The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 4, pp.29-33, Aug. 31, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.4.29
JIIBC 2024-4-5

모바일 시스템의 고속 스왑 지원을 위한 메모리 할당 및 회수 기법

Memory Allocation and Reclamation Policies for Fast Swap Support in Mobile Systems

반효경*

Hyokyung Bahn*

요 약 최근 스마트폰을 통해 다양한 앱이 실행되면서 모바일 시스템의 메모리 요구량이 지속적으로 증가하고 있다. 메모리 공간이 소진될 경우 프로세스의 풋프린트를 스토리지의 스왑 영역에 백업하는 데스크탑과 달리, 스마트폰은 앱을 종료시켜 상당 부분의 문맥이 유실된다. 이는 스마트폰에서 스왑 지원 시 플래시메모리로의 대용량 I/O 작업이 심각한 지연을 초래하기 때문이다. 본 논문에서는 쓰기 연산이 플래시메모리에 비해 빠른 eMRAM을 모바일 시스템의 스왑 영역으로 사용할 경우 어떻게 효율적인 메모리 관리를 수행할 수 있는지에 대해 논의한다. 백업 스토리지가 플래시메모리인 파일시스템 영역과 eMRAM인 스왑 영역의 특성과, 해당 페이지의 참조 특성을 고려한 메모리 할당 및 회수를 통해 스마트폰의 I/O 성능을 평균 15% 개선할 수 있음을 보인다.

Abstract Recent advancements in mobile apps have led to continuously increasing memory demands on smartphone systems. Unlike desktops, which use swap functions to backup the entire memory footprint to storage when memory space is exhausted, smartphones terminate apps and lose significant context. This occurs because large-scale I/O operations to flash memory cause severe delays when swap is enabled on smartphones. This paper discusses how efficient memory management can be performed by using eMRAM, which is faster in write operations than flash memory, as the swap area in mobile systems. Considering the characteristics of backup storage (i.e., flash memory for the file system and eMRAM for the swap areas) as well as the reference characteristics of each page, we demonstrate that the proposed memory allocation and reclamation policies can improve the smartphone's I/O performance by an average of 15%.

Key Words : smartphone, memory allocation, memory reclamation, swap, eMRAM

*정회원, 이화여자대학교 컴퓨터공학과
접수일자 2024년 7월 6일, 수정완료 2024년 7월 26일
게재확정일자 2024년 8월 9일

Received: 6 July, 2024 / Revised: 26 July, 2024 /
Accepted: 9 August, 2024
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

Ⅰ. 서 론

모바일 플랫폼 기술의 발전으로 카메라, 센서, GPS 등을 활용한 수 많은 서비스들이 스마트폰을 통해 지원되고 있다[1, 2, 3]. 스마트폰의 하드웨어는 다양한 앱들의 동시 실행을 지원하기에 충분한 수준에 이르렀으며, 최신 안드로이드 레퍼런스 폰인 구글 픽셀 8의 경우 9 코어의 1.70~2.91 GHz CPU, 10 코어의 Mali-G715 GPU, 8 GB의 RAM, 256 GB의 UFS 3.1 스토리지를 갖추어 멀티태스킹을 수행하기에 충분한 수준에 이르렀다.

특히, 스마트폰은 비디오 촬영 및 실시간 방송 등 개인화된 소셜 미디어 역할을 수행할 뿐 아니라, 최근에는 인공지능 기능을 수행하는 사례 또한 늘고 있다[4, 5, 6]. 이러한 앱들의 멀티태스킹으로 인해 스마트폰의 메모리 요구량 또한 크게 늘어나고 있다. 한편, 스마트폰은 여러 앱의 동시 실행으로 메모리 용량이 부족할 때 플래시 스토리지로 스왑을 하는 대신 앱을 종료시킨 후 추후 재시작하는 방식을 택하고 있다[7]. 전통적인 컴퓨터 시스템과 달리 스마트폰에서 스왑을 지원하지 않는 것은 메모리 풋프린트 전체를 스토리지에 저장하는 스왑보다는 개별 앱 차원에서 필요한 문맥만을 파일에 백업하는 방식이 오버헤드가 적고 빠른 응답성을 제공할 수 있기 때문이다. 하지만, 이는 높은 신뢰성이 요구되는 분야에서 앱의 문맥이 유실되는 문제점을 발생시킬 수 있다.

스마트폰에서 스왑을 사용했을 때의 또다른 문제는 스왑을 위한 빠른 백업을 수행하기에 플래시메모리가 심각한 쓰기 지연을 발생시킨다는 점이다. 이를 해결하기 위해 플래시메모리가 아닌 고속 스토리지 매체인 NVM을 이용한 스왑 연구가 진행된 바 있다[8, 9]. 본 논문은 이러한 기존 연구와 두 가지 점에서 차별화된다. 첫째, 기존 연구는 NVM을 이용한 스왑 자체에 초점을 맞춘 반면, 본 연구는 이러한 시스템의 메모리 관리가 어떻게 이루어져야 하는지에 초점을 맞춘다. 메모리 내의 페이지들은 그 백업 스토리지가 플래시메모리인 파일시스템 페이지와 NVM인 스왑 페이지로 양분되므로, 본 논문은 이들의 접근 패턴과 I/O 비용을 고려한 메모리 할당 및 회수 기법을 다룬다. 둘째, 기존 연구는 NVM 매체로 PCM을 주로 사용하고자 하나 최근들어 PCM의 모바일 시장 진입은 부정적인 상황이 되었으며, 이에 본 연구에서는 임베디드용 NVM인 eMRAM을 스왑 장치로 사용하고자 한다. 2024년 현재 삼성전자가 eMRAM의 14 나노미터 공정 개발을 완료했고, 8나노 공정도 마무리단계인 것으로 보고되고 있어 본 연구의 내용은 기존 PCM 기반 스왑에 비해 실현 가능성이 높다고 볼 수 있다. 트레이스 기반 시뮬레이션을 통해 제안한 기법이 스왑 지원 스마트폰의 I/O 성능을 평균 15%, 최대 35%까지 개선함을 보인다.

Ⅱ. 메모리 할당 및 회수 기법 설계

본 논문의 파일시스템은 기존 스마트폰 세팅대로 플래시메모리에 배치하고 새롭게 추가된 eMRAM은 스왑 영역으로 사용한다. 따라서, 메모리에 존재하는 페이지는 플래시메모리에서 올라온 페이지와 eMRAM에서 올라온 페이지로 구분되며, 메모리 영역에 대한 할당과 회수도 이 두 매체의 특성을 반영하는 것이 필요하다. 파일시스템 영역에 있는 페이지가 요청될 때, 해당 페이지는 플래시메모리에서 읽어오게 되며, 이러한 페이지를 파일 매핑 페이지라고 부른다. 반면, 스왑 영역에 있는 페이지가 요청되는 경우 eMRAM에서 읽어오게 되며, 이러한 페이지를 익명 페이지라고 부른다. 플래시메모리와 eMRAM의 읽기 쓰기 성능이 서로 다를 뿐 아니라, 파일 매핑 페이지와 익명 페이지의 접근 패턴이 다르기 때문에 효율적인 메모리 관리를 위해서는 각 메모리 페이지의 참조 특성, 그리고 이들의 I/O를 위한 스토리지의 읽기 쓰기 성능을 고려한 방법이 필요하다[10, 11]. 그림 1은 본 논문이 제안하는 메모리 할당 및 회수 기법의 기본 구조를 보여주고 있다.

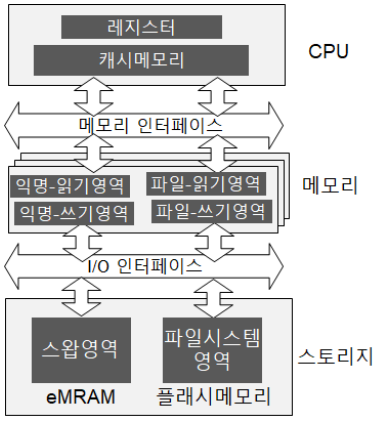

그림 1. 제안하는 기법의 기본 구조
Fig. 1. Basic architecture of the proposed policy.

그림에서 eMRAM은 단위 공간당 비용이 플래시메모리에 비해 높기 때문에 스토리지 전체를 대체하기는 어

려우며, 따라서 본 아키텍처에서는 플래시메모리 기반 파일시스템을 그대로 유지하고, 스왑 장치로 eMRAM을 추가해서 사용한다. 한편, eMRAM과 플래시메모리의 성능 격차로 인해, 페이지를 파일시스템으로부터 인출하는 비용이 스왑 영역으로부터 인출하는 비용보다 높다. 따라서, 메모리 상에서 파일시스템 페이지를 회수했는데 이를 플래시메모리에서 다시 읽어와야 하는 경우 상대적으로 더 높은 비용이 소요된다. 본 논문에서는 이를 감안해서 다른 모든 조건이 동일할 경우 파일 매핑 페이지의 메모리 내 우선 순위를 익명 페이지에 비해 더 높게 책정한다. 또한, 스토리지에서 읽기와 쓰기를 수행하는 비용이 이질적이므로 이 또한 알고리즘의 설계에 반영한다.

제안하는 기법은 메모리 내의 페이지 참조를 파일-읽기, 파일-쓰기, 익명-읽기, 익명-쓰기로 분류한 다음 메모리 공간을 논리적으로 파일-읽기 영역, 파일-쓰기 영역, 익명-읽기 영역, 익명-쓰기 영역으로 분할해서 관리한다. 이때, 각 영역의 크기는 접근 빈도와 I/O 비용을 고려해서 동적으로 조절된다. 영역의 크기를 조절했을 때 성능 개선 혹은 성능 저하가 어느 정도인지 모니터링하기 위해 각 영역에서 회수된 페이지의 기록을 일정 기간 유지하는 용도로 그림자 영역을 둔다. 그림자 영역은 4종의 각 영역별로 두며 해당 영역에서 최근 회수된 페이지가 어떤 것인지를 한시적으로 보관하며 실제 페이지의 내용은 저장하지는 않는다[12].그림자 영역에서의 페이지 참조 정보를 통해 해당 영역의 크기를 확대했을 때 얻게 되는 성능 개선 효과를 예상할 수 있다. 예를 들어 익명-읽기 영역의 그림자 영역에 속한 페이지가 빈번히 참조되는 경우 제안한 기법은 익명-읽기 영역을 확장하여 이 영역으로 인해 발생하는 I/O를 줄인다. 그림자 영역에서의 참조량뿐 아니라 스토리지(플래시메모리, eMRAM) 및 연산(읽기, 쓰기)의 성능 차이를 고려해서 각 영역의 크기 확장을 결정한다.

그림 2는 각 영역과 해당 그림자 영역의 모습을 보여주고 있다. 한편, 동일 페이지에 대해 읽기와 쓰기 연산이 모두 발생할 수 있으므로 물리적으로는 하나의 페이지가 메모리에 할당되더라도 해당 페이지에 대한 헤더는 읽기 및 쓰기 영역에 동시에 포함될 수 있다. 따라서, 페이지 헤더가 어느 영역에도 속하지 않는 경우 물리적인 메모리에서 해당 페이지가 회수될 수 있다.

각 영역에서 페이지의 회수가 필요한 경우 제안하는 기법은 기존의 CLOCK 알고리즘처럼 최근 참조 여부를 나타내는 1비트의 정보를 활용해서 해당 영역에서 회수할 페이지를 결정한다. 예를 들어 익명-읽기 영역에서 페이지 회수가 필요한 경우 페이지들의 참조비트를 순차적으로 체크하면서 비트가 1인 경우 최근에 참조된 페이지이므로 회수하는 대신 비트를 0으로 클리어하고, 비트가 0인 페이지를 발견하면 해당 페이지를 영역에서 방출한다. 방출된 페이지는 일정 기간 그림자 영역에 존재하게 된다. 그림자 영역에서 페이지 회수가 필요할 때에는 가장 오래된 페이지를 방출한다. 읽기 영역이 아닌 쓰기 영역에서의 페이지 회수는 쓰기 연산에 대한 최근 기록을 활용하는 것이 더 정확한 정보를 제공하므로 참조비트가 아닌 수정비트를 클리어시키는 방식을 통해 최근에 쓰기 연산이 발생하지 않은 페이지를 회수하는 방법을 사용한다[12].

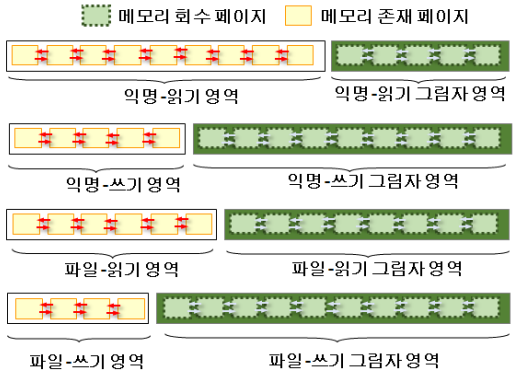

그림 2. 제안하는 기법의 메모리 영역 할당 방식
Fig. 2. Memory area allocation of the proposed policy.

III. 성능 평가

본 장에서는 시뮬레이션을 통해 제안한 기법의 성능을 평가한다. 안드로이드 앱의 메모리 참조 트레이스를 Valgrind 툴의 Cachegrind 유틸리티를 통해 수집한 후 이를 재현하는 실험을 진행했으며, 제안한 기법과 CLOCK 알고리즘이 탑재된 시스템의 I/O 시간을 비교하였다. 한편, 수집된 트레이스에는 명령어-읽기, 데이터-읽기, 데이터-쓰기의 구분이 존재하며, 이들은 본 실험에서 파일-읽기, 익명-읽기, 익명-쓰기로 변환하여 실험을 진행하였다. 본 실험에서 파일-쓰기는 성능 평가 대상에서 배제했는데, 이는 파일 매핑 페이지의 경우 메모리에 쓰기가 발생한 후 페이지 회수 시점까지 스토리지 쓰기가 지연되는 것이 아니라 저널링 또는 플러시 데몬에 의해 곧바로 스토리지에 반영되므로 페이지 회수 알고리즘의 성능과 무관하기 때문이다. 이와 달리 익명 페이지에 대한 메

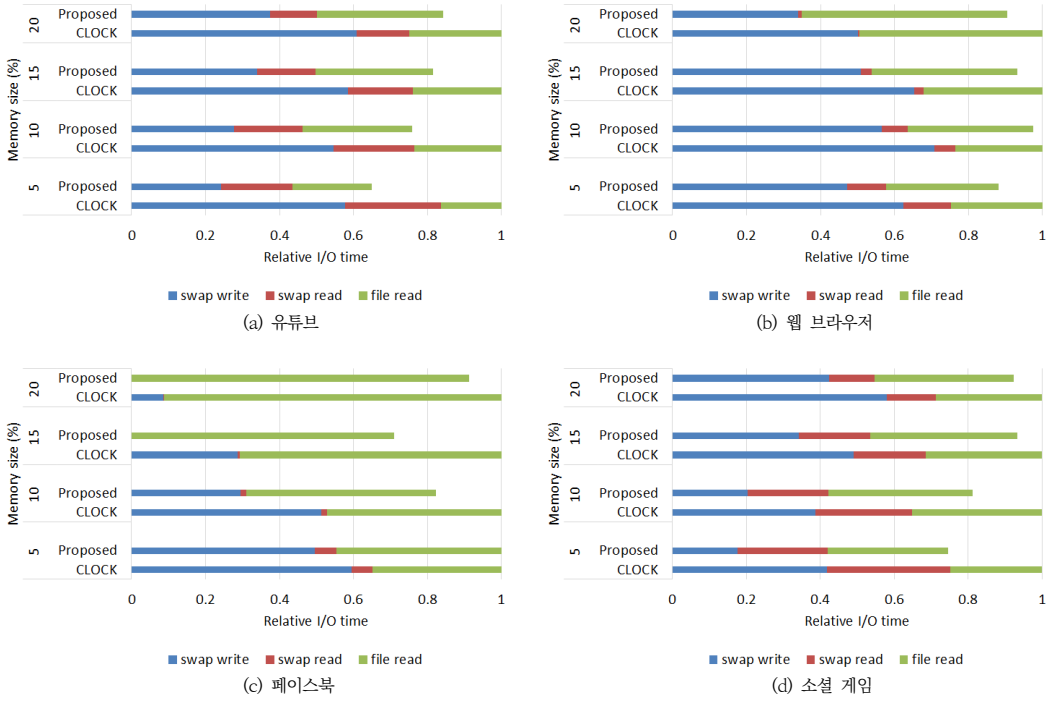

그림 3. CLOCK 알고리즘과의 성능 비교
Fig. 3. Performance comparison with CLOCK

모리 쓰기는 메모리에서 회수되는 시점에 스토리지 쓰기가 발생하므로 페이지 회수 알고리즘의 우수성과 밀접한 관련이 있다.

그림 3은 메모리 크기가 전체 워크로드 풋프린트의 5%에서 20%까지 변함에 따라 CLOCK 알고리즘과 제안한 기법의 I/O 시간을 상대적인 스케일로 보여주고 있다. 그림에서 보는 것처럼 제안한 기법은 모든 경우에 있어 CLOCK 알고리즘보다 우수한 성능을 나타내었다. 이는 각 영역의 참조 빈도와 각 저장 매체의 읽기 쓰기 성능을 종합적으로 고려한 메모리 할당 및 회수가 이루어졌기 때문으로 볼 수 있다. 구체적으로, 제안한 기법은 CLOCK에 비해 평균 15%, 최대 35%까지 I/O 시간을 개선한 것을 확인할 수 있었다. 특히, 대부분의 경우 메모리 크기가 상대적으로 작을 때 제안한 기법의 효과가 명확하게 나타났으며, 이는 한정된 메모리 용량을 효율적으로 관리하는, 비용 효율적인 공간 관리 방식에 기인한 것으로 볼 수 있다.

그림 3에서 또 다른 흥미로운 부분은 모든 경우에 있어 스왑 쓰기가 스왑 읽기에 비해 더 많은 시간 비중을 차지한다는 점이다. 이는 메모리 공간이 부족할 때 많은 수의 페이지가 스왑 영역에 쓰기를 발생시키고 메모리에서 회수되지만, 그 중 지극히 한정적인 페이지만 재사용되어 스왑 영역으로부터 다시 읽기를 발생시킨다는 점을 의미한다. 한편, 웹 브라우저와 페이스북의 경우 메모리 크기 증가에 따라 스왑 I/O의 비율이 급격히 감소하는 것을 확인할 수 있다. 이는 이 두 앱의 메모리 접근 지역성이 높고, 따라서 메모리 크기가 일부 제한된 핫 페이지를 수용할 만큼 크면 더 이상 스왑이 발생하지 않기 때문이다. 이 경우, 메모리 크기가 증가함에 따라 파일 읽기의 비율이 크게 증가하지만, 이는 그림 3에서 I/O 시간을 상대적 척도로 나타내었기 때문이며, 실제 I/O 시간의 증가를 의미하지는 않는다.

IV. 결 론

스마트폰이 다양한 서비스를 제공하면서 스마트폰의 메모리 용량에 대한 요구가 지속적으로 증가하고 있다. 본 논문에서는 스마트폰의 메모리가 부족할 때 성능 상의 이유로 앱을 종료시키는 기존 방식 대신 eMRAM을 탑재하고 이를 스왑 장치로 운영하기 위한 시스템 구조와 메모리 관리 기법에 대해 논의하였다. 특히, 본 논문

에서는 파일 시스템 영역과 스왑 영역에서 참조된 페이지의 특성뿐 아니라 플래시메모리와 eMRAM의 읽기 쓰기 비용을 고려한 페이지 할당 및 회수 기법을 연구하였다. 다양한 안드로이드 앱의 메모리 참조 트레이스를 재현하는 시뮬레이션 실험을 통해 제안한 메모리 관리 기법이 스왑을 지원하는 스마트폰의 I/O 성능을 크게 개선할 수 있음을 확인하였다.

References

[1] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep Learning on Mobile and Embedded Devices: State-of-the-art, Challenges, and Future Directions," ACM Computing Surveys, Vol. 53, Iss. 4, No. 84, pp. 1-37, 2021.
DOI: https://doi.org/10.1145/3398209

[2] D. Kim, S. Lee, and H. Bahn, "An adaptive location detection scheme for energy-efficiency of smartphones," Pervasive and Mobile Computing, vol. 31, pp. 67-78, 2016.
DOI: https://doi.org/10.1016/j.pmcj.2016.04.012

[3] S. Ki, G. Byun, K. Cho and H. Bahn, "Co-Optimizing CPU Voltage, Memory Placement, and Task Offloading for Energy-Efficient Mobile Systems," IEEE Internet of Things Journal, Vol. 10, No. 10, pp. 9177-9192, 2023.
DOI: https://doi.org/10.1109/JIOT.2022.3233830

[4] J. Wang, B. Cao, P. Yu, L. Sun, W. Bao and X. Zhu, "Deep Learning towards Mobile Applications," Proc. IEEE ICDCS, pp. 1385-1393, 2018.
DOI: https://doi.org/10.1109/ICDCS.2018.00139

[5] S. Kim, W. Hur, and J. Ahn, "A progressive web application for mobile crop disease diagnostics based on transfer learning," Journal of the Korea Academia-Industrial cooperation Society (JKAIS), vol. 23, no. 2 pp. 22-29, 2022.
DOI: https://doi.org/10.5762/KAIS.2022.23.2.22

[6] K. Lee, H. Jung, and S. Lee, "Anomaly detection method of user trajectories based on deep learning technologies," JKIIT, vol. 20, no. 11, pp. 101-116, 2022.
DOI: https://doi.org/10.14801/jkiit.2022.20.11.101

[7] S. Kim, J. Jeong, J. Kim, and S. Maeng, "SmartLMK: a memory reclamation scheme for improving user-perceived app launch time," ACM Transactions on Embedded Computing Systems, vol. 15, no. 47, Article 25, 2016.
DOI: https://doi.org/10.1145/2894755

[8] J. Kim and H. Bahn, "Analysis of smartphone I/O characteristics — toward efficient swap in a smartphone," IEEE Access, vol. 7, pp. 129930-129941, 2019.
DOI: https://doi.org/10.1109/ACCESS.2019.2937852

[9] D. Liu, K. Zhong, X. Zhu, Y. Li, L. Long, and Z. Shao, "Non-volatile memory based page swapping for building high-performance mobile devices," IEEE Transactions on Computers, vol. 66, no. 11, pp. 1918-1931, 2017.
DOI: https://doi.org/10.1109/TC.2017.2711620

[10] O. Kwon, H. Bahn, and K. Koh, "Popularity and prefix aware interval caching for multimedia streaming servers," Proc. IEEE Conf. on Computer and Information Technology, pp. 555-560, 2008.
DOI: https://doi.org/10.1109/CIT.2008.4594735

[11] T. Kim and H. Bahn, "Implementation of the storage manager for an IPTV set-top box," IEEE Transactions on Consumer Electronics, vol. 54, no. 4, pp. 1770-1775, 2008.
DOI: https://doi.org/10.1109/TCE.2008.4711233

[12] H. Lee, H. Bahn, and K. Shin, "Page replacement for write references in NAND flash based virtual memory systems," Journal of Computing Science and Engineering, vol. 8, no. 3, pp. 157-172, 2014.
DOI: https://doi.org/10.5626/jcse.2014.8.3.157

저 자 소 개

반 효 경(정회원)

- 1997년 2월 : 서울대학교 계산통계학과 학사
- 1999년 2월 : 서울대학교 전산과학과 석사
- 2002년 2월 : 서울대학교 컴퓨터공학부 박사
- 2002년 9월 ~ : 이화여자대학교 컴퓨터공학과 교수
- 주관심분야 : 운영체제, 스토리지시스템, 임베디드시스템