The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 3, pp.43-48, Jun. 30, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.3.43
JIIBC 2022-3-7

이종 모바일 멀티태스킹 환경을 위한 실시간 작업 인지형 메모리 할당 기술 연구

Real-time Task Aware Memory Allocation Techniques for Heterogeneous Mobile Multitasking Environments

반효경*

Hyokyung Bahn*

요 약 최근 스마트폰의 성능이 급격히 향상되고 모바일 플랫폼에서 백그라운드 앱의 실행이 늘면서 모바일 환경의 멀티태스킹이 활성화되고 있다. 모바일 환경에서는 종래의 데스크탑 및 서버 응용들과 달리 응답시간이 중요한 대화형 작업들이 대부분을 차지하고 있으며, 일부 응용은 데드라인이 존재하는 실시간 작업에 해당된다. 본 논문에서는 스마트폰에서 실시간 작업과 대화형 작업이 동시에 실행될 때 메모리 관리를 어떻게 함으로써 이질적인 멀티태스킹 환경의 요구사항을 충족할 수 있는지에 대해 연구한다. 본 논문에서는 실시간 작업의 요구 조건 만족을 위해 필요한 메모리 크기를 분석 및 모델링하고 이에 기반해서 멀티태스킹 작업 간의 메모리를 할당하는 방안을 제안한다. 이종 앱의 스토리지 접근 트레이스를 추출하고 이에 기반한 시뮬레이션을 통해 제안한 기법이 실시간 작업의 요구를 일정 수준으로 보장하면서 대화형 작업에 합리적인 성능을 제공함을 확인하였다.

Abstract Recently, due to the rapid performance improvement of smartphones and the increase in background executions of mobile apps, multitasking has become common on mobile platforms. Unlike traditional desktop and server apps, response time is important in most mobile apps as they are interactive tasks, and some apps are classified as real-time tasks with deadlines. In this paper, we discuss how to meet the requirements of heterogeneous multitasking in managing memory of real-time and interactive tasks when they are executed together on a smartphone. To do so, we analyze the memory requirement of real-time tasks, and propose a model that has the ability of allocating memory to multitasking tasks on a smartphone. Trace-driven simulations with real-world storage access traces captured by heterogeneous apps show that the proposed model provides reasonable performance for interactive tasks while guaranteeing the requirement of real-time tasks.

Key Words : Real-time task, memory allocation, multitasking, mobile platform, smartphone

*정회원, 이화여자대학교 컴퓨터공학과
접수일자 2022년 5월 15일, 수정완료 2022년 6월 3일
게재확정일자 2022년 6월 10일

Received: 15 May, 2022 / Revised: 3 June, 2022 /
Accepted: 10 June, 2022
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

Ⅰ. 서 론

안드로이드 등의 모바일 플랫폼은 그 하위에 리눅스 커널을 탑재하고 있어 기본적으로 멀티태스킹 기능을 지원한다[1]. 그러나, 초창기 안드로이드는 음악 재생이나 파일 다운로드 등 특정 응용을 제외하고는 화면을 장악하고 있는 하나의 앱만을 활성화하는 방식을 사용했었다. 이는 스마트폰 기기의 화면이 작아 서로 다른 앱이 떠있는 창을 동시에 보여주기 어렵고 스마트폰의 메모리 용량 한계 및 스왑의 미사용 등으로 현실적으로 멀티태스킹이 실현되기에 한계가 있었기 때문이다. 그러나, 최근 스마트폰의 성능이 급격히 향상되면서 모바일 환경에서의 멀티태스킹이 활성화되고 있는 추세이다. 최신의 안드로이드 레퍼런스 폰인 픽셀 6 pro의 경우 12GB의 메모리 용량을 탑재하고 있으며, CPU 코어도 8개가 장착되어 있다[2]. 또한, 안드로이드의 메모리 스왑 기능이 지나친 성능 저하를 일으켜 LMK (low memory killer)로 앱을 강제 종료시키던 방식 역시 최근 고속 스토리지의 출현으로 해소될 수 있을 것으로 보인다[3]. 스마트폰의 화면 또한 폴더블 폰의 출현으로 충분히 커지면서 한 화면에 여러 앱을 보여줄 수 있는 방식이 도입될 수 있을 것이 기대된다.

따라서, 앞으로 스마트폰은 점점 더 멀티태스킹을 지향하는 쪽으로 발전할 것으로 보인다. 한편, 모바일 환경에서는 종래의 데스크탑 및 서버 응용들과 달리 응답시간이 중요한 대화형 작업들이 대부분을 차지하고 있으며, 일부 응용은 데드라인이 존재하는 실시간 작업에 해당된다[4, 5, 6]. 범용 컴퓨터로 사용되는 데스크탑이나 서버에서 실행되는 응용은 사용자의 입력에 곧바로 응답해야 하는 대화형 작업 외에 오랜 연산 후 결과를 한꺼번에 출력하는 배치형 작업이 공존하므로 우선순위를 적절히 부여하여 두 작업이 효율적으로 수행될 수 있는 방법이 존재했다.

그러나, 스마트폰 환경에서는 대부분의 작업에 있어 시간 요구조건이 매우 중요하여 멀티태스킹 시 이들을 잘 관리할 수 있는 메커니즘이 필요하다. 대화형 작업과 실시간 작업이 동시에 실행될 때 CPU 코어의 스케줄링과 메모리의 분배는 가장 중요한 두 가지 문제라 할 수 있다. CPU의 경우 코어의 수가 늘어나면서 실시간 작업용 전담 코어를 두는 방식으로 어느 정도의 요구사항은 해소할 수 있으나[7] 메모리의 경우 사용자가 스마트폰의 전원을 켜놓은 후 오랜 시간이 흐를 경우 멀티태스킹 중인 앱의 수가 지속적으로 증가하여 여유 공간이 소진될 수밖에 없다[8].

본 논문에서는 스마트폰에서 실시간 작업과 대화형 작업이 동시에 실행될 때 메모리 관리를 어떻게 함으로써 이질적인 멀티태스킹 환경의 요구사항을 충족할 수 있는지에 대해 연구한다. 현재 안드로이드 등 대부분의 스마트폰 환경에서는 실시간 작업에 대한 특별한 고려 없이 메모리 공간을 멀티태스킹 중인 작업들끼리 경쟁하는 방식을 사용하고 있다. 이러한 방식은 멀티태스킹이 활성화되면서 실시간 작업의 스토리지 접근을 증가시켜 데드라인 미스를 발생시키는 문제를 초래하게 된다[9].

이를 해결하기 위해 본 논문에서는 실시간 작업의 요구 조건 만족을 위해 필요한 메모리 크기를 분석 및 모델링하고 이에 기반해서 멀티태스킹 작업 간의 메모리를 할당할 수 있는 방안을 제안한다. 이종 앱의 스토리지 접근 트레이스를 추출하고 이에 기반한 시뮬레이션을 통해 제안한 모델의 효과에 대해 검증한다. 실험결과 제안한 모델은 실시간 작업의 요구를 일정 수준으로 보장하면서 대화형 작업에 합리적인 성능을 제공함을 확인하였다.

Ⅱ. 메모리 요구량 모델링

실시간 작업의 데드라인을 만족시키기 위해서는 데이터가 요청되기 전에 메모리에 올라와 있도록 하여 스토리지 접근이 온디맨드 방식으로 이루어질 경우 발생하는 지연을 방지하는 것이 중요하다. 이는 실시간 작업의 경우 현재 수행중인 작업집합을 메모리에 보장할 수 있는 정도의 메모리 할당이 필요함을 뜻한다[10]. 예를 들어 동영상 플레이와 같은 실시간 작업이 실행될 때 메모리가 부족하여 요청 데이터를 즉석에서 스토리지로부터 읽어야 하는 상황이 지속적으로 발생한다면 끊김 현상이 나타나게 될 것이다. 반면, 필요한 데이터를 미연에 메모리에 읽어들일 수 있다면 이런 문제는 해결이 가능하다.

실시간 작업만을 전담하는 임베디드 시스템에서는 작업에 필요한 모든 데이터를 상주시킬 충분한 용량의 메모리를 사전에 장착하여 스토리지 접근이 작업 중에 발생하지 않는 방식을 택한다[11]. 그러나 이러한 방식은 미리 정해지지 않은 다양한 작업들이 그때그때 실행되는 스마트폰 환경에서는 적용되기 어렵다.

메모리 용량이 늘어남에 따라 워크로드의 스토리지 접근은 줄어들게 되며, 일정 수준 이상의 메모리 용량에 이르면 그 개선 폭이 감소하여 어느 순간부터는 메모리 용량을 늘여도 더 이상 스토리지 접근을 줄일 수 없는 지점

에 이르게 된다. 이는 2개의 제어 파라미터를 가지는 지수함수로 모델링이 가능한 것으로 알려져 있다[12]. 이 때, 모델링 함수의 파라미터는 워크로드를 구성하는 데이터의 인기 편향성에 따라 결정해야 할 요소이다. 본 논문에서는 실행 중인 실시간 작업의 특성을 반영하는 파라미터를 결정하여 메모리 할당량 변화에 따른 스토리지 접근 비율을 잘 예측할 수 있는 모델을 설계하였다.

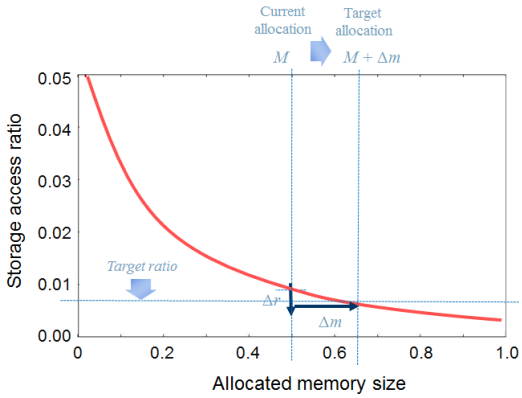
실시간 작업의 데드라인을 100% 보장하기 위해서는 스토리지 접근을 완전하게 제거해야 하며, 그럴 경우 실시간 작업의 메모리 풋프린트를 한꺼번에 적재할 수 있는 메모리 할당이 필요하다. 이러한 방식은 멀티태스킹 환경에서 지나친 메모리 요구량을 발생시키기 때문에 본 논문에서는 실시간 작업의 스토리지 접근에 대한 타겟 비율을 정하고 이 비율을 만족할 수 있는 수준의 메모리를 할당하도록 하였다. 스마트폰에서 실행되는 실시간 작업처럼 연성 실시간 작업(soft real-time task)의 경우 이러한 방식을 사용해서 작은 비율의 데드라인 미스를 허용하는 것이 실제 시스템에서는 좀 더 현실적인 접근에 해당한다.

그림 1은 이러한 모델에 근거한 실시간 작업의 메모리 할당 방식을 보여주고 있다. 빨간 곡선은 할당 메모리에 따른 스토리지 접근 비율을 나타내고 있으며, 그림 1의 (a)와 (b)는 각각 실시간 작업의 데이터 편향도에 따라 곡선의 경사가 상이하게 나타나는 것을 보여주고 있다. 그림에서 현재 실시간 작업에 할당된 메모리 용량이 x축의 0.5에 해당하는 지점이라 할 때, 이 지점의 스토리지 접근 비율이 타겟 비율보다 높은 경우(a에 해당)와 낮은 경우(b에 해당) 메모리 할당량을 M에서 Δm만큼 증가(a) 또는 감소(b)시켜 실시간 작업의 요구를 만족시키는 모습을 보여주고 있다.

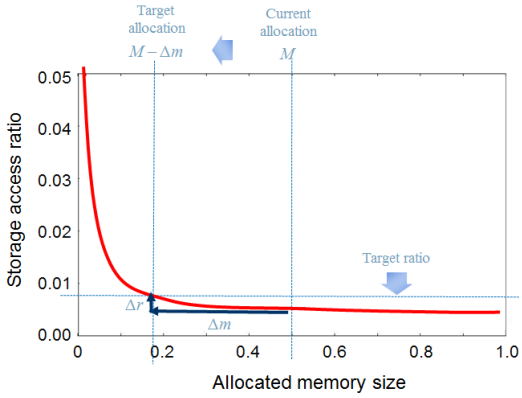
한편, 이러한 방식은 실시간 작업이 시작되어 종료되는 전체 기간 동안 사용하는 모든 풋프린트를 메모리에 보관하는 전통적인 실시간 시스템과 비교할 때 워크로드의 동적 변화를 파악하고 이에 필요한 최소한의 메모리만을 주기적인 모니터링을 통해 할당한다는 측면에서 멀티태스킹의 효율을 높일 수 있는 장점이 있다.

Ⅲ. 성능 평가

본 논문에서는 안드로이드 앱이 실행되면서 발생하는 파일 입출력 트레이스를 추출하고 이를 재현하는 실험을 통해 제안한 모델의 성능을 분석한다. 트레이스 추출에



(a) 스토리지 접근 비율이 타겟비율보다 높은 경우



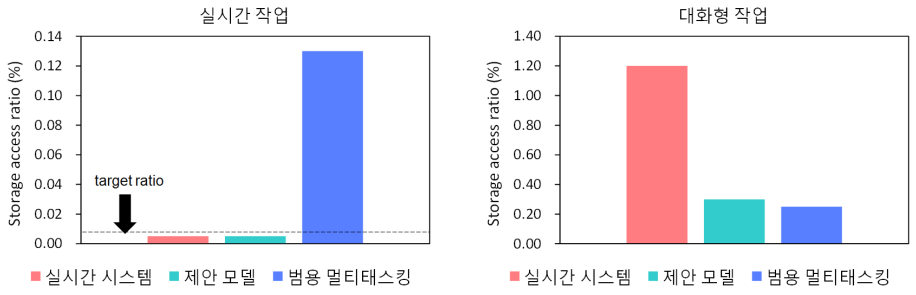
(b) 스토리지 접근 비율이 타겟비율보다 낮은 경우

그림 1. 제안 모델에 근거한 실시간 작업의 메모리 할당
Fig. 1. Memory allocation of real-time tasks based on the proposed model.

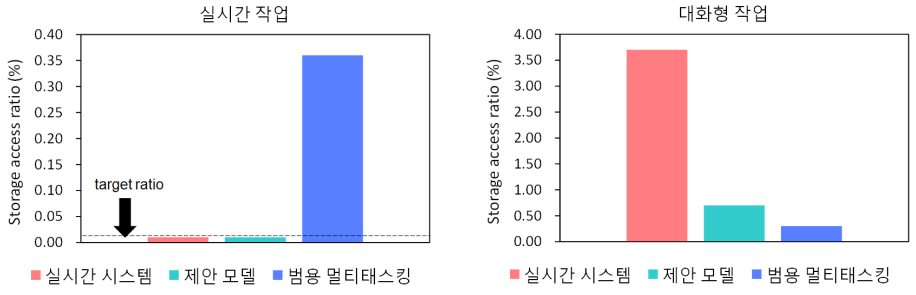
사용된 앱은 동영상 재생, 실시간 메신저, 비디오 게임, 지도 앱, 이메일 클라이언트, 사회관계망 등 총 6종이다. 이들은 모두 사용자 응답시간이 중요한 앱이지만 이들을 실시간 작업과 대화형 작업으로 나눌 경우, 동영상 재생, 비디오 게임, 실시간 메신저 등은 실시간 작업으로 분류할 수 있고, 지도 앱, 이메일 클라이언트, 사회관계망 등은 실시간 작업처럼 데드라인이 존재하지 않아 대화형 작업으로 분류할 수 있다.

현재 스마트폰 시스템의 동작 방식이 화면을 장악하고 있는 하나의 앱만을 활성화시키는 방식으로 운영되고 있는 점을 감안해서 본 실험에서는 실시간 앱과 대화형 앱이 각각 하나씩 메모리 경합을 하고 있는 상황에서 이들 간의 메모리 할당 방식에 따른 성능을 시뮬레이션을 통해 보여주고자 한다.

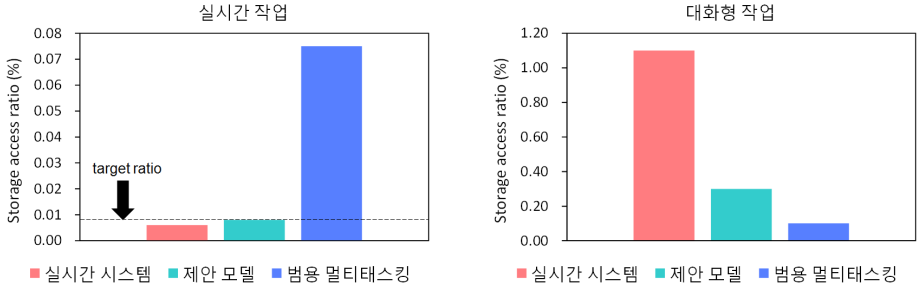
본 논문에서 제안한 메모리 할당 모델과의 비교 대상



(a) 비디오 게임과 이메일 클라이언트의 멀티태스킹 상황



(b) 동영상 재생과 사회관계망 앱의 멀티태스킹 상황



(c) 실시간 메신저와 지도 앱의 멀티태스킹 상황

그림 2. 범용 멀티태스킹 및 실시간 시스템과의 성능 비교
Fig. 2. Performance comparison with general purpose multi-tasking and real-time systems.

으로는 전통적인 실시간 시스템 방식, 즉 실시간 작업의 풋프린트 전체를 메모리에 상주시키는 방식과 범용 멀티태스킹 방식, 즉 리눅스 및 안드로이드에서처럼 작업의 특성을 고려하지 않고 메모리를 경쟁시키는 방식을 사용했다.

그림 2는 메모리 할당 방식에 따라 동시에 실행시킨 실시간 작업과 대화형 작업의 스토리지 접근 비율을 보여주고 있다. 구체적으로 살펴보면 그림 2(a)는 비디오 게임과 이메일 클라이언트의 멀티태스킹 상황을 나타내고 있으며, 그림 2(b)는 동영상 재생과 사회관계망 앱, 그림 2(c)는 실시간 메신저와 지도 앱의 멀티태스킹 상황에서 각 기법의 결과를 실시간 작업과 대화형 작업으로 나누어 보여주고 있다.

그림에서 보는 것처럼 제안하는 모델은 모든 경우에 있어 실시간 작업의 스토리지 접근 비율을 타겟 비율 이하로 보장하고 있음을 확인할 수 있다. 물론 실시간 시스템 방식의 경우에 실시간 작업의 스토리지 접근 비율이 가장 낮은 것을 확인할 수 있지만, 이러한 방식은 실시간 작업의 풋프린트를 메모리에 모두 올려놓기 위해 대화형 작업의 메모리 할당량을 지나치게 축소시키는 문제점을 야기하게 된다. 그 결과 대화형 작업에 있어 성능저하가 뚜렷이 나타나는 것을 확인할 수 있다. 구체적

으로는 제안 모델 대비 2.6배에서 4배까지의 스토로지 접근 증가를 대화형 작업에 대해 발생시키는 것을 확인할 수 있었다.

한편, 범용 멀티태스킹 방식의 경우 실시간 작업의 요구를 제대로 반영하지 못하고 스토리지 접근의 타겟 비율을 보장하지 못하는 것을 확인할 수 있다. 이는 범용 멀티태스킹 방식이 모든 앱을 동일한 기준으로 경쟁시키기 때문에 나타난 결과로 볼 수 있다. 그림에서 보는 것처럼 범용 멀티태스킹 방식은 실시간 작업과 대화형 작업의 스토리지 접근 비율의 편차가 비교적 크지 않음을 확인할 수 있으며, 그 결과 다른 두 방식에 비해 대화형 작업의 스토리지 접근 비율은 확연히 줄어드는 효과를 확인할 수 있다. 한편, 제안한 모델의 경우 비록 범용 멀티태스킹 방식보다는 대화형 작업의 성능이 나쁘지만 실시간 시스템 방식에 비해서는 크게 개선된 결과를 얻었으며, 이는 실시간 작업의 요구사항을 충족하면서 얻을 수 있는 합리적인 결과라고 볼 수 있다.

Ⅳ. 결 론

본 논문에서는 스마트폰에서 실시간 작업과 대화형 작업이 동시에 실행될 때 메모리 관리를 어떻게 함으로써 이질적인 멀티태스킹 환경의 요구사항을 충족할 수 있는지에 대해 연구하였다. 안드로이드 등 대부분의 스마트폰 환경에서는 실시간 작업에 대한 특별한 고려 없이 메모리 공간을 멀티태스킹 중인 작업들끼리 경쟁하는 방식을 사용하고 있어 실시간 작업의 요구를 충족시키지 못하는 문제점이 있다. 이에 반해 전통적인 실시간 시스템에서는 작업에 필요한 모든 데이터를 상주시킬 충분한 용량의 메모리를 사전에 장착하여 스토리지 접근이 작업 중에 발생하지 않는 방식을 택하나, 이러한 방식은 다양한 작업들이 시간에 따라 변화하면서 실행되는 스마트폰 환경에서는 활용되기 어렵다. 이를 해결하기 위해 본 논문에서는 실시간 작업의 요구 조건 만족을 위해 필요한 메모리 크기를 모델링하고 이에 기반해서 멀티태스킹 작업 간의 메모리를 동적으로 할당하는 모델을 제안하였다. 이종 앱의 스토리지 접근 트레이스를 추출하고 이에 기반한 시뮬레이션을 통해 제안한 모델이 실시간 작업의 요구를 일정 수준으로 보장하면서 대화형 작업에 합리적인 성능을 제공함을 확인하였다.

References

[1] F. Khomh, H. Yuan, and Y. Zou, “Adapting Linux for mobile platforms: An empirical study of Android,” 28th IEEE Int’l Conf. Software Maintenance (ICSM), pp. 629-632, 2012.
DOI: https://doi.org/10.1109/ICSM.2012.6405339

[2] Google Pixel 6 Pro, https://store.google.com/gb/product/pixel_6_pro

[3] 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

[4] J. Kim and H. Bahn, “Maintaining Application Context of Smartphones by Selectively Supporting Swap and Kill,” IEEE Access, vol. 8, pp. 85140-85153, 2020.
DOI: https://doi.org/10.1109/ACCESS.2020.2992072

[5] 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

[6] 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

[7] S. Yoo, Y. Jo, and H. Bahn, “Integrated Scheduling of Real-Time and Interactive Tasks for Configurable Industrial Systems,” IEEE Trans. on Industrial Informatics, vol. 18, no. 1, pp. 631-641, 2022.
DOI: https://doi.org/10.1109/TII.2021.3067714

[8] Y. Park and H. Bahn, “Challenges in memory subsystem design for future smartphone systems,” IEEE Inte’l Conf. Big Data and Smart Computing (BigComp), pp. 255-260, 2017.
DOI: https://doi.org/10.1109/BIGCOMP.2017.7881707

[9] 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

[10] S. Nam, K. Cho, and H. Bahn, “Tight Evaluation of Real-Time Task Schedulability for Processor’s DVS and Nonvolatile Memory Allocation,” Micromachines, vol. 10, no. 6, 2017.
DOI: https://doi.org/10.3390/mi10060371

[11] X. Cheng, Y. Guan, and Y. Zhang, “Design and Implementation of Dynamic Memory Allocation Algorithm in Embedded Real-Time System,” Int’l Conf. Pioneering Computer Scientists, Engineers and Educators, pp. 539-547, 2018.
DOI: https://doi.org/10.1007/978-981-13-2203-7_43

[12] S. Lee and H. Bahn, "Characterization of Android Memory References and Implication to Hybrid Memory Management," IEEE Access, vol. 9, pp. 60997-61009, 2021.
DOI: https://doi.org/10.1109/ACCESS.2021.3074179

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

※ This work was supported by the IITP grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub) and the ICT R&D program of MSIT/IITP (2018-0-00549, Extremely Scalable Order Preserving OS for Manycore and Non-volatile Memory).