The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 4, pp.77-82, Aug. 31, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.4.77
JIIBC 2024-4-12

모바일 인공지능 워크로드의 파일 접근 특성 분석

Analysis for File Access Characteristics of Mobile Artificial Intelligence Workloads

이정하[*], 임수정[**], 반효경[***]

Jeongha Lee[*], Soojung Lim[**], Hyokyung Bahn[***]

요 약 최근 인공지능 기술의 발전으로 모바일 환경에서 AI 응용을 수행하는 사례가 늘고 있다. 하지만, 모바일 환경은 데스크탑이나 서버에 비해 자원이 제한적이므로 인공지능 워크로드를 모바일에서 효율적으로 수행하기 위한 연구가 최근 주목받고 있다. 대부분의 연구는 컴퓨팅 자원의 제약을 해소하기 위한 엣지 또는 클라우드로의 오프로딩에 초점이 맞추어져 있으며, 스토리지 접근과 관련한 파일 입출력 특성에 관한 연구는 아직까지 널리 이루어지지 않고 있다. 본 논문에서는 모바일 환경에서 딥러닝 애플리케이션의 실행 시 발생하는 파일 입출력 트레이스를 분석하고, 기존 모바일 워크로드와의 차이점에 대해 분석한다. 본 논문의 분석 결과가 딥러닝의 파일 접근 특성을 고려하여 미래의 스마트폰 시스템 소프트웨어를 효율적으로 설계하는 데에 활용되기를 기대한다.

Abstract Recent advancements in artificial intelligence (AI) technology have led to an increase in the implementation of AI applications in mobile environments. However, due to the limited resources in mobile devices compared to desktops and servers, there is growing interest in research aimed at efficiently executing AI workloads on mobile platforms. While most studies focus on offloading to edge or cloud solutions to mitigate computing resource constraints, research on the characteristics of file I/O related to storage access in mobile settings remains underexplored. This paper analyzes file I/O traces generated during the execution of deep learning applications in mobile environments and investigates how they differ from traditional mobile workloads. We anticipate that the findings of this study will be utilized to design future smartphone system software more efficiently, considering the file access characteristics of deep learning.

Key Words : deep learning, file access, mobile device, smartphone, artificial intelligence

[*]준회원, 이화여자대학교 컴퓨터공학과
[**]준회원, 이화여자대학교 컴퓨터공학과
[***]정회원, 이화여자대학교 컴퓨터공학과
접수일자 2024년 6월 30일, 수정완료 2024년 7월 25일
게재확정일자 2024년 8월 9일

Received: 30 June, 2024 / Revised: 25 July, 2024 /
Accepted: 9 August, 2024
[*]Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

Ⅰ. 서 론

최근 모바일 하드웨어 기술의 빠른 발전과 인공지능 모델의 경량화 등으로 스마트폰에서 인공지능을 활용한 앱이 수행되는 사례가 늘고 있다[1, 2, 3]. 특히, 이미지 탐지, 얼굴 인식, 자연어 처리 등을 수행하는 앱들이 증가하면서 다양한 딥러닝 모델들이 스마트폰에 적용되고 있다[4].

딥러닝 모델을 사용한 학습 및 추론은 기존 워크로드에 비해 많은 컴퓨팅 자원을 요구하기 때문에, 제한된 자원을 가진 모바일 디바이스에서는 딥러닝 연산을 클라우드 또는 엣지 서버로 오프로딩하는 전략이 많이 사용되고 있다[5, 6]. 그러나, 이러한 오프로딩은 네트워크 지연과 프라이버시 등의 문제를 유발할 수 있고, 최근 고사양 GPU, ASIC, FPGA와 같은 가속기 하드웨어 기술의 발전으로 학습은 클라우드에서 수행하더라도 추론은 점차 모바일 기기에서 직접 수행하는 온 디바이스 AI 연구가 주목받고 있다[7].

모바일 기기에서 딥러닝 모델의 추론을 위해서는 이미지, 텍스트, 오디오 등의 입력 데이터와 사전에 훈련된 모델 가중치를 메모리에 적재한 후 연산을 수행해야 한다. 한편, 모바일 기기의 메모리 용량이 제한적이므로 서버 환경과 달리 딥러닝 워크로드 수행시 잦은 스토리지 접근이 발생할 수 있다[8]. 그러나, 기존의 모바일 딥러닝 연구들은 GPU 등 컴퓨팅 리소스를 효율적으로 사용하는 데에 초점을 맞추고 있으며, 파일 입출력 등 스토리지 접근에 대한 특성 분석이나 성능 개선에 관한 연구는 거의 이루어지지 않고 있다.

한편, 스마트폰 시스템의 성능에 관한 기존 연구에서 스마트폰 성능의 병목점이 되는 자원은 CPU나 메모리, 네트워크 장치보다는 스토리지인 것이 밝혀진 바 있다[9]. 이는 특히 스마트폰에서 파일 연산을 위해 사용하는 경량 데이터베이스 라이브러리인 SQLite가 잦은 동기적 쓰기를 스토리지에 발생시키기 때문으로 알려져 있다[10]. 또한, 스마트폰 워크로드의 파일 접근은 기존 데스크탑 또는 서버 상의 워크로드와는 상당히 다른 특성을 나타내는 것으로 보고된 바 있다. 안드로이드 환경에서 앱의 시작부터 종료 시까지 지속적으로 접근되는 제한된 수의 핫 파일 블록이 존재함이 밝혀졌으며, 이러한 핫 블럭들이 전체 파일 참조의 80%를 대변하는 것으로 분석된 바 있다[11]. 또한, 쓰기 연산이 전체 파일 접근의 50-90%를 차지하며, 이들의 대부분이 동기식 쓰기로 즉각적인 스토리지 플러시를 유발하는 것으로 알려진 바 있다.

본 연구에서는 이러한 스마트폰에서의 고유한 파일 접근 특성이 딥러닝 워크로드의 수행으로 어떻게 달라지는지를 조사하고자 한다. 딥러닝의 데이터 크기가 점점 커짐에 따라 파일 데이터 접근 특성을 분석하는 것이 중요해지고 있다. 스마트폰의 메모리 크기가 계속해서 증가하고 있지만, DRAM의 집적도 제한과 많은 유휴 전력 소모 때문에 끊임없이 증가하는 딥러닝 워크로드를 수용하기는 쉽지 않다[12]. 이러한 이유로 파일 접근 패턴을 분석하고 그 결과를 효율적인 캐싱에 활용하는 것이 모바일 딥러닝 환경에서 매우 중요하다.

본 논문에서는 모바일 딥러닝 응용의 파일 접근 트레이스를 분석한다. 이를 위해 카메라, 마이크 또는 저장소의 데이터를 입력으로 받아 분할(segmentation), 분류(classification) 등을 수행하는 다양한 딥러닝 앱을 동작시키며 파일 접근 트레이스를 추출하고 그 특성을 분석하였다.

본 분석을 통해 모바일 딥러닝 워크로드의 파일 참조에서 고유한 특성을 발견했다. 이는 전통적인 스마트폰 워크로드와는 차이가 있으며, 구체적으로는 읽기/쓰기 작업과 그 접근 패턴, 접근 편향성, 재참조 예측 등의 관점에서 비교 및 분석을 수행하였다. 본 논문을 통해 분석된 점을 다음과 같이 요약할 수 있다. 첫째, 읽기와 쓰기 접근을 비교할 때, 쓰기는 전체 파일 접근의 60-95%를 차지하며, 이는 쓰기가 50-90%를 차지하는 전통적인 모바일 앱과 유사하다[11]. 그러나, 대부분의 데스크탑 워크로드가 읽기 위주의 파일 참조를 보인다는 점과는 차별화된다. 둘째, 모바일 딥러닝 워크로드에서 쓰기 접근은 데이터 유형과 모델에 관계없이 반복적인 긴 루프 패턴을 포함한다. 이러한 접근 특성은 파일 블록을 어떻게 캐시할지에 대한 유용한 정보를 제공함으로써 버퍼 캐시의 적중률을 높일 수 있다. 셋째, 쓰기 접근이 파일 접근의 대다수를 차지하지만, 그 편향성은 매우 낮다. 구체적으로, 상위 20% 파일 블록이 전체 파일 접근의 22-25%를 차지한다. 이는 상위 20% 블록이 전체 접근의 80%를 차지하는 전통적인 안드로이드 앱의 경우와 차별화된다[11]. 넷째, 재참조 가능성을 예측할 때 일반적으로 최근성이 참조 빈도보다 더 의미 있는 정보를 제공하지만, 모바일 딥러닝 워크로드에서는 참조 빈도가 더 중요하다는 정보를 제공한다는 것을 확인했다.

본 논문에서 수행된 분석의 결과는 모바일 딥러닝 워크로드의 파일 접근을 고려한 효율적인 버퍼 캐시의 관리 정책 설계에 기여할 수 있을 것으로 보인다. 이를 통해 인공지능 위주의 작업을 수행하는 미래의 스마트폰 시스템 설계에 활용될 수 있기를 기대한다.

II. 트레이스 수집 및 분석

본 장에서는 모바일 기기에서 딥러닝 워크로드가 실행될 때 파일 접근이 전통적인 워크로드와 어떻게 다른지를 분석한다. 이를 위해 strace를 사용하여 입출력 시스템 콜 발생시의 이벤트 트레이스를 추출하였다. 실험에 사용한 디바이스는 삼성 Galaxy S8+이며, 실험을 위한 워크로드로는 Google AI Edge의 MediaPipe를 활용해 제작된 온디바이스 AI 샘플 앱들을 사용하였다. 공개된 여러 앱들 중 본 실험에 사용된 앱은 Face Landmarker, Hand Landmarker, Pose Landmarker, Audio Classification의 총 4종이다. 표 1은 추출된 트레이스에서 파일 접근의 읽기, 쓰기 비율을 조사한 것이다. 모든 워크로드에서 쓰기 참조의 비율이 60-95%로 많은 비중을 차지했으며, 이는 기존 모바일 워크로드의 읽기, 쓰기 비율과 일관된 결과이다[11].

표 1. 딥러닝 워크로드의 파일 참조 횟수 및 비율
Table 1. File Access Statistics of Deep Learning Workloads

	읽기 참조수	쓰기 참조수	읽기 : 쓰기
Face Landmarker	4,413	12,515	26:74
Hand Landmarker	28,757	43,018	40:60
Pose Landmarker	4,546	35,929	11:89
Audio Classification	958	17,656	5:95

그림 1은 시간이 흐름에 따라 파일 블록의 읽기 쓰기 접근이 어떤 트렌드로 나타나는지를 워크로드별로 보여주고 있다. 그림에서 x축은 논리시간으로, 파일 접근이 한 번 일어날 때마다 1씩 증가하는 개념이며, y축은 논리적인 블록 번호로, 매번 새로운 블록이 참조될 때마다 새 번호를 할당한다. 그림에서 보는 것처럼 워크로드의 종류와 무관하게 파란색으로 표시된 읽기에 비해 빨간색으로 표시된 쓰기가 훨씬 빈번히 나타나는 것을 확인할 수 있다. 읽기의 경우 순차참조를 나타내는 경우가 대부분이었으며 앱의 런치 초기에 주로 나타나고 이후에는 다량의 쓰기 사이에 간헐적으로 나타나는 것을 확인할 수 있다. 쓰기의 경우 상당히 긴 루프 패턴이 여러 차례에 걸쳐 반복해서 나타나는 것을 확인할 수 있다. 이러한 패턴은 MediaPipe가 사전 훈련된 모델 파일을 asset으로부터 본인의 전용 캐시 디렉토리로 복사한 후 다양한 형태의 추론이 진행됨에 따라 반복적인 쓰기 작업을 수행하기 때문이다.

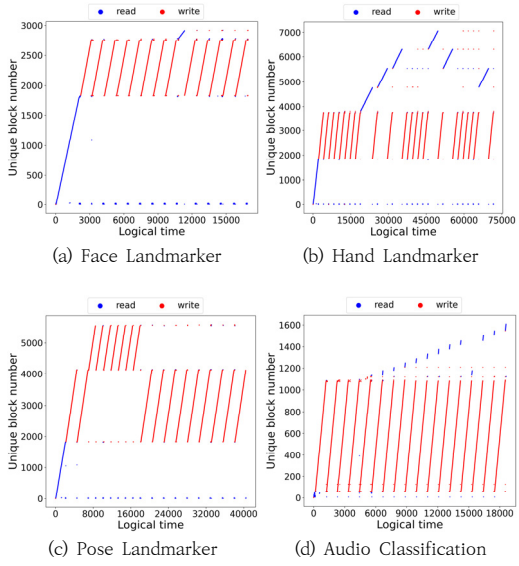


그림 1. 딥러닝 워크로드의 시간별 블록 참조
Fig. 1. Block references of deep learning workloads over time.

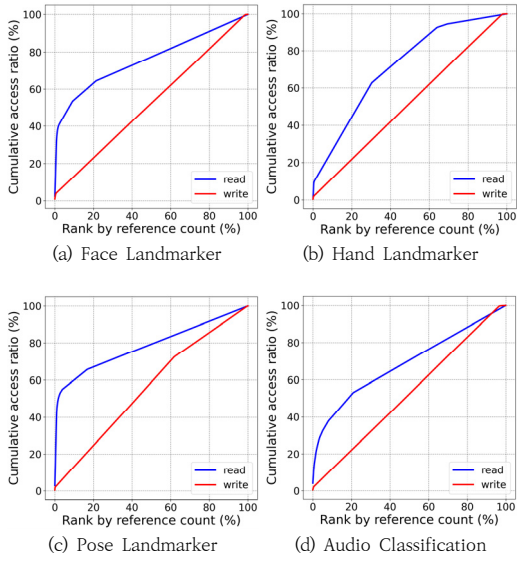


그림 2. 딥러닝 워크로드의 블록 참조 편향도
Fig. 2. Block reference bias of deep learning workloads.

III. 파일 블록의 편향도 분석

참조 편향도 분석은 인기 데이터를 캐싱하거나 특정 위치에 집중시켜 빠른 접근성을 제공하기 위해 중요한 역할을 한다[13]. 본 장에서는 파일 블록의 참조 편향도에

초점을 맞추어 모바일 딥러닝 앱의 파일 참조를 분석한다. 그림 2는 파일 블록들을 그 인기도 순서에 따라 정렬했을 때 상위 블록들의 누적 참조 비율을 보여주고 있다. 그림에서 x축은 각 블록을 참조횟수별로 순위를 매겨 이를 전체 파일블록 수 대비 비율로 나타낸 것이고, y축은 주어진 비율에 대한 상위 순위 블록의 접근 횟수를 파일 블록 전체의 접근 횟수 대비 비율로 나타낸 것이다. 그림 2에서 보이는 것처럼, 파일 참조의 편향도가 높지 않은 것으로 나타났으며, 특히 쓰기 접근의 경우 편향성이 더욱 낮은 모습을 보였다. 쓰기 접근의 경우 상위 20%의 파일 블록이 전체 참조의 평균 23%를 차지하며, 이는 상위 20%가 전체 접근의 80%를 차지하는 기존 스마트폰 앱의 특징과는 큰 차이가 있다[11].

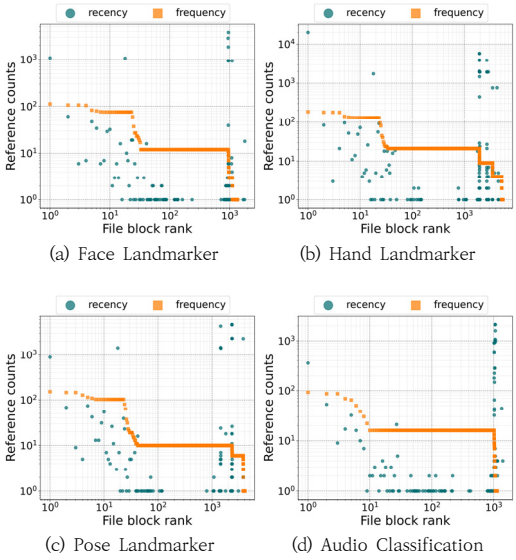
읽기 접근의 경우는 상위 20%의 블록이 전체 참조의 44-67%를 차지한다. 읽기의 경우 낮은 번호의 소수 파일 블록이 앱 실행 과정 중에 지속적으로 참조되기 때문에, 쓰기 접근에 비해 참조 편향도가 다소 높은 것으로 조사되었다.

IV. 파일 블록의 재참조 가능성 분석

전통적인 컴퓨팅 환경에서 스토리지 접근으로 인한 병목을 해결하기 위해 가장 널리 사용되어 온 방법은 버퍼 캐싱이다[14, 15]. 버퍼 캐시는 스토리지에서 로드된 파일 데이터를 메모리 중 일정 영역에 유지하여 느린 스토리지를 접근하지 않고도 다음 요청 시 캐시에서 즉시 데이터를 접근할 수 있도록 한다. 파일 캐시의 성능 개선을 위해서는, 파일 블럭의 재참조 가능성을 잘 예측해서 참조 가능성이 높은 블록을 최대한 캐시에 유지하는 것이 중요하다.

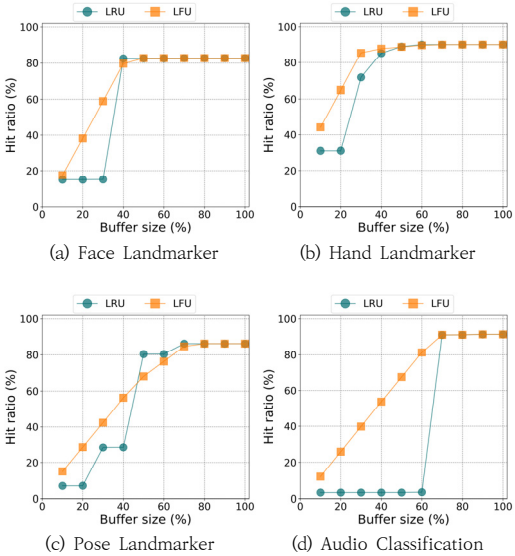
그림 3은 각 파일 블록의 재참조 가능성을 예측하기 위해, 참조의 최근성과 참조 빈도를 지표로 사용했을 때 얼마나 효과적인지를 비교한 것이다. 예를 들어 최근성 지표의 경우, 매 단위 시간마다 모든 블록들을 마지막 참조 순으로 정렬한 후 해당 단위 시간에 몇 번째 순위의 블록이 참조되었는지를 기록하여 각 순위별 참조 횟수를 기록한 것이다. 높은 순위의 블록일수록 참조 횟수가 많다면, 해당 지표가 재참조 예측을 더욱 효과적으로 한다고 할 수 있다. 그림 3에서 x축은 해당 지표에 따른 순위를 나타내고, y축은 해당 순위에서 나타난 참조 횟수를 나타낸다. 그림 3에서 보는 것처럼, 최근성 지표는 후순위의 블록이 선순위보다 더 높은 참조횟수를 기록하는 경우가 종종 관찰되지만, 참조 빈도 지표는 선순위에서 더 높은 참조를 나타내고 후순위로 갈수록 그 횟수가 낮아지는 것을 확인할 수 있다. 따라서 모바일 환경의 딥러닝 워크로드에서는 참조 빈도 지표가 참조 최근성 지표에 비해 미래의 참조를 더 잘 예측한다고 할 수 있다.

그림 4는 기존 LRU, LFU 알고리즘을 사용한 버퍼 캐시의 성능을 비교한 것이다. 각 앱을 실행하는 동안 접근된 파일 블럭들의 전체 크기를 기준으로 캐시의 용량을



(a) Face Landmarker (b) Hand Landmarker
(c) Pose Landmarker (d) Audio Classification

그림 3. 재참조 예측 지표 비교
Fig. 3. Comparison of re-reference estimator.



(a) Face Landmarker (b) Hand Landmarker
(c) Pose Landmarker (d) Audio Classification

그림 4. 버퍼 캐시의 성능 비교
Fig. 4. Performance comparison of buffer cache.

10%에서 100%까지 변화시키며 캐시의 적중률을 확인하였다. 참고로 100%에 가까운 캐시 크기를 사용할 경우 접근된 대부분의 파일 블록을 캐시에 담을 수 있기 때문에 어떤 알고리즘을 적용하더라도 적중률이 동일한 값으로 수렴하게 된다.

그림에서 볼 수 있는 것처럼, LRU 알고리즘의 경우 10-40% 크기의 캐시 구간에서 적중률의 변화가 거의 없다가, 특정 캐시 크기에서 적중률이 갑작스럽게 높아지는 모습을 보인다. 특히 앱의 실행 중 여러 개의 모델을 번갈아가며 사용하는 Pose Landmarker의 경우 적중률이 갑작스럽게 상승하는 복수의 캐시 구간이 나타난다. 이와 다르게, LFU 알고리즘을 사용한 경우 캐시 용량의 변화에 따라 적중률이 꾸준하게 상승하며, 대부분의 구간에서 LRU에 비해 높은 적중률을 나타내었다.

본 실험의 결과는, 오래 전 참조된 파일 블록을 캐시에서 쫓아내는 것이 효과적이라는 기존의 지식과 달리, 모바일 환경의 딥러닝 워크로드에서는 참조 횟수를 기준으로 캐시 방출 대상을 선정하는 것이 더 효과적이라는 점을 시사한다.

V. 결 론

본 논문에서는 모바일 환경에서 딥러닝 워크로드의 실행 시 나타나는 파일 접근의 특성을 파악하기 위해, 다양한 모바일 딥러닝 워크로드의 파일 입출력 트레이스를 추출하고 그 특징을 분석하였다. 본 논문의 분석 결과, 딥러닝 워크로드는 기존의 모바일 앱들과는 상이한 접근 특성을 나타내는 것을 확인하였다. 본 논문의 분석 결과가 딥러닝의 파일 접근 특성을 고려하여 미래의 스마트폰 시스템 소프트웨어, 특히 버퍼 캐시를 효율적으로 설계하는 데에 활용될 수 있을 것으로 기대된다.

References

[1] 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

[2] 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

[3] 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

[4] O. Incel and S. Bursa, “On-Device Deep Learning for Mobile and Wearable Sensing Applications: A Review,” IEEE Sensors Journal, Vol. 23, No. 6, pp. 5501-5512, 2023. DOI: https://doi.org/10.1109/JSEN.2023.3240854

[5] 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

[6] 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

[7] 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

[8] J. Lee and H. Bahn, “Analyzing Data Access Characteristics of Deep Learning Workloads and Implications,” Proc. IEEE EIECS, pp. 546-551, 2023. DOI: https://doi.org/10.1109/EIECS59936.2023.10435537

[9] H. Kim, N. Agrawal, and C. Ungureanu, “Revisiting storage for smartphones,” ACM Transactions on Storage, Vol. 8, Iss. 4, No. 14, pp. 1-25, 2012. DOI: https://doi.org/10.1145/2385603.2385607

[10] D. Q. Tuan, S. Cheon, and Y. Won, “On the IO characteristics of the SQLite transactions,” Proc. MOBILESoft, pp. 214-224, 2016. DOI: https://doi.org/10.1145/2897073.2897093

[11] S. Lim and H. Bahn, “Characterizing File Accesses in Android Applications and Caching Implications,” IEEE Access, vol. 9, pp. 150292-150303, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3125779

[12] S. Yoo, Y. Jo, and H. Bahn, “Integrated scheduling of real-time and interactive tasks for configurable industrial systems,” IEEE Transactions on Industrial Informatics, vol. 18, no. 1, pp. 631-641, 2022. DOI: https://doi.org/10.1109/TII.2021.3067714

[13] E. Lee, J. Whang, U. Oh, K. Koh, H. Bahn, “Popular channel concentration schemes for efficient channel navigation in internet protocol televisions,” IEEE Transactions on Consumer Electronics, vol. 55, no. 4, pp. 1945-1949, 2009. DOI: https://doi.org/10.1109/TCE.2009.5373754

[14] 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

[15] D. Teng, L. Guo, R. Lee, F. Chen, S. Ma, Y. Zhang, and X. Zhang "LSbM-tree: Re-Enabling Buffer Caching in Data Management for Mixed Reads and Writes," Proc. ICDCS, pp. 68-79, 2017.
DOI: https://doi.org/10.1109/ICDCS.2017.70

저 자 소 개

이 정 하(준회원)

- 2022년 2월 : 이화여자대학교 컴퓨터공학전공 학사
- 2022년 3월 ~ : 이화여자대학교 컴퓨터공학전공 대학원생

임 수 정(준회원)

- 2012년 2월 : 이화여자대학교 컴퓨터공학전공 학사
- 2020년 3월 ~ : 이화여자대학교 컴퓨터공학전공 대학원생

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