The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 5, pp.43-48, Oct. 31, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.5.43
JIIBC 2022-5-7

# 모바일 기기의 실시간 작업 지원을 위한 종단간 자원 관리 기술

# End-to-End Resource Management Techniques for Supporting Real-time Tasks in Mobile Devices

**반효경***

Hyokyung Bahn*

**요 약** 최근 모바일 기기의 성능이 급격히 향상되고 다양한 앱이 등장하면서 대화형 작업뿐 아니라 실시간 작업을 동반하는 앱이 늘고 있다. 한편, 실시간 작업은 데드라인 제약 조건이 있어 종래의 시분할 시스템이 추구하던 자원 관리 정책으로는 실시간 제약 조건을 만족하는 데에 한계가 있다. 본 논문에서는 모바일 기기에서 대화형 작업과 실시간 작업이 동시에 실행될 때 CPU, 메모리, 스토리지로 이어지는 종단간 자원 관리를 어떻게 함으로써 실시간 작업의 제약 조건을 만족하면서 자원의 효율적인 관리가 가능한지에 대해 알아본다. 본 논문에서는 복잡한 자원관리 정책의 제안보다 각 자원들이 실시간 작업의 조건을 만족하기 위해 필요한 기본 개념에 대해 살펴보는 데에 초점을 맞춘다. CPU의 경우 실시간 작업을 위한 전담코어 할당, 메모리의 경우 워킹셋의 일정 비율을 보장하는 방식, 스토리지의 경우 고속 스토리지를 사용하고 문맥교환을 생략하는 방식 등 기본적인 지원 방안을 알아보고 이를 어떻게 효율화할 수 있는지에 대해 알아본다.

**Abstract** With the recent performance improvement of mobile devices as well as the emergence of various applications, not only interactive tasks but also real-time tasks are rapidly increasing. As real-time tasks have deadline requirements, resource management policies used in the conventional time-sharing systems have limitations in satisfying real-time constraints. In this paper, we examine how to efficiently manage resources while satisfying the constraints of real-time tasks through end-to-end resource management of CPU, memory, and storage when interactive and real-time tasks are executed concurrently on a mobile device. Instead of suggesting complicated resource management policies, we focus on examining the basic concepts necessary for each resource management. Specifically, we first look at basic policies such as assigning dedicated CPU cores for real-time tasks, allocating a certain working set of real-time tasks in memory, and using fast storage without context switch in I/O. We then consider how these basic policies can be adopted efficiently.

**Key Words :** Real-time task, resource management, mobile device, mobile platform, smartphone

*정회원, 이화여자대학교 컴퓨터공학과
접수일자 2022년 9월 13일, 수정완료 2022년 9월 30일
게재확정일자 2022년 10월 7일

Received: 13 September, 2022 / Revised: 30 September, 2022 / Accepted: 7 October, 2022
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

## Ⅰ. 서 론

모바일 기기의 소프트웨어 플랫폼은 대부분 그 하위에 멀티태스킹 기능을 지원하는 시분할 운영체제가 탑재되어 여러 앱의 동시 실행을 지원할 수 있다[1]. 한편, 최근 모바일 기기의 성능이 급격히 향상되고 다양한 앱이 등장하면서 대화형 작업뿐 아니라 실시간 작업을 모바일 기기에서 실행하는 사례가 늘고 있다[2, 3]. 2022년 출시된 안드로이드 레퍼런스 폰 Pixel 6a의 경우 구글 텐서 코어를 장착하고 8개의 CPU 코어와 6GB의 메모리, 128GB의 UFS 3.1 스토리지로 구성되며, 이러한 하드웨어 사양은 대화형 작업의 멀티태스킹을 지원하기에는 충분하다[4]. 그러나, 하드웨어 사양과 별개로 기존의 시분할 운영체제가 추구하던 자원 관리 정책으로는 실시간 작업의 요구 조건을 충족하기 어렵다. 특히, 동영상 재생과 같은 실시간성 작업이 다른 앱과 함께 실행될 때, CPU나 메모리의 할당이 늦어져 끊김 현상이 발생할 수 있으며, 느린 스토리지 접근으로 응답성이 떨어져 사용자의 불편함을 가중시킬 수 있다. 이는 현재 모바일 기기의 소프트웨어 플랫폼에 탑재된 시분할 운영체제가 실시간성 작업의 우선순위만을 높여줄 뿐 데드라인을 보장해 주지는 못하기 때문이다.

본 논문은 모바일 기기에서 실시간 작업이 일반적인 대화형 작업과 함께 실행될 때, 다른 작업들의 간섭으로 인한 지연을 최소화하고 데드라인을 최대한 충족시킬 수 있는 자원 관리 방안에 대해 살펴본다. 특히, 실시간 작업의 데드라인 만족을 위해서는 CPU, 메모리, 스토리지로 이어지는 종단간(end-to-end) 자원 관리에 있어 멀티태스킹의 간섭을 배제하고 시간 제약을 충족시킬 수 있는 방안이 필요하다. 본 논문에서는 복잡한 자원관리 정책을 제안하기보다 각 자원들에 있어 실시간 작업의 조건을 만족하기 위한 기본 정책을 살펴보는 데에 초점을 맞추고 어떻게 하면 멀티태스킹 환경에서 이들 자원을 좀더 효율적으로 관리하는 것이 가능한지에 대해 살펴본다.

CPU의 경우 시분할 운영체제가 채택하는 할당 시간(time quantum) 기반의 선점형(preemptive) 방식에서는 동시 작업의 수가 늘어남에 따라 실시간 작업의 시간 제약 조건을 만족하지 못할 가능성이 커진다[5]. 리눅스의 경우 실시간성 작업의 우선순위를 높여주는 방식을 채택하고 있지만 시분할 운영체제의 근본적인 특성상 데드라인을 만족시키는 방식으로 설계돼 있지 않다. 한편, 모바일 기기의 특성상 동시 실행 중인 다수의 작업 중 대화형 작업은 여럿이 존재하더라도 실시간 작업은 최대 1개가 동시 실행 중인 것으로 가정하는 것이 현실적이다. 따라서, 다수의 코어가 장착된 모바일 기기에서 실시간 작업의 시간 요구조건을 충족하기 위해서는 실시간 작업을 전담하는 코어를 두어 해당 코어는 실시간 작업 실행 중 비실시간 작업들이 스케줄링되지 못하도록 하는 방식을 통해 실시간 작업의 데드라인을 만족하도록 설계하는 것이 가능하다.

메모리 관리의 경우 한정된 크기의 메모리 공간에 여러 작업이 동시에 실행되면 앱 당 할당 메모리 공간이 줄어들 수밖에 없다[6]. 이는 결국 잦은 스토리지 접근을 유발하여 실행시간의 급격한 지연을 초래하게 된다[7]. 실시간 작업의 데드라인을 만족시키기 위해서는 느린 스토리지 접근이 없어야 하므로 전통적인 실시간 시스템의 경우 가상메모리 기능을 사용하지 않고 실시간 작업 전체를 물리적 메모리에 올려놓는 방식을 채택해 왔다[8]. 그러나, 이러한 방식은 비실시간 작업들의 메모리 할당량을 크게 줄어들게 하여 지나친 성능 차별을 초래하게 된다. 이를 해결하기 위해서는 실시간 작업 전체를 메모리에 올려놓기보다 성능저하가 발생하지 않을 정도의 메모리 공간을 할당하는 정책이 필요하다. 이러한 방식을 사용할 경우 작업 전체가 메모리에 적재돼 있지 않으므로 비록 실시간 작업이라 하더라도 스토리지 접근이 불가피하다. 이 때, 기존의 관리 정책은 비록 실시간성 작업이라 하더라도 스토리지 I/O를 수행 중인 경우 CPU를 선점(preemption)당하는 방식으로 설계되어 데드라인을 보장할 수 없다. 이를 해결하기 위한 방법으로 실시간성 작업을 기존의 스토리지가 아닌 고속 스토리지에 보관하여 빠른 I/O를 보장하고 I/O 작업이 진행되는 동안 CPU를 선점당하던 기존 방식과 달리 고속 스토리지를 통한 I/O를 수행중인 실시간 작업에 대해서는 문맥교환을 생략하고 I/O의 완료를 대기하는 방식을 사용할 수 있다.

본 논문의 이후 구성은 다음과 같다. II장에서는 모바일 기기의 실시간 작업 지원을 위한 방안을 구체적으로 살펴본다. III장에서는 실험을 통해 이러한 방법들의 유효성을 검증한다. 끝으로 IV장에서는 본 논문의 결론을 제시한다.

## Ⅱ. 실시간 작업의 지원 방안

본 장에서는 모바일 기기에서 실시간성 작업의 데드라

인 조건을 만족하기 위해 시분할 운영체제가 어떻게 개선되어야 하는지에 대해 구체적으로 살펴본다. 그 첫째는 실시간 작업을 위한 스토리지로 기존의 플래시메모리 대신 고속 스토리지를 사용하는 것이다. 실시간 작업은 데드라인 내에 주어진 작업을 수행하는 것이 중요하기 때문에 실행속도에 가변성이 적어야 한다. 그러나 스토리지 연산은 여러 작업 간의 간섭 및 접근 속도의 가변성 등으로 접근 시간이 일정하지 않아 실시간 작업의 실행 시간 예측 및 데드라인 만족을 어렵게 한다. 특히, 모바일 기기의 스토리지로 널리 사용되는 플래시메모리는 읽기/쓰기 연산의 속도가 비대칭적이고 비정기적으로 삭제 및 가비지 컬렉션 연산을 수행한다. 가비지 컬렉션은 일반적인 읽기/쓰기 연산보다 수십배가 느리기 때문에, 스토리지의 접근 속도를 크게 저하시키며 실시간 작업의 시간 요구조건을 충족할 수 없게 하는 요인이 된다[9]. 또한, 여러 작업이 동시에 실행되며 플래시메모리를 접근할 때 간섭으로 인한 스토리지 접근 지연은 더욱 가중된다. 실시간 작업의 스토리지 접근 시 이와 같은 예측이 어려운 지연을 없애기 위한 방법으로 최근 새롭게 출현한 고속 스토리지를 활용할 수 있다. 인텔의 옵테인 등 고속 스토리지는 플래시메모리보다 10배 이상 빠른 접근이 가능하며 삭제 연산이나 가비지 컬렉션 등을 수행하지 않아 접근 속도가 비교적 균일하므로 실시간 작업의 실행 시간 예측에 큰 도움이 된다[10].

그러나 실시간 작업을 고속 스토리지에서 실행하더라도 시분할 운영체제가 추구하는 I/O 방식을 유지한다면 여전히 예측 불가능한 지연이 발생한다[11]. 시분할 운영체제의 I/O 방식은 봉쇄형(blocking) 입출력을 사용하며, 이는 I/O를 요청한 작업으로부터 CPU를 선점하고 I/O 처리가 완료되어 스토리지로부터 인터럽트가 발생할 때 해당 프로세스를 재개하는 식으로 동작하기 때문이다. 즉, 고속 스토리지의 탑재로 I/O 시간의 예측이 어느 정도 가능해지더라도 I/O 완료 후 다시 CPU에서 실행되기까지 큐에서 기다리는 시간이 일정하지 않아 실시간 작업의 데드라인 보장을 어렵게 한다[12]. 현재 리눅스 등 시분할 운영체제의 CPU 스케줄러가 실시간 작업에 높은 우선 순위를 부여하므로 I/O가 끝난 후 CPU를 얻기까지의 시간은 비교적 짧은 편이다. 그러나, 시분할 운영체제의 CPU 스케줄러는 할당 시간이 남아 있는 프로세스로부터 CPU를 선점하지 않기 때문에 비록 실시간 작업의 I/O가 완료되었더라도 곧바로 CPU를 사용한다는 보장이 없다. 이러한 문제점을 해결하기 위해 본 논문에서는 실시간 작업이 고속 스토리지를 통한 I/O를 수행하더라도 문맥교환(context switch)을 하지 않고 CPU를 계속 보유하여 I/O 후 곧바로 CPU에서 실행될 수 있도록 한다. 즉, 실시간 작업이 I/O를 요청하더라도 프로세스를 봉쇄시키지 않도록 하여 기존의 봉쇄형 입출력으로 인한 CPU 대기 및 문맥교환 오버헤드를 없애 데드라인을 만족할 수 있도록 한다.

모바일 기기의 실시간 작업 지원을 위한 두 번째 이슈는 CPU 스케줄링이다. 할당 시간에 기반한 선점형 스케줄링 방식에서는 동시 작업의 수가 늘어남에 따라 CPU에 대한 대기 시간도 길어질 수밖에 없으며, 이는 실시간 작업의 데드라인을 만족시키지 못하는 결과를 초래하게 된다. 한편, 최근 모바일 기기에 다수의 코어가 장착되면서 실시간 작업의 시간 요구조건을 충족하기 위해 실시간 작업을 전담하는 코어를 두는 방식을 사용할 수 있다. 즉, 특정 코어를 실시간 작업에 전담하고 실시간 작업이 스토리지 접근을 하는 도중에도 해당 코어를 점유하면서 고속 입출력 직후 곧바로 CPU 실행을 가능하게 할 경우 기존의 스케줄러에서처럼 CPU의 대기 큐에서 작업들이 기다리면서 실행 시간 예측을 어렵게 하는 문제를 해결할 수 있다. 비록 실시간 작업이 I/O를 진행하는 동안 해당 코어의 CPU 사이클이 낭비되나, 고속 스토리지가 충분히 빠른 입출력을 제공하기 때문에 이로 인한 CPU 이용률 저하는 크지 않을 것이기 때문이다.

모바일 기기의 실시간 작업 지원을 위한 세 번째 이슈는 메모리 관리이다. 실시간 작업의 데드라인을 보장하기 위한 가장 확실한 방법은 작업 집합 전체를 메모리에 한꺼번에 로드하는 방식이지만 이런 방식은 멀티태스킹 중인 다른 작업들의 성능을 지나치게 저하시키는 부작용을 초래한다. 즉, 실시간 작업의 데드라인을 100% 보장하기 위해서는 스토리지 접근을 완전하게 제거해야 하지만, 그럴 경우 멀티태스킹 환경에서 지나친 메모리 요구량을 발생시킨다[13]. 따라서, 차선책으로 실시간 작업의 메모리 부재율이 일정 수준 이하가 될 정도의 메모리 용량을 할당하는 방식이 사용될 수 있다. 이는 모바일 기기에서 실행되는 실시간 작업이 연성 실시간 작업으로 작은 비율의 데드라인 미스를 허용하는 것이 실제 시스템에서는 좀 더 현실적인 접근에 해당하기 때문이다. 또한, 메모리 부재로 인해 스토리지 접근이 발생하더라도 본 논문에서는 실시간 작업을 위한 고속 스토리지 사용 및 전담코어 스케줄링을 통해 데드라인 미스 가능성은 높지 않게 된다. 실시간 작업에 할당될 메모리 크기를 결정하기 위해서는 메모리 부재율에 대한 목표 비율을 정하고 해당 비율을 만족하기 위한 메모리 할당 크기를 한계 효

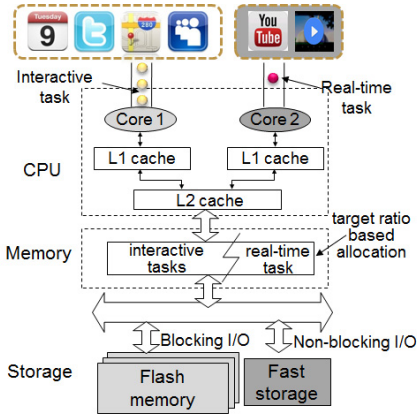


**그림 1. 실시간 작업 지원용 모바일 기기 아키텍쳐**
**Fig. 1. A mobile device architecture for supporting real-time tasks.**

용에 근거한 모델을 통해 구할 수 있다[14]. 이러한 방식은 실시간 작업 전체를 메모리에 상주시키는 전통적인 방식에 비해 작업 집합의 동적 변화를 파악하고 그에 소요되는 최소한의 메모리만을 할당한다는 측면에서 멀티태스킹 환경의 메모리 효율을 극대화할 수 있는 장점이 있다.

그림 1은 상기의 CPU, 메모리, 스토리지 관리 기법들을 채택하는 실시간 작업 지원용 모바일 기기 아키텍쳐를 보여주고 있다. 스토리지는 기존의 플래시메모리와 고속 스토리지의 조합이 사용되며, 고속 스토리지는 실시간 작업 및 그 데이터를 보관하는 용도로 사용된다. CPU 코어 중 일부가 실시간 작업 전담 코어로 사용되며, 실시간 작업이 활성 상태가 아닌 동안에는 해당 코어가 다른 작업에 할당되는 것을 허용한다. L1 캐쉬는 각 코어별로 할당되고 L2 캐쉬는 코어끼리 공유한다. 실시간 작업에 대해서는 비봉쇄형 입출력(non-blocking I/O)을 수행하며, 그 외의 작업들은 기존 시분할 운영체제와 마찬가지로 봉쇄형 입출력을 적용하여 CPU 이용률을 높인다. 한편, 메모리는 실시간 작업과 비실시간 작업들이 공유하며, 실시간 작업의 경우 메모리 부재율이 일정 수준 이하가 되는 크기만큼을 동적으로 할당한다.

## Ⅲ. 성능 평가

본 장에서는 모바일 기기에서 실시간 작업과 대화형 작업을 재현하는 실험을 통해 본 논문에서 소개한 기법들의 효과를 분석한다. 본 논문에서 소개한 방식과의 비교 대상으로는 전통적인 시분할 운영체제가 사용하는 방

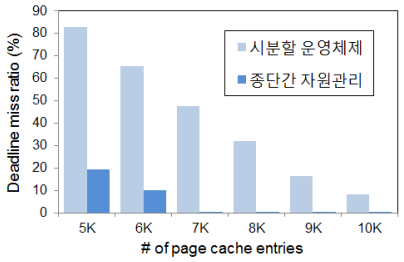


**그림 2. 페이지 캐시 크기 변화에 따른 데드라인 미스율**
**Fig. 2. Deadline miss ratio as the size of the page cache is varied.**

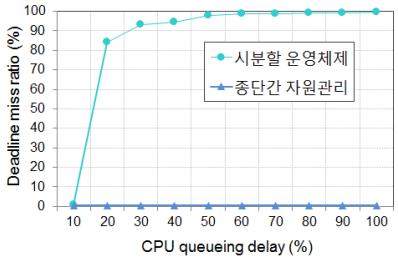


**그림 3. CPU 대기열의 지연에 따른 데드라인 미스율**
**Fig. 3. Deadline miss ratio as the CPU queueing delay is varied.**

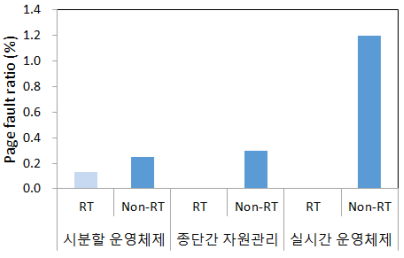


**그림 4. 실시간 작업과 대화형 작업의 메모리 부재율**
**Fig. 4. Page fault ratio of real-time tasks and interactive tasks.**

식, 즉 실시간 작업의 특성을 고려하지 않고 CPU, 메모리, 스토리지 등의 자원을 경쟁시키는 방식을 사용했다. CPU 스케줄링의 경우 실시간 작업 전담 코어를 사용하지 않고 선점형 스케줄링을 진행했으며, 메모리 및 스토리지 관리의 경우 실시간 작업과 대화형 작업이 대등하게 경쟁하는 방식을 사용했다.

그림 2는 실시간 작업용 전담 코어를 두고 고속 스토리지를 통한 비봉쇄형 입출력을 사용한 방식("종단간 자원 관리 기술"로 지칭)과 전통적인 시분할 운영체제의 방식을 비교해서 보여주고 있다. 그림에서 $x$축은 메모리 페이지 캐시 엔트리의 수를 나타내며 $y$축은 실시간 작업

의 데드라인 미스율을 나타내고 있다. 그림에서 보는 것처럼 종단간 자원 관리 기술은 기존 시분할 운영체제의 데드라인 미스를 92%까지 줄이는 것을 확인할 수 있다. 특히, 종단간 자원 관리 기술은 캐시 크기가 7K인 경우 데드라인 미스를 거의 발생시키지 않는 반면, 기존 시분할 운영체제는 48%의 데드라인 미스율을 나타내었다. 그림 3은 CPU 큐에서 대기 중인 작업들의 시간 분포에 따른 데드라인 미스율을 보여주고 있다. 대기 시간 100%의 의미는 실시간 작업이 큐에 도착했을 때 현재 실행 중인 프로세스가 그 CPU 할당 시간의 100%를 사용해야 하는 경우를 의미한다. 그림에서 보는 것처럼 시분할 운영체제의 데드라인 미스율은 대기 시간 증가에 따라 급격히 증가하는 반면, 종단간 자원 관리 기술은 꾸준히 좋은 성능을 나타내고 있다.

그림 4는 종단간 자원 관리 기술과 전통적인 시분할 운영체제의 메모리 부재율을 비교해서 보여주고 있다. 그림에서는 실시간 작업(RT)과 대화형 작업(Non-RT)의 메모리 부재율을 각각 보여주고 있다. 그림에서 보는 것처럼 종단간 자원관리 기술은 실시간 작업의 메모리 부재율을 일정 수준 이하의 매우 낮은 값으로 보장하고 있음을 확인할 수 있다. 그에 비해 시분할 운영체제의 경우 실시간 작업의 요구를 제대로 반영하지 못하고 메모리 부재율이 높게 발생하는 것을 확인할 수 있다. 이는 시분할 운영체제가 모든 앱을 동일한 기준으로 경쟁시키기 때문에 나타난 결과로 볼 수 있다. 한편, 종단간 자원 관리 기술의 경우 대화형 작업의 성능이 다소 저하되었지만 이는 부족한 자원 상황으로 인해 발생하는 필연적인 결과로 볼 수 있다. 이를 확인하기 위해 실시간 작업 전체를 메모리에 상주시키는 실시간 운영체제 방식을 비교 대상에 추가했으며, 그림에서 보는 것처럼 실시간 운영체제의 경우 실시간 작업의 메모리 부재율은 0이지만 대화형 작업의 성능저하가 뚜렷이 나타나는 것을 확인할 수 있다. 즉, 종단간 자원 관리 기술은 실시간 작업의 요구사항을 충족하면서도 대화형 작업에 대해 주어진 자원으로 제공할 수 있는 가장 합리적인 수준의 결과를 도출한다고 볼 수 있다.

## Ⅳ. 결 론

최근 모바일 기기에서 실시간 작업이 실행되는 사례가 늘면서 종래의 시분할 운영체제가 추구하던 자원 관리 정책에 대한 재고가 필요한 시점에 이르렀다. 본 논문에서는 모바일 기기에서 실시간 작업이 여러 작업들과 함께 실행될 때 다른 작업의 간섭을 받지 않고 데드라인을 만족할 수 있는 자원 관리 정책에 대해 살펴보았다. 특히, 실시간 작업을 위한 새로운 아키텍쳐와 CPU, 메모리, 스토리지로 이어지는 종단간 자원 관리를 통하여 실시간 제약 조건을 만족하면서 자원의 효율적인 관리가 가능한 방안을 알아보았다. 제안된 방안은 실시간 작업을 위한 전담 CPU 코어의 사용, 메모리 부재율을 일정 수준 이하로 하는 메모리 할당 기법, 고속 스토리지 탑재 후 문맥교환 없이 I/O을 실행할 수 있게 하는 비봉쇄형 입출력 등을 사용하여 실시간 작업의 데드라인 미스를 최소화하였다. 또한 제안된 기법은 실시간 작업의 요구를 일정 수준으로 보장하면서도 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] 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] Google Pixel 6a, https://store.google.com/gb/product/pixel_6a

[5] 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

[6] 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

[7] 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

[8] 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

[9] O. Kwon, K. Koh, J. Lee, and H Bahn, "FeGC: an efficient garbage collection scheme for flash memory based storage systems," Journal of Systems and Software, vol. 84, no. 9, pp. 1507-1523, 2011.
DOI: https://doi.org/10.1016/j.jss.2011.02.042

[10] 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

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

[12] E. Lee, Y. Kim, and H Bahn, "QoS management of real-time applications in NVRAM-based multi-core smartphones," Int'l Conf. Information Science & Applications (ICISA), pp. 1-4, 2014.
DOI: https://doi.org/10.1109/ICISA.2014.6847452

[13] 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

[14] H. Bahn, "Real-time task aware memory allocation techniques for heterogeneous mobile multitasking environments," The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), vol. 22, no. 3, pp. 43-48, 2022.
DOI: https://doi.org/10.7236/JIIBC.2022.22.3.43

## 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 National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2019R1A2C1009275) and also by the ICT R&D program of MSIT/IITP (2020-0-00121, development of data improvement and dataset correction technology based on data quality assessment).