스트 메모리는 응용들이 사용하기에도 여전히 충분하지

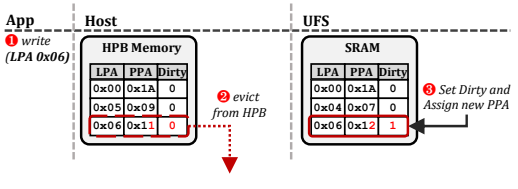


그림 7 HPB 메모리 공간 중복 엔트리 방지 기법
Fig. 7 Co-existing Entry Eviction Technique in the HPB Memory

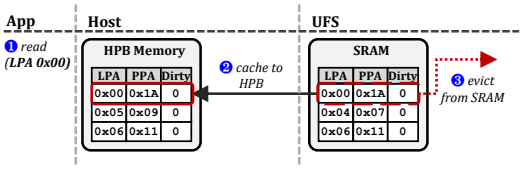


그림 8 SRAM 메모리 공간 중복 엔트리 방지 기법
Fig. 8 Co-existing Entry Eviction Technique in the SRAM Memory

않은 자원으로 제안하는 기법을 통해 단순 저장장치의 성 능을 위해 낭비되는 메모리 자원을 최소화한다. 또한 최신 항목을 가져오기 위한 동기화 오버헤드를 방지한다.

4.2 UFS 내장 SRAM 캐시 관리기법

HPB 관리기법에 의해 호스트 메모리로 적재되는 엔트리들은 무조건 최근에 UFS에 전달된 읽기 요청들에 대한 엔트리이다. 따라서 해당 엔트리들은 항상 UFS SRAM에 적재되어 있다. 이러한 엔트리들은 HPB 관리기법에 의해 호스트 메모리에서 쫓겨나지 않는 이상, HPB메모리에서 항상 적중되며 더 이상 SRAM에서 참조되지 않는다. 또한 SRAM의 사이즈는 매우 한정적이기 때문에 두 계층이 중복되는 엔트리를 보유하는 것은 메모리 공간적으로 낭비다. 이 문제를 해결하기 위해, HPB로 엔트리들을 적재할 때 SRAM 캐시에서 해당 엔트리들을 즉시 쫓아내는 기법을 제안한다. 그림 8과 같이, HPB로 적재되는 엔트리들은 SRAM에서 바로 쫓아내어 공존 엔트리로 인한 공간 낭비를 최소화한다.

제안하는 두 관리기법을 결합하면 자연스럽게 많은 더티 엔트리들은 SRAM에 남게 되고 클린(플래시에 저장된 물리 주솟값과 캐시에 적재된 엔트리의 값이 같은 경우) 엔트리들은 HPB에 대부분 남게 된다. 또한, 각 계층에 적재되는 하나의 엔트리는 1024개의 연속적인 논리 주솟값들로 이루어져 있어, 본 기법을 통해 더티 엔트리들이 쫓겨날 때 동일한 페이지로 한 번에 제거될 가능성이 높아져 테이블 업데이트로 인한 플래시 쓰기 동작 횟수가 크게 감소하게 된다.

5. 실 험

본 장에서는 본 논문에서 제안한 두 캐시 계층의 특성을 고려한 HPB 캐시 관리기법과 SRAM 캐시 관리기법을 통합하여 구현한 시스템에 대한 성능 평가를 진행한다. 평가 지표로는 캐시 적중률, 두 캐시 계층 사이에 공존하는 엔트리 비율을 사용한다. 추가로, 제안하는 기법을 통하여 플래시 메모리에 주소변환 테이블 업데이트 빈도가 줄어들며 감소하는 가비지 컬렉션 빈도에 대해서도 평가한다.

5.1 실험 환경

제안한 기법들의 효과를 증명하기 위해 UFS 내부 SRAM 관리기법을 수정하여 평가해야 한다. 하지만, 실제 UFS의 내부 펌웨어를 수정하는 것은 불가능하기 때문에 FlashDriver[13] 시뮬레이터를 사용해 HPB 계층까지 동작하도록 확장하고 제안하는 시스템을 평가하였다. 시뮬레이터 상에서 사용한 모바일 워크로드로는 실제 Pixel 5a 모바일 기기 상에서 유명한 응용 9개를 실제 사용자 패턴을 고려해 실행하며 blktrace[14]를 통해 수집한 I/O 트레이스를 사용하였다. HPB의 사이즈는 전체 주소변환 테이블의 50%를 적재할 수 있도록 설정하였다.

5.2 실험 결과

그림 9는 제안하는 두 캐시 계층의 통합 관리기법을 적용했을 때와 기존 관리기법을 적용했을 때의 캐시 적중률을 보여준다. 제안하는 통합 관리기법을 적용하였을 때, 기존 통합 캐시 시스템 대비 적중률을 향상시켜 물리적으로는 불가능 하지만 제안하는 시스템이 목표로 하는 단일 캐시 시스템의 적중률에 가까워진다. 그 이유에 대해서는 제안하는 기법을 통해 주어진 캐시 공간 사이에 공존하는 엔트리들을 최소화하여 공간적으로 더 효율적이기 때문이다.

그림 10을 통해 제안하는 기법의 주어진 캐시 공간 효율 정도를 확인할 수 있다. 해당 그래프는 전체 호스트 읽기 및 쓰기 요청 중 HPB 메모리와 UFS의 SRAM에서 공통적으로 캐시 적중이 발생했던 경우의 비율을 나타낸다. 기존 관리기법의 경우 전체 읽기/쓰기 요청 중 20.1%의 요청에 해당하는 엔트리가 두 캐시 계층에서 공존하여 공간적 낭비가 발생하였다. 반면에, 제안하는

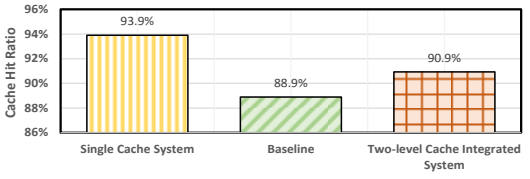


그림 9 다른 캐시 시스템의 캐시 적중률

Fig. 9 Cache Hit Ratio of Different L2P Cache Settings

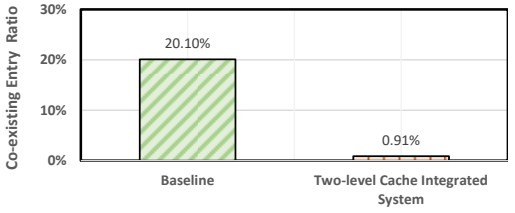


그림 10 공존 엔트리 비율 차이

Fig. 10 Difference in the Co-existing Entry Ratios

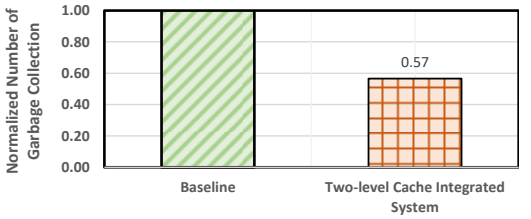


그림 11 디바이스 가비지 컬렉션 차이

Fig. 11 Difference in the Number of Device-side Garbage Collections

통합 관리기법은 0.9%의 요청만 공존하게 된다. 제안하는 관리기법은 주어진 캐시 공간을 최대한으로 활용하여 중요한 캐시 공간이 낭비되지 않도록 하며 보다 더 높은 캐시 적중률을 보장한다.

제안하는 기법을 통해 플래시 메모리에 저장되어 있는 전체 논리-물리 주소변환 테이블에 대한 업데이트 빈도가 줄어들어 전체적인 저장장치 가비지 컬렉션 발생 빈도가 줄어들게 된다. 그림 11은 기존 관리기법과 제안하는 기법의 가비지 컬렉션 횟수에 대한 변화를 나타내며 제안하는 통합 관리기법은 가비지 컬렉션 횟수를 기존 대비 43% 감소시킨다. 그 이유는, SRAM에는 대부분 더티 엔트리들만 위치하고, 클린 엔트리들을 적재하기 위해 더티 엔트리들이 쫓아내는 빈도가 줄어들게 되어 업데이트로 인한 플래시 페이지 쓰기 비용이 줄어들기 때문이다. 따라서 주소변환 테이블을 플래시 메모리에 저장하기 위한 쓰기 및 지우기 트래픽이 줄어들어 전체적인 저장장치 수명이 늘어나게 된다.

6. 결 론

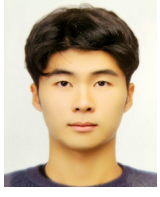
본 논문에서는 기 제안된 HPB 기법을 모바일 시스템에 적용했을 때, UFS의 내장 SRAM 캐시와 통합적으로 동작하지 못해 캐시 자원이 공간적으로 낭비되며 주어진 메모리 자원을 최대한 활용하지 못하는 문제점을 처음으로 지적하였다. 이러한 현상을 개선하기 위해, 논리-물리 주소변환 테이블을 적재하는 호스트 DRAM 메모리와 UFS 내장 SRAM의 두 캐시 계층의 특성을 분석하였다. 분석한 각 캐시 계층의 특성에 기반을 두어 두 캐시 계층의 통합 논리-물리 주소변환 테이블 캐시 관리기법을 제안하였다. 제안한 기법을 활용하여 불필요한 호스트 메모리 자원 낭비를 최소화하며 주어진 캐시 공간 자원을 최대한으로 활용할 수 있음을 실제 모바일 트레이스를 사용한 실험을 통하여 검증하였다.

References

[1] Samsung, https://semiconductor.samsung.com/estorage/

ufs/.
[2] Aayush Gupta et al, “DFTL: A Flash Translation Layer Employing Demand-Based Selective Caching of Page-Level Address Mappings”, *ASPLOS*, 2009.
[3] Chao Wu et al, “Boosting User Experience via Foreground-Aware Cache Management in UFS Mobile Devices”, *TCAD*, 2020.
[4] Yoona Kim et al, “Integrated Host-SSD Mapping Table Management for Improving User Experience of Smartphones”, *FAST*, 2023.
[5] JEDEC, “Universal Flash Storage (UFS) host performance booster (HPB) extension, version2.0”, 2020.
[6] Jeroen Dorjelo et al, Host Memory Buffer (HMB) Based SSD System, *Flash Memory Summit, 2015*.
[7] Konosuke Watanabe et al, “19.3 66.3 KIOPS-Random-Read 690MB/s-Sequential-Read Universal Flash Storage Device Controller with Unified Memory Extension”, *ISSCC*, 2014.
[8] Toshiba Corporation, “UFS Unified Memory Extension”, *JEDEC Mobile Forum*, 2014.
[9] Woo-Ram Jeong et al, “Improving Flash Storage Performance by Caching Address Mapping Table in Host Memory”, *HotStorage*, 2017.
[10] HPB Android kernel code. https://android.googlesource.com/kernel/common/+/refs/heads/android12-5.10/drivers/scsi/ufs/ufshpb.c.
[11] Sam Son et al, “ASAP: Fast Mobile Application Switch via Adaptive Prepaging”, *ATC*, 2021.
[12] Yu Liang et al, “Acclaim: Adaptive Memory Reclaim to Improve User Experience in Android Systems”, *ATC*, 2020.
[13] FlashDriver, https://github.com/dgist-datalab/FlashDriver.
[14] blktrace, http://linux.die.net/man/8/blktrace.

김 윤 아
2018년 한동대학교 컴퓨터공학부 학사. 2018년~현재 서울대학교 컴퓨터공학과 석박통합과정. 관심분야는 낸드 플래시 저장장치, 임베디드 시스템, 시스템 소프트웨어, 모바일 시스템



최 인 혁
2021년 홍익대학교 컴퓨터공학과 학사. 2021년~현재 서울대학교 컴퓨터공학과 석박통합과정. 관심분야는 낸드 플래시 저장장치, 임베디드 시스템, 시스템 소프트웨어, 모바일 시스템

이 성 진
2005년 고려대학교 전자공학 학사. 2007년 서울대학교 컴퓨터공학 석사. 2013년 서울대학교 컴퓨터공학 박사. 2013년~2016년 MIT CSAIL 박사후연구원. 2016년~2017년 인하대학교 컴퓨터공학과 조교수. 2017년~현재 DGIST 전기전자컴퓨터공학과 부교수. 관심분야는 저장장치 시스템, 운영체제, 시스템 소프트웨어

김 지 홍
1986년 서울대학교 계산통계학과 학사. 1988년 University of Washington 컴퓨터과학과 석사. 1995년 University of Washington 컴퓨터과학 및 공학과 박사. 1995년~1997년 미국 Texas Instruments 선임연구원. 1997년~현재 서울대학교 컴퓨터공학과 교수. 관심분야는 낸드 플래시 저장장치, 저전력 시스템, 임베디드 소프트웨어, 컴퓨터 구조