The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 2, pp.61-66, Apr. 30, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.2.61
JIIBC 2023-2-8

스왑 지원 스마트폰의 메모리 쓰레싱 분석 및 관리 방안

Analysis and Management Policies for Memory Thrashing of Swap-Enabled Smartphones

반효경*, 김지선**

Hyokyung Bahn*, Jisun Kim**

요 약 스마트폰의 기능이 다양한 영역으로 확대되고 멀티태스킹이 활성화됨에 따라 스왑 기능의 지원이 점점 중요한 이슈로 부각되고 있다. 그러나, 스마트폰에서의 스왑 지원은 지나친 스토리지 트래픽을 유발하여 메모리 쓰레싱을 발생시키는 것으로 알려져 있다. 본 논문에서는 초창기 스마트폰의 스왑 지원시 발생하던 이러한 쓰레싱 현상이 스마트폰 하드웨어가 발전함에 따라 어떻게 변화하였는지를 분석한다. 분석 결과 메모리 용량이 늘어남에 따라 스왑으로 인한 쓰레싱 문제가 일정 부분 해소되는 것을 확인하였다. 그러나, 실행 앱의 수를 지속적으로 증가시킬 경우 쓰레싱은 여전히 발생하는 것을 확인하였다. 본 논문에서는 이러한 쓰레싱의 유발이 일부 핫 데이터에 기인한다는 것을 보이고 이를 NVM 기반의 아키텍처를 통해 해결할 수 있는 방안을 제시한다. 특히, 소량의 NVM으로 효율적인 관리를 통해 스왑 기능을 지원하면서 성능 저하 문제를 해소할 수 있음을 보인다.

Abstract As the use of smartphones expands to various areas and the level of multitasking increases, the support of swap is becoming increasingly important. However, swap support in smartphones is known to cause excessive storage traffic, resulting in memory thrashing. In this paper, we analyze how the thrashing of swaps that occurred in early smartphones has changed with the advancement of smartphone hardware. As a result of this analysis, we show that the swap thrashing problem can be resolved to some extent when the memory size increases. However, we also show that thrashing still occurs when the number of running apps continues to increase. Based on further analysis, we observe that this thrashing is caused by some hot data and suggest a way to solve this through an NVM-based architecture. Specifically, we show that a small size NVM with judicious management can resolve the performance degradation caused by smartphone swap.

Key Words : Smartphone, swap, thrashing, Android, mobile platform, NVM

*정회원, 이화여자대학교 컴퓨터공학과
**비회원, 이화여자대학교 컴퓨터공학과
접수일자 2023년 3월 11일, 수정완료 2023년 3월 30일
게재확정일자 2023년 4월 7일

Received: 11 March, 2023 / Revised: 30 March, 2023 /
Accepted: 7 April, 2023
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

Ⅰ. 서 론

최근 다양한 기능의 스마트폰 앱이 등장하면서 스마트폰의 멀티태스킹이 점점 중요해지고 있다[1, 2, 3]. 그러나, 안드로이드와 같은 모바일 플랫폼은 시스템 내의 메모리가 부족할 때 프로세스의 문맥을 시스템 차원에서 유지하지 않고 강제로 종료하는 방식을 채택하고 있다[4]. 이는 사용자의 동의 하에 이루어지는 방식이 아니므로, 특정 앱을 시작한 후 오랜 시간이 지나서 해당 앱에 되돌아갔을 때 프로세스의 문맥이 사라질 수 있음을 뜻한다. 초창기 스마트폰에서는 이와 같은 방식이 큰 문제가 되지 않았으나, 최근 스마트 폰의 기능이 엔터테인먼트에 머무르지 않고, 오피셜한 작업 또한 스마트폰에서 수행하는 사례가 늘면서 사용자에게 시스템의 신뢰성을 떨어뜨리는 요소가 된다. 이러한 문제를 해결하기 위해 일부 앱들은 문맥을 스스로 저장했다가 종료 후 재시작 시 저장 문맥을 복원하는 방식을 채택하고 있다. 그러나, 이러한 방식은 운영체제가 아닌 앱 차원에서 이루어지기 때문에 문맥 저장에 한계가 있으며, 많은 앱들은 이러한 기능마저 제공하지 않는다.

앱을 종료시키는 대신 데스크탑 시스템의 경우 스왑 기능을 제공하여 메모리 부족 문제를 해결한다[5]. 스마트폰이 점점 멀티태스킹 기능을 가진 범용 컴퓨터 역할을 흡수하면서 스왑을 제공하는 것은 필수불가결한 상황이 되고 있다. 그러나, 스마트폰에서 스왑을 지원할 경우 많은 양의 스토리지 입출력을 발생시켜 앱의 실행시간을 크게 증가시키는 쓰레싱(thrashing) 현상이 발생하는 것으로 알려진 바 있다[6].

본 연구에서는 안드로이드 환경에서 스왑의 오버헤드를 분석하고 스마트폰 시스템이 발전함에 따라 이러한 스왑의 오버헤드가 어떤 식으로 변화했는지를 분석한다. 분석 결과 메모리 용량이 늘어날 경우 스왑으로 인해 발생하던 쓰레싱 문제가 일정 부분 해소되는 것을 확인하였다. 그러나, 실행 앱의 수가 증가함에 따라 이러한 쓰레싱은 여전히 발생하는 것을 확인하였다. 이와 달리 스왑이 비활성화된 표준 안드로이드 기기에서는 실행 앱의 수가 증가하더라도 쓰레싱이 발생하지 않는 것을 확인하였다. 본 논문에서는 스왑 지원 안드로이드에서 발생하는 쓰레싱이 일부 핫 데이터에 기인한 것을 정량적으로 분석하였으며, 소량의 NVM(non-volatile memory)을 사용할 경우 쓰레싱 문제를 상당 부분 해소할 수 있음을 워크로드 분석을 통해 보인다[7,8].

본 논문의 이후 구성은 다음과 같다. II장에서는 스왑 지원 스마트폰의 I/O 분석을 통해 쓰레싱 현상을 발생시키는 데이터의 특성을 분석한다. III장에서는 실측을 통해 스왑으로 인한 성능 저하와 그 해결책을 알아본다. 끝으로 IV장에서는 본 논문의 결론을 제시한다.

Ⅱ. 스왑 지원 스마트폰의 I/O 분석

본 장에서는 안드로이드 커널이 스왑을 지원하도록 빌드한 버전과 스왑을 지원하지 않는 오리지널 안드로이드에서 동일한 앱들을 수행하면서 발생하는 스토리지 I/O의 특성을 분석한다. 기존 연구에서 사용된 응답성이 중요한 9개의 게임 앱 Candy Crush, Clash Royale, 8 Ball Pool, Traffic Rider, Slither.io, Subway Surfers, Pokemon Go, 1010!, Piano Tiles 2 등을 반복 실행하였으며[6], 오리지널 안드로이드에서는 앱이 강제 종료 후 재시작되고 스왑 지원 안드로이드에서는 스왑 아웃후 스왑 인이 발생하도록 하였다. 하드웨어의 발전에 의한 영향을 확인하기 위해 Odroid-Q, Nexus5, Pixel 등의 레퍼런스 디바이스에서 1GB와 2GB 메모리로 실험한 결과 기기 자체보다 메모리 크기에 따른 차이가 확연히 드러났다.

먼저 메모리 참조의 편향성 분석에서 스왑 미지원 안드로이드의 경우 메모리 크기에 무관하게 상위 50%의 데이터가 전체 스토리지 접근의 약 80%를 대변하는 약한 편향성을 나타내었다. 이와 달리, 스왑 지원 안드로이드의 스토리지 접근은 일부 핫 데이터에 집중되는 것을 확인할 수 있었으며, 이같은 추세는 메모리 크기가 작을 때 더욱 뚜렷이 나타났다. 메모리 크기가 1GB일 경우 상위 10%의 데이터가 전체 스토리지 참조의 80%를 대변하는 강한 편향성을 나타내었다. 반면 메모리 크기가 2GB인 경우 동일한 80%의 스토리지 접근을 발생시키는 데이터가 상위 약 25%로, 편향성은 다소 약화되는 것을 확인할 수 있었다.

사실상 시스템이 정상적인 상태에서는 이러한 핫 데이터가 메모리에 보존되어야 하며 스왑의 대상이 되지 않아야 할 것이다. 이는 안드로이드가 그 하단에 리눅스 커널을 탑재하므로 비활성 페이지만을 메모리에서 쫓아내고 핫 데이터는 메모리에 상주하기 때문이다. 안드로이드의 스왑 I/O를 좀 더 정밀하게 분석한 결과 이러한 핫 데이터의 정체는 공유 라이브러리와 일부 안드로이드 서비스인 것으로 확인되었다. 수행 중인 앱의 수가 증가함에 따라 이러한 핫 데이터 또한 메모리에서 쫓겨나고 다

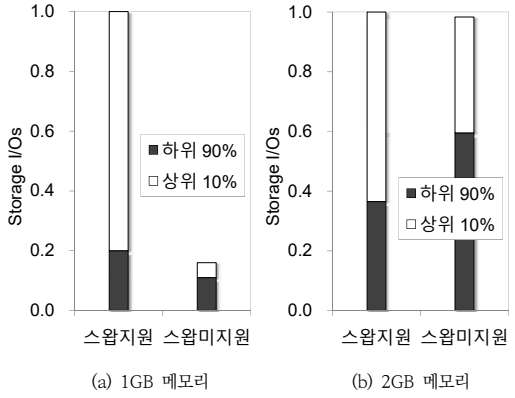


그림 1. 스왑 지원 여부에 따른 스토리지 입출력 분포
Fig. 1. Storage I/O distributions in swap on/off systems.

시 로드되는 쓰레싱 현상이 발생한 것이다. 그와 달리 스왑 미지원 안드로이드에서는 실행 앱의 증가로 메모리 빈 공간이 부족할 때 오래 전에 사용된 앱을 강제 종료시키는 방식을 사용하므로 이러한 현상이 발생하지 않았다.

그림 1은 메모리 크기가 1GB에서 2GB로 변함에 따라 스왑 지원 안드로이드와 스왑 미지원 안드로이드에서 스토리지 I/O가 어떻게 달라지는지를 보여주고 있다. 그림에서 보는 것처럼 스왑 지원 안드로이드가 더 많은 스토리지 접근량을 발생시켰으며, 특히 1GB 메모리 하에서는 스왑 미지원 안드로이드 대비 6-7배의 스토리지 접근량을 나타내었다. 이는 스왑 기능의 지원을 위해 앱의 메모리 데이터를 스토리지에 저장하고 이를 다시 복원하는 데에 많은 I/O를 필요로 하기 때문이다. 이에 비해 스왑 없이 앱을 종료시키고 재시작하는 데에는 대부분의 작업을 메모리에서 직접 수행하여 I/O가 훨씬 적게 발생했다. 그러나, 메모리 크기가 2GB인 경우 스왑으로 인한 스토리지 접근량은 거의 늘지 않음을 확인할 수 있다. 그림 1(b)에서 보는 것처럼 스왑 지원 안드로이드가 추가적으로 발생시키는 스토리지 트래픽은 미미한 수준임을 확인할 수 있다.

한편, 그림 1에서 상위 10%의 데이터가 발생시키는 스토리지 접근과 그렇지 않은 접근을 나누어 표시한 결과 스왑 지원 안드로이드에서 쓰레싱을 유발하는 데이터의 특성을 확인할 수 있었다. 메모리가 1GB일 때 스왑 지원 안드로이드는 상위 10%의 데이터가 전체 스토리지 접근량의 80%를 차지하고 있어 이러한 10%의 데이터가 발생시키는 반복적인 I/O를 해결할 경우 스왑 미지원 안드로이드와 유사한 수준의 I/O만 발생하여 쓰레싱을 해결할 수 있음을 뜻한다.

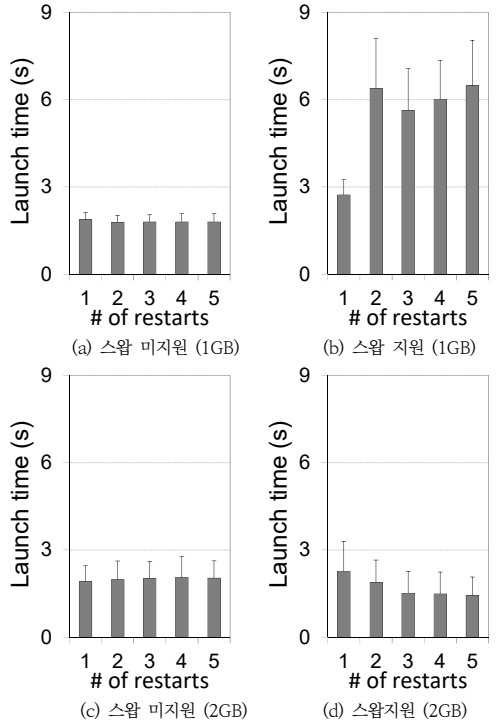


그림 2. 반복 수행에 따른 런치 시간 분포
Fig. 2. Launch time distributions for repeated executions.

III. 스왑 I/O 성능 측정 및 관리

본 장에서는 안드로이드 스왑의 오버헤드를 실측을 통해 확인하고 이를 해소할 수 있는 방안에 대해 논의한다. 그림 2는 II장과 동일한 방법으로 일련의 앱들에 대한 런치를 5회 반복했을 때 스왑 지원 안드로이드와 스왑 미지원 안드로이드의 회차별 런치 시간의 평균 및 표준편차를 보여주고 있다. 그림에서 보는 것처럼 앱의 최초 실행 시에는 스왑 지원 및 미지원 안드로이드의 런치 시간이 2-3초 정도로 큰 차이가 없는 것을 확인할 수 있다. 그러나, 앱의 실행 횟수가 증가함에 따라 스왑 미지원 안드로이드의 경우 2초 정도의 런치 시간이 일관되게 소요되는 반면, 스왑 지원 안드로이드의 경우 1GB 메모리 하에서는 런치 시간이 5-7초로 증가하고 그 편차도 큰 것을 확인할 수 있다. 이러한 결과는 II장에서 소개한 것처럼 스토리지 접근량이 크게 늘어났기 때문으로 추정할 수 있다.

한편, 메모리 크기가 2GB인 경우 그림에서 보는 것처럼 앱의 실행을 반복하더라도 스왑 지원 안드로이드의

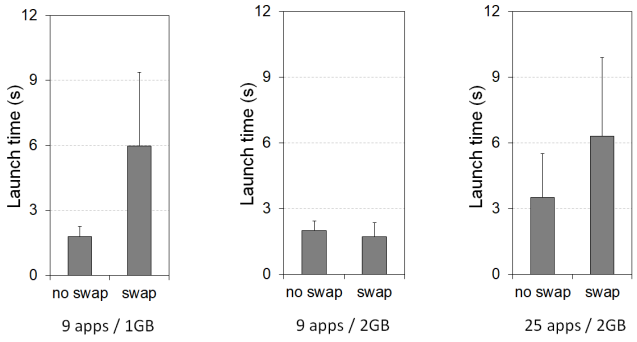


그림 3. 메모리 크기 및 앱의 수 변화에 따른 런치 시간
Fig. 3. Launch time as the memory size and the number of apps are varied.

런치 시간이 증가하지 않는 것을 확인할 수 있다. 즉, 스왑을 사용하더라도 1-3초의 비교적 안정된 런치 시간이 소요되는 것을 알 수 있다. 이를 통해 메모리 크기 증가가 안드로이드 스왑의 쓰레싱 현상을 해소할 수 있는 것으로 기대할 수 있다. 그러나, 본 논문에서는 앱의 수를 증가시킴에 따라 2GB 메모리를 사용하는 경우에도 1GB 메모리 때와 유사한 현상이 발생하는 것을 확인했다.

그림 3은 실행 앱의 수를 변화시키면서 메모리 크기에 따른 앱들의 런치 시간을 측정한 평균 및 표준편차를 보여주고 있다. 그림에서 보는 것처럼 1GB 메모리 하에서는 9개의 앱만을 실행했음에도 스왑 지원에 따른 런치 시간 증가가 확연히 나타났으나, 2GB 메모리 하에서는 실행 앱의 수가 9개일 때 스왑지원 여부와 무관하게 앱들의 런치 시간 분포가 비슷한 것을 확인할 수 있다. 그러나, 실행 앱의 수가 25개로 늘어남에 따라 스왑 지원에 따른 성능 저하는 1GB 메모리일 때와 유사하게 나타나는 것을 확인할 수 있다. 요약하자면 메모리 크기에 비해 실행 앱의 수가 확연히 늘어날 경우 스왑에 의한 스토리지 I/O가 급격히 늘어나는 쓰레싱이 여전히 발생하며, 이러한 현상은 메모리 크기 증가로 어느 정도 완화될 수 있으나, 앱의 수가 임계치를 넘어서면 또다시 발생한다는 의미이다.

본 논문에서는 그림 4와 같이 소량의 NVM을 메모리와 스토리지 사이에 배치하여 이러한 문제를 해결하는 새로운 아키텍쳐를 제안한다. 아키텍쳐의 핵심은 메모리에서 방출된 핫 데이터를 고속의 NVM이 흡수하는 것으로 본 논문에서는 스왑에 의해 발생하는 핫 데이터를 보관하기 위해 필요한 NVM의 용량에 대해 분석하고자 한다. 메모리의 여유 공간이 일정 임계치 이하로 떨어질 경우 스왑 지원 안드로이드는 최근에 사용되지 않은 데이터를 메모리에서 방출한다. 이때 방출 데이터가 메모리에 올라온 이후 수정되었을 경우 이를 먼저 스토리지에 반영해야 한다. 본 논문의 아키텍처에서는 방출 데이터를 스토리지에 반영하는 대신 NVM에 유지하도록 한다. 따라서, 메모리에서 방출된 데이터가 다시 요청될 경우 NVM에 해당 데이터가 존재하는지를 먼저 체크하여 불필요한 스토리지 접근을 억제할 수 있다.

한편, 본 논문의 아키텍쳐에서는 NVM이 메모리 계층이 아닌 그 아랫단에 위치하기 때문에 데이터 접근의 특성을 정교하게 활용하는 알고리즘 설계가 가능하다. 메모리 시스템에서는 각 데이터의 참조 시점을 정확히 파악할 수 없으며 최근에 사용된 데이터인지 여부를 1비트 정보로만 확인할 수 있어 비교적 단순한 알고리즘의 설계만이 허용된다. 예를 들어 CLOCK 알고리즘은 데이터가 최근에 사용되었는지를 나타내는 참조 비트의 2진 정보를 바탕으로 알고리즘을 운영한다[9]. 이와 달리 본 논문의 NVM 계층에서는 데이터가 요청된 시간과 요청 빈도 등을 활용한 알고리즘의 설계가 가능하다. 본 논문에서는 핫 데이터만을 NVM이 흡수하도록 하기 위해 메모

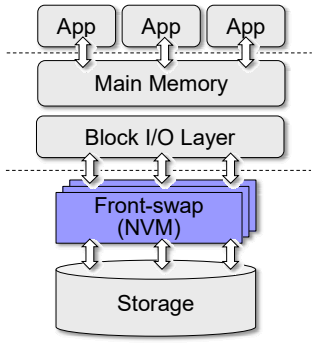
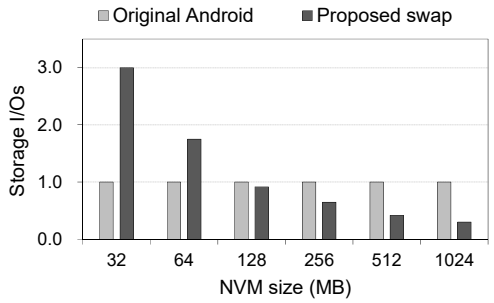
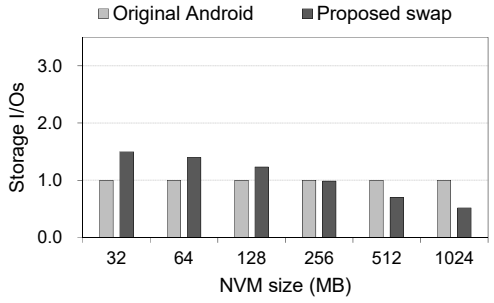


그림 4. NVM을 활용한 스마트폰 스왑 아키텍처
Fig. 4. Swap architecture of smartphones with NVM



(a) 1GB 메모리 시스템



(b) 2GB 메모리 시스템

그림 5. 메모리 및 NVM 크기 변화에 따른 스토리지 I/O
Fig. 5. Storage I/Os as memory and NVM sizes are varied.

리에서 최초로 쫓겨난 데이터의 경우 NVM에 저장하는 대신 메타데이터만을 기록해 두고, 두 번째 시점부터 NVM 진입을 허용하였다. 이러한 방식은 핫 데이터만을 NVM이 보관하도록 하여 소량의 NVM만으로도 핫 데이터를 흡수할 수 있도록 하기 위한 전략이다. NVM에 데이터 저장이 필요한 시점에 빈 공간이 없는 경우에는 가장 오래 전에 요청된 데이터를 방출하도록 설계하였다. 이는 메모리 시스템과 달리 데이터 요청 시점의 정확한 정보를 활용한 알고리즘 설계가 가능하기 때문이다.

그림 5는 본 논문의 NVM 아키텍처를 사용했을 때 스왑 지원 안드로이드의 스토리지 접근량을 기존의 스왑 미지원 안드로이드와 비교해서 보여주고 있다. 그림에서 보는 것처럼 스왑 지원 안드로이드는 NVM 크기가 128MB 미만일 경우 더 많은 I/O를 발생시키는 것을 확인할 수 있다. 그러나, NVM 크기가 256MB 이상인 경우 스왑 지원으로 인한 I/O 증가가 더 이상 발생하지 않았으며 이후 NVM 크기 증가 시 기존 시스템보다 더 적은 I/O량을 나타내었다. 이는 NVM의 크기가 스왑으로 인해 방출되는 핫 데이터를 담을 수 있을 만큼 충분히 크다는 것을 의미한다. 즉, 1-2GB 정도의 메모리 용량인 시스템에서 256MB 정도의 NVM이 지원될 경우 효율적인 관리 정책을 통해 스마트폰에 스왑을 지원하면서도 성능 저하를 방지할 수 있음을 뜻한다.

IV. 결 론

스마트폰이 기존 데스크탑에서 수행되던 다양한 작업을 흡수하는 멀티태스킹 디바이스로 발전함에 따라 가상 메모리 스왑 기능이 점점 중요해지고 있다. 그러나, 스마트폰에서의 스왑 기능 지원은 지나친 스토리지 트래픽을 유발하여 메모리 쓰레싱을 발생시키는 것으로 알려져 있다. 본 논문에서는 초창기 스마트폰의 스왑 지원시 발생하던 이러한 쓰레싱 현상이 메모리 크기가 증가함에 따라 어느 정도 해소될 수 있음을 보였다. 그러나, 스왑이 지원되지 않는 기존의 스마트폰과 달리 실행 앱의 수를 지속적으로 증가시킬 경우 여전히 성능 저하 현상이 발생하는 것을 확인하였다. 본 논문에서는 이러한 쓰레싱 유발이 일부 핫 데이터에 의해 발생된다는 것을 보이고 이를 NVM 기반의 스왑 가속 아키텍처를 통해 해결할 수 있는 방안을 제시하였다. 특히, 소량의 NVM만으로도 효율적인 관리를 통해 스왑 기능을 지원하면서 성능 저하 문제를 해소할 수 있음을 보였다.

References

[1] I. Nayeem and R. Want, “Smartphones: past, present, and future,” IEEE Pervasive Computing, vol. 13, no. 4, pp. 89-92, 2014
DOI: https://doi.org/10.1109/MPRV.2014.74.

[2] B. Lee and C. Son, “Improving evaluation metric of mobile application service with user review data,” Journal of the Korea Academia-Industrial cooperation Society, vol. 21, no. 1, pp. 380-386, 2020.
DOI: https://doi.org/10.5762/KAIS.2020.21.1.3

[3] B. Choi, S. Eom, C. Kim, and H. Lee, “Counterfeit bill identification based on deep learning using smartphone camera shooting images,” Journal of KIIT, vol. 19, no. 3, pp. 1-8, 2021.
DOI: https://doi.org/10.14801/jkiit.2021.19.3.1

[4] S. Kim, J. Jeong, J. Kim, and S. Maeng, “SmartLMK: a memory reclamation scheme for improving user-perceived app launch time,” ACM Trans. Embed. Comput. Syst., vol. 15, no. 47, pp. 1-25, 2016.
DOI: https://doi.org/10.1145/2894755

[5] S. Yoon, H. Park, K. Cho, and H. Bahn, “Supporting swap in real-time task scheduling for unified power-saving in CPU and memory,” IEEE Access, vol.

10, pp. 3559-3570, 2022.
DOI: https://doi.org/10.1109/ACCESS.2021.3140166

[6] 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.

[7] H. Bahn and K. Cho, "Implications of NVM based storage on memory subsystem management," Applied Sciences, vol. 10, no. 3, 2020.
DOI: https://doi.org/10.3390/app10030999

[8] Y. Park and H. Bahn, "Modeling and analysis of the page sizing problem for NVM storage in virtualized systems," IEEE Access, vol. 9, pp. 52839-52850, 2021.
DOI: https://doi.org/10.1109/ACCESS.2021.3069966

[9] S. Lee, H. Bahn, and S. H. Noh, ''CLOCK-DWF: A write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures,'' IEEE Trans. Computers, vol. 63, no. 9, pp. 2187-2200, 2014.
DOI: https://doi.org/10.1109/TC.2013.98

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

김 지 선(비회원)

- 2011년 2월: 한신대학교 컴퓨터공학과 학사
- 2014년 3월 ~ : 이화여자대학교 컴퓨터공학과 대학원생
- 주관심분야 : 운영체제, 스토리지 시스템

※ This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1009275) and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub).