스마트폰의 사용자 경험 향상을 위한 SLC 모드 활용 모바일 저장장치 스왑 기법*

최인혁 ○, 김윤아, 천명준, 김지홍
서울대학교 컴퓨터공학부
{ihchoi, yoonakim, mjchun, jihong}@davinci.snu.ac.kr

Improving User Experience of Smartphones by Utilizing SLC Mode Swap Technique in Mobile Storage Devices

Inhyuk Choi○, Yoona Kim, Myoungjun Chun, Jihong Kim
Department of Computer Science and Engineering, Seoul National University

요 약

최근 모바일 응용의 데이터 양이 증가함에 따라 높은 사용자 경험을 제공하기 위해 충분한 크기의 메모리 공간을 제공해야 하지만 여러 제약으로 인해 사용자 경험이 저하되는 문제가 있다. 이를 해결하기 위해 낸드 플래시 저장장치로 스왑 공간을 확장하는 방법이 제안되고 있다. 하지만 저장장치의 단점인 느린 성능에 의해 사용자 경험이 저하하는 것을 방지하기 위해 ZRAM과 저장장치를 적절하게 혼용할 수 있도록 시스템 계층에서 연구들이 진행되어왔다. 본 논문에서는 제안된 시스템 계층의 기법들이 모바일 저장장치의 근본적인 문제점을 고려하지 않아 사용자 경험이 여전히 악화되는 것을 확인했다. 따라서, 모바일 저장장치의 본질적인 문제를 해결하고 사용자경험을 향상시키기 위해 SLC 모드를 활용하여 포그라운드 응용의 시작 지연시간을 개선하는 기법을 제안한다. 실제 개발 보드에서 실험한 결과, 기존 스왑 기법 대비 응용의 시작 지연시간이 평균 18.8%, 최대 32.5% 개선하였다.

1. 서 론

최근 모바일 기기의 활용도가 다양해지고 응용의 기능과 필요로 하는 데이터의 양이 증가하며 응용의 크기 및 사용자 데이터양이 점차 증가하는 추세이다. 모바일 환경에서 높은 사용자 경험을 제공하기 위해서는 응용이 사용하는 데이터가 메모리 공간에 존재해야 하므로, 증가하는 메모리 요구량에 따라 충분한 크기의 메모리 공간을 제공해야 한다. 하지만 모바일 기기의 하드웨어 및 가격 등의 제약으로 인해 호스트 메모리의 크기는 요구량에 맞추어 확장되지 못하는 추세이다. 이로 인해 메모리 부족 현상이 빈번하게 발생해 캐시 실패가 자주 발생하거나 최악의 경우 응용이 강제 종료되어 사용자 경험이 저하되는 문제가 발생한다.

모바일 환경에서는 사용자 경험에 미치는 악영향을 고려하여 메모리 부족 문제에 대한 전통적인 대응 방법인 저장장치를 활용한 스왑 기법 대신, 호스트 메모리 일부를 압축 기반의 스왑 공간으로 활용하는 ZRAM 기법이 주로 사용되어 왔다.[1] 급격하게 증가하는 메모리 수요를 ZRAM만으로 대응하기에 어려워지는 추세에 따라, 최근 저장장치 기반 스왑도 함께 활용하는 시스템이 제안되었다.[2] 이러한 시스템은 호스트 메모리보다 비교적 느린 저장장치의 응답시간으로 인해 여러 계층에서의 최적화 방안 없이는 오히려 사용자 경험이 치명적으로 저하될 수 있다고 알려져 있기 때문이다. 이를 해결하기 위해 최근 ZRAM 스왑과 저장장치 스왑을 적절하게 혼용하는 방법에 대한 연구들이 활발하게 진행되고 있다.[3,4,5,6] 해당 연구들 에서는 사용자 경험을 개선을 위해 저장장치 스왑이 사용자 응답시간에 최대한 반영되지 않도록 시스템 계층에서 응용 및 메모리 상황에 따라 스왑 횟수와 시점을 적절하게 조절하였다.

본 논문에서는 기 제안된 시스템 계층에서의 기법들은 저장장치 스왑으로 인한 사용자 경험 저하를 어느정도 완화할 수 있지만 기존 연구들에서 고려되지 않은 모바일 저장장치의 특성으로 인해 여전히 사용자 경험 저하에 치명적인 요인임을 확인하였다. 우리의 실제 모바일 실험 플랫폼을 적용한 실험에 따르면, 메모리 부족 상황에서 포그라운드 응용의 시작까지 걸리는 사용자 응답시간 중 저장장치 스왑으로 인한 응답시간 지연이 평균 28.9% 차지하고 있음을 확인하였고 저장장치 스왑을 적용하였을 때 미적용 시스템 대비 포그라운드 응용의 시작 지연시간이 평균 7.6% 증가하는 것을 확인하였다. 모바일 저장장치는 일반적인 SSD와 달리 하드웨어 제약으로 인해 내부 자원이 제한되어 저장장치 내부 캐시 자원을 활용하여 빠른 읽기 및 쓰기 성능을 제공하기에 어려움이 있다. 이러한 느린 저장장치의 읽기 쓰기 응답시간은 저장장치 스왑을 적용한 시스템에서 두가지 문제를 야기시킨다. 첫째, 호스트 메모리 및 ZRAM 스왑 공간이 부족한 상황에서 빠르게 메모리 공간을 확보해야 하는 경우 저장장치 스왑이 필수적으로 동반되며 사용자 응답시간에 느린 저장장치의 쓰기 성능이 그대로 포함되어 사용자 경험이 저하된다. 둘째, 저장장치로 스왑되어 있는 데이터를 사용자 요청에 의해 호스트 메모리로 스왑인 해야 할 때 느린 저장장치 읽기 성능만큼 사용자 응답시간이 지연된다.

본 논문에서는 모바일 환경에서 저장장치 기반 스왑에 의한 사용자 경험 저하의 본질적인 원인이 되는 저장장치의 느린 응답시간 문제를 해결할 수 있도록 기존 저장장치 스왑 기법을 다음과 같이 개선한다. 첫째, 느린 저장장치 쓰기 성능으로 인한 사용자 응답시간 저하를 최소화하기 위해 포그라운드 응용의 시작 지연시간 중 발생하는 저장장치 스왑아웃 요청에 대해 SLC 모드를 사용하여 빠르게 처리하여 응용 시작 중 메모리 부족 현상을 재빨리 해결한다. 둘째, 응용 재시작 중 저장장치로 스왑아웃 되어 있는 페이지들로 인한 응답시간 지연 문제를 응용 시작 중 사용하는 페이지를 식별해 두었다 저장장치로 스왑되는 경우 SLC 모드를 사용하게 하여 개선하도록 한다. 실제 모바일 보드를 사용해 성능 평가한 결과, 기존 기법 대비 응용 별 시작 지연시간이 평균 18.8%, 최대 32.5% 개선되는 것을 확인하였다.

* 이 논문은 삼성전자의 지원(IO201207-07809-01)을 받아 수행된 결과임. (교신저자: 김지홍)

본 논문의 구성은 다음과 같다. 2장에서는 기존 모바일 저장장치 기반 스왑의 문제점을 분석한다. 이러한 문제점을 해결하기 위해 3장에서는 포그라운드 응용의 시작 지연시간을 개선하는 SLC 모드 스왑 기법을 설명한다. 4장에서는 실제 안드로이드 모바일 보드 환경에 제안한 기법의 프로토타입을 구현하여 성능 및 효과를 평가하고 5장에서 결론을 맺는다.

2. 기존 모바일 저장장치 기반 스왑의 문제점과 개선 방안

본 장에서는 기존 ZRAM+저장장치 스왑 시스템의 문제점 및 개선 가능성을 파악하기 위해, 시스템 계층의 최적화 기법인 “SWAM”을[4] 적용한 시스템을 사용해 실험하였다. SWAM은 전체적인 저장장치 스왑 횟수 개선에 초점을 둔 기법으로 여러 응용에 의해 자주 사용되는 페이지들은 ZRAM으로 스왑 하고 그렇지 않은 각 응용 별 고유 페이지 위주로 저장장치로 스왑하여 저장장치 스왑인으로 인한 사용자 경험 저하를 방지한다. 실험은 안드로이드 기반 모바일 보드에서 유명 모바일 응용들을 실행하며 응용 시작 지연시간 중 스왑에 의한 지연시간, 응용 시작 구간 정보와 저장장치 스왑 횟수를 측정하였고, 자세한 실험 환경과 실험 방법은 4장에서 상술한 것과 동일하다.

사용자 경험에서 매우 중요한 요소인 포그라운드 응용의 시작 지연시간에서 스왑에 의한 지연시간을 분석하기 위해, 각 응용 별 시작 지연시간에서 (사용자가 응용을 실행한 시점부터 실 사용이 가능할 때까지의 지연시간) ZRAM 과 저장장치 스왑에 의한 지연시간을 세분화하여 그림 1 로 나타냈다. 각 응용 별로 전체 시작 지연시간에서 저장장치 스왑아웃에 의한 지연시간이 평균 19.1%, 저장장치 스왑인에 대한 지연시간이 평균 9.8% 차지한다. 시스템 계층에서의 최적화 기법을 통해 저장장치 스왑으로 인한 사용자 응답시간 저하를 최소화하였음에도 불구하고, 저장장치 스왑에 의한 지연시간이 사용자 응답시간에 여전히 상당부분을 차지하고 있음을 확인할 수 있다.

SWAM과 같은 시스템 계층에서의 최적화를 통해 사용자 응답 시간을 개선할 수 있지만, 저장장치 스왑이 필수적으로 수반되는 상황에서는 여전히 느린 저장장치의 응답시간이 사용자 경험을 저하시키는 요인이 된다. 저장장치 스왑으로 인한 사용자 응답시간 지연에 대한 이유를 구체적으로 파악하기 위해 시간에 따른 저장장치 기반 스왑아웃과 스왑인 횟수와 사용자가 새로운 응용을 실행하는 시점을 함께 그림 2로 나타냈다. 전체 시간에 걸쳐 저장장치 스왑이 지속적으로 발생하지만 포그라운드 응용의 시작 중에 스왑아웃과 스왑인이 더욱 더 빈번하게 발생하는 것을 확인할 수 있다. 어느 한 응용이 시작된 시점에 메모리가 부족하며 ZRAM 스왑 공간이 꽉찬 경우 필수적으로 저장장치 스왑을 사용하여 메모리를 회수하게 된다. 또한, 해당 응용과 관련된 데이터가 저장장치로 스왑아웃 되어 있는 경우 스왑된 데이터를 스왑인해야 하기 때문에 저장장치 스왑이 많이 발생하게 된다. 응용 시작 중 불가피하게 발생하는 저장장치 스왑 동작으로 인해 시스템 계층에서의 최적화 기법만으로는 저장장치 스왑으로 인한 사용자 경험 저하를 최소화하기에 어려움이 있으며 저장장치 내부적으로 사용자 경험 저하를 최소화할 수

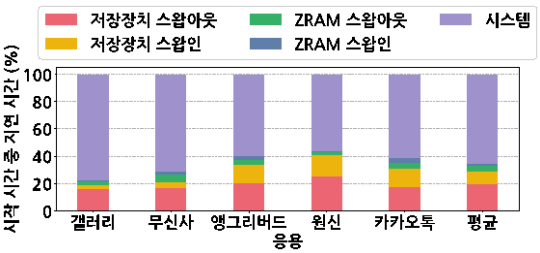

그림 1. 응용 별 응용 시작 지연시간 중 스왑에 의한 지연시간 분석

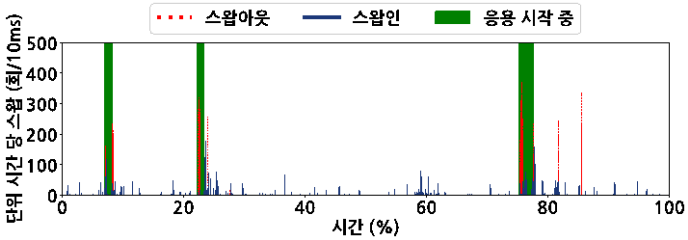

그림 2. 시간 당 저장장치 스왑아웃과 스왑인 횟수 분석

있는 방안이 필요하다.

모바일 저장장치의 느린 성능은 모바일 기기의 근본적인 공간, 전력 등 제약으로 인한 저장장치 내부 DRAM 메모리의 부재로 전통적인 SSD 와 같이 쓰기 버퍼, 미리 읽기와 같은 기법들이 적용될 수 없어 모든 읽기 및 쓰기 동작을 낸드 플래시에 직접적으로 수행하기 때문이다. 최근 모바일 저장장치는 한정된 크기에서 제공 가능한 용량을 늘리기 위해 TLC 를 (한 셀에 3 개의 비트 저장) 넘어 QLC 까지 (한 셀에 4 개의 비트 저장) 상용화되는 추세에 따라 저장장치의 성능은 더욱 더 악화될 수 있다. 한 셀에 더 많은 비트를 저장하게 되면 같은 하드웨어 자원에서 더 큰 용량을 제공하는 장점이 있지만 더욱 정교한 읽기 쓰기 작업으로 인해 저장장치의 성능은 더욱 느려지기 때문이다. 이와 같은 문제로 높은 성능을 보장해야 하는 스왑 데이터에 대해 저장장치 내부적으로 스왑 데이터에 대한 특별한 관리 기법 없이는 사용자 경험 저하를 방지할 수 없다.

느린 저장장치의 성능을 해결하기 위해 빠른 성능을 보장하는 SLC 를 (한 셀에 1 개의 비트 저장) 우선적으로 사용하고 추후에 TLC/QLC 로 마이그레이션하는 ‘Write Booster’ 기법이 제안된 바 있다.[7] SLC 를 사용하는 것이 성능 측면에서 무조건 적으로 더 유리해 보이나 TLC 혹은 QLC 모드를 사용하는 것 대비 같은 용량을 저장하기 위해 3 배 혹은 4 배 더 많은 셀을 사용해야 하는 단점이 있다. 또한, 모든 데이터에 대해 SLC 를 우선적으로 사용하게 되면 마이그레이션을 위한 메타데이터와 WAF 가 증가하기 때문에 항상 SLC 를 우선적으로 사용하는 것은 자원이 한정적인 모바일 저장장치의 특성상 공간 및 효율 측면에서 적합하지 않다.

3. 사용자 경험 향상을 위한 저장장치 스왑 관리 기법

3.1 제안하는 기법의 구조

그림 3 은 본 논문에서 제안하는 기법이 적용된 시스템의 전체 구조를 보여준다. 제안하는 기법은 크게 세 가지 모듈로 구성된다. 먼저 안드로이드 시스템과 리눅스 커널 사이에 포그라운드 응용의 정보 및 실행 상태 변화를 관찰하는 모니터와 각 응용 별 시작 중 사용된 페이지를 추적하는 트래커가 있고 저장장치 내부 컨트롤러에 스왑아웃하는 페이지의 정보를 커널의 모니터와 트래커로부터 전달받아 SLC 와 TLC 중 플래시 모드를 선택하는 셀렉터가 있다.

3.2 포그라운드 응용 시작 중 SLC 스왑아웃 기법

새로운 응용이 시작될 때 저장장치로의 스왑아웃 요청이 많이 발생한다. 이 경우 실제 저장장치가 전달받은 스왑 데이터를 플래시에 쓰기 요청을 완료할 때까지의 시간만큼 응용 시작 지연시간을 증가시켜 사용자 경험을 저하시키는 치명적인 요인이다.

제안하는 기법에서는 모니터가 포그라운드 응용 시작의 처음과 끝을 감지하여, 해당 정보를 저장장치 스왑아웃 시 전달되는 데이터의 bio에 표시하여 저장장치의 컨트롤러내 셀렉터에 전달한다. 정보를 전달받은

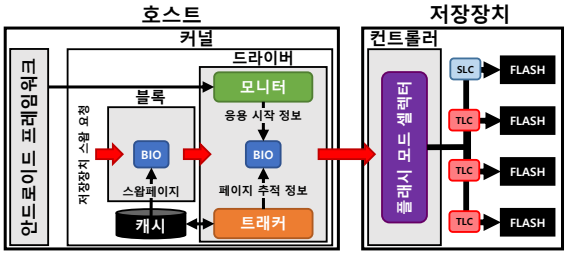

그림 3. 제안하는 기법의 전체 구조

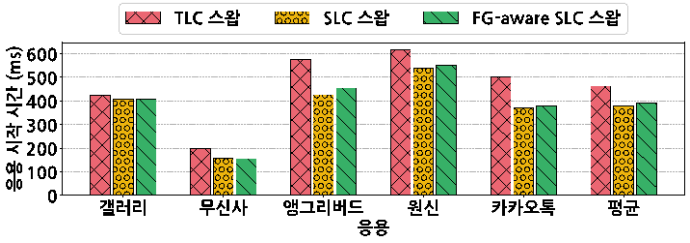

그림 4. 응용 별 시작 지연시간 기법 간 비교

셀렉터는 해당 데이터가 응용 시작 중 스왑아웃한 것으로 판단하면 SLC 모드를 사용하고, 그렇지 않은 경우 TLC 모드를 사용하여 스왑 데이터를 저장한다. 본 기법을 통해 포그라운드 응용 시작 중 발생한 스왑아웃을 빠르게 처리하여 메모리 회수 지연시간이 개선되어 저장장치 스왑으로 인한 사용자 응용 시작 지연시간을 개선할 수 있다.

3.3 포그라운드 응용의 재시작을 고려한 SLC 스왑 기법

모바일 환경에서 포그라운드 응용의 시작 지연시간을 최소화하기 위해서는 필요한 데이터가 메모리에 존재하는 것이 중요하다. 만약 필요한 데이터가 저장장치에 스왑되어 있는 경우 저장장치 스왑인으로 인해 사용자 경험이 저하되게 된다.

이를 해결하기 위해 각 응용이 시작할 때마다 사용하는 페이지들을 트래커 모듈을 통해 기억해두고, 이후 해당 페이지가 저장장치로 스왑될 경우 스왑되는 데이터의 bio에 표시하여 저장장치의 컨트롤러 내 셀렉터에 전달한다. 정보를 받은 셀렉터는 해당 데이터가 추후 응용 시작 중 사용될 데이터라고 판단하면 SLC 모드를 사용하고, 그렇지 않은 경우 TLC 모드를 사용하여 데이터를 저장한다. 본 기법을 통해 응용 재시작 중 필요한 페이지를 빠르게 스왑인하여 재시작 지연시간을 개선함으로써 사용자 경험을 향상시킬 수 있다. 또한 앞의 두 기법 모두 사용자 경험에 중요한 요소인 응용 시작과 관련된 경우만 SLC 모드를 사용했기 때문에 저장장치 내 공간 오버헤드도 완화할 수 있다.

4. 실험

4.1 실험 환경

본 논문에서 제안하는 기법의 실제 성능 향상 정도를 평가하기 위해 안드로이드 기반 snapdragon 888 HDK[8]를 사용하였다. 개발 보드의 호스트 메모리는 8GB, 낸드 플래시 저장장치는 Hynix P41 2TB SSD 로 구성했다. 그리고 호스트 메모리 중 25%인 2GB 를 ZRAM 으로 할당하고, SWAM 방식의 저장장치 스왑이 가능하도록 추가 구현하였다.

제안하는 기법들은 리눅스 커널과 AOSP 를 수정하여 구현하였고, 저장장치의 내부 컨트롤러 수정은 불가능하기 때문에 SLC 모드에서 읽기 25us, 쓰기 200us 의 지연시간을 갖고 TLC 모드에서는 읽기 75us, 쓰기 700us 의 지연시간을 갖도록 리눅스 커널의 디바이스 드라이버 계층을 수정하였다. 실험에는 기존 TLC 기반 저장장치 스왑, SLC 기반 저장장치 스왑과 본 연구가 제안하는 포그라운드 응용을 고려한 SLC 기반 저장장치 스왑 기법을 각각 응용 40 개를 사용하는 동일한 시나리오를 실제 동작시켜 비교하였다.

4.2 기법 평가

그림 4 는 기법 간 응용 별 시작 지연시간을 비교하고 있다. SLC 기반 저장장치 스왑을 수행한 경우 기존 TLC 기반 대비 시작 지연시간이 평균 21.7% 개선되었고, 카카오톡의 경우 35.6%까지 개선되는 것을 확인할 수 있었다. 본 연구가 제안하는 기법의 경우에는 포그라운드 응용에 해당하는 일부 데이터만을 SLC 모드로 스왑했음에도 불구하고, TLC 기반 저장장치 스왑 대비 시작 지연시간이 평균 18.8% 개선되었고, 카카오톡의 경우 32.5%까지 개선되어 SLC 기반 저장장치 스왑 기법의 경우와 거의 비슷한 성능을 보여주는 것을 확인할 수 있다.

SLC 스왑 사용에 의한 공간 오버헤드 평가를 위해 SLC 기반 저장장치 스왑 기법과 제안하는 기법의 저장장치 공간 사용량을 측정했다. 제안하는 기법의 경우 SLC 기반 스왑 기법 대비 평균 40% 적은 용량을 사용하였다. 제안하는 기법을 통해 더 적은 저장장치 용량을 사용하며 저장장치 스왑의 사용자 응답시간 증가로 인한 사용자 경험 저하를 최소화하였다.

5. 결론 및 향후 연구

본 논문에서는 모바일 저장장치 스왑의 시스템 계층 최적화의 한계점과 저장장치의 본질적인 문제점을 분석하고, 사용자 경험 향상을 위해 포그라운드 응용의 시작 지연시간을 개선하는 기법을 제안하였다. 실험 결과 포그라운드 응용 시작 지연시간이 평균 18.8% 개선되어 제안한 기법을 통해 저장장치 스왑으로 인한 사용자 경험 저하를 방지할 수 있음을 확인하였다. 본 연구는 시작 지연시간 개선에만 초점을 두었지만, 응용 실행 중에도 스토리지 스왑은 빈번하게 발생한다. 따라서 스왑 페이지의 패턴 또는 타입에 따라 적절하게 SLC, TLC 를 선택하여 응용 실행 지연시간도 개선하는 연구를 진행할 계획이다.

참 고 문 헌

[1] ZRAM, https://developer.android.com/topic/performance/memory-management, 2023.
[2] Samsung Ramplus, https://www.samsung.com/sg/support/mobile-devices/what-is-ram-plus-and-how-to-use-it, 2022.
[3] Niel Lebeck et al, “End the Senseless Killing: Improving Memory Management for Mobile Operating Systems”, ATC, 2020.
[4] Geunsik Lim et al, “SWAM: Revisiting Swap and OOMK for Improving Application Responsiveness on Mobile Devices”, MobiCom, 2023.
[5] Weichao Guo et al, “MARS: Mobile Application Relaunching Speed-Up through Flash-Aware Page Swapping”, IEEE Transactions on Computers, vol.65, pp.916-928, 2016.
[6]LI, Changlong et al. “SEAL: User experience-aware two-level swap for mobile devices”, TCADICS, vol.39, pp.4102-4114, 2020.
[7] KIOXIA, “Understanding the WriteBooster Feature”, 2022.
[8] Snapdragon 888 HDK, https://developer.qualcomm.com/hardware/snapdragon-888-hdk, 2021.