The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 21, No. 6, pp.71-76, Dec. 31, 2021. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2021.21.6.71
JIIBC 2021-6-10

고속 스토리지를 이용한 실시간 IoT 시스템의 전력 절감 최적화 기술

Optimization Techniques for Power-Saving in Real-Time IoT Systems using Fast Storage Media

윤수지[*], 박희진[*], 조경운[**], 반효경[**]

Suji Yoon[*], Heejin Park[*], Kyungwoon Cho[**], Hyokyung Bahn[**]

요 약 최근 사물인터넷의 데이터가 대용량화됨에 따라 실시간 시스템의 메모리 전력 소모가 급증하고 있다. 이는 실시간 시스템이 태스크 전체를 항상 메모리에 올려놓고 처리함으로 인한 DRAM 용량 증가에 기인한다. 본 논문은 최근 각광 받는 고속 스토리지를 활용하여 실시간 태스크의 일부를 스토리지에 내려놓고 필요시 메모리에 올리는 전력 절감 기술을 제안한다. 또한, 이를 CPU의 동적 전압조절 기법과 결합하여 CPU와 메모리의 전력 절감을 동시에 최적화한다. 제안하는 기술은 CPU의 유휴시간을 최대한 줄이는 전압 모드를 결정하는 동시에 메모리 크기를 최소화하는 스왑 비율을 결정하여, 태스크의 데드라인을 어기지 않으면서 전력 소모를 최소화하는 최적의 조합을 탐색한다. 시뮬레이션 실험을 통해 제안하는 기술이 실시간 시스템의 전력 소모를 크게 줄임을 보인다.

Abstract Recently, as the size of IoT data grows, the memory power consumption of real-time systems increases rapidly. This is because real-time systems always place entire tasks in memory, which increases the demand of DRAM significantly. In this paper, we adopt emerging fast storage media and move a certain portion of real-time tasks from DRAM to storage. The part of tasks in storage are, then, loaded into memory when they are actually used. We incorporate our memory/storage power-saving into the dynamic voltage/frequency scaling of processors, thereby optimizing power consumptions in CPU and memory simultaneously. Specifically, the proposed technique aims at minimizing the CPU idle time and the DRAM memory size by determining appropriate voltage modes of CPU and the swap ratio of memory, without violating the deadlines of all tasks. Through simulation experiments, we show that the proposed technique significantly reduces the power consumption of real-time systems.

Key Words : fast storage, swapping, real-time task scheduling, dynamic voltage frequency scaling.

[*]비회원, 이화여자대학교 컴퓨터공학과
S. Yoon and H. Park contributed equally to this work.
[**]정회원, 이화여자대학교 컴퓨터공학과
접수일자 2021년 9월 30일, 수정완료 2021년 11월 8일
게재확정일자 2021년 12월 10일

Received: 30 September, 2021 / Revised: 8 November, 2021 / Accepted: 10 December, 2021
[*]Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

Ⅰ. 서 론

최근 다양한 분야에서 사물인터넷(IoT)에 대한 관심이 높아지고 있다[1, 2]. 사물인터넷은 사물에 센서 및 전송 모듈을 탑재하여 물리적 데이터를 센싱하고 이를 저장, 처리 후 무선 통신으로 인터넷에 연결하는 기술을 뜻한다. 이러한 사물인터넷용 시스템은 배터리 기반으로 동작하는 일종의 실시간 시스템으로 볼 수 있다. 따라서, 사물인터넷용 시스템은 (1) 실시간 태스크의 데드라인을 만족하면서, (2) 배터리의 전력 소모를 줄이는 것이 시스템 설계의 중요한 목표이다. 본 논문은 실시간 태스크의 데드라인을 만족하면서 시스템의 전력 절감을 위해 메모리의 선별적 스왑(partial swap) 기법을 제안하고 이를 CPU의 동적 전압조절(DVFS: dynamic voltage/frequency scaling) 기법과 결합하여 동시에 최적화하는 기술을 제안한다.

동적 전압조절 기술은 태스크의 작업량이 CPU의 처리 용량에 크게 못 미치는 경우 CPU의 공급 전압을 낮추어 전력 소모를 줄이는 방법을 말한다[3]. CPU에 공급되는 전압을 낮추면 작업의 수행 시간은 늘어나지만 데드라인을 만족하는 범위 내에서의 전압 조절은 낮아진 전압의 제곱에 비례해 전력 소모가 절감되므로 실시간 시스템의 요구 조건을 만족하면서 전체적인 에너지가 절약되는 효과를 얻게 된다.

한편, 최근 데이터의 크기가 증가하면서 사물인터넷 기기의 메모리 용량이 늘어나 전력 소모가 급격히 증가하고 있다. 메모리 매체인 DRAM은 모든 셀이 지속적인 전원재공급 연산을 수행해야 내용이 유지되는 휘발성 매체로 메모리 크기 증가 시 전력소모도 그에 비례해 늘어난다. 한편, 통상적으로 실시간 시스템은 데드라인 보장을 위해 모든 태스크를 메모리에 올려놓고 처리하므로 범용 시스템처럼 당장 사용하지 않는 데이터를 스토리지에 내려놓는 스왑 기법을 사용하지 못한다[4]. 이는 CPU가 사용하려는 코드나 데이터가 느린 스토리지에 있을 경우 시간 예측이 어려워 실시간 태스크의 데드라인 보장을 어렵게 하기 때문이다. 본 논문은 최근 각광 받는 차세대 고속 스토리지를 활용하여 실시간 태스크의 일부를 스토리지에 내려놓고 필요시 메모리에 올리는 선별적 스왑 기술을 제안한다. 이러한 방식은 실시간 시스템의 메모리 크기를 줄여 전력 절감에 기여할 수 있다.

한편, CPU의 전압조절기술과 메모리의 스왑 기술을 사용할 경우 태스크의 실행시간이 늘어나므로 이에 대한 적절한 예측을 통해 데드라인을 어기지 않게 하는 모델링이 필요하다. 본 논문에서는 태스크 별 CPU의 전압 모드와 스왑 비율을 동시에 최적화하여 모든 태스크가 데드라인을 만족하면서 전력 소모를 최소화하는 조합을 유전 알고리즘을 이용해서 탐색한다. 다양한 워크로드 환경에서의 시뮬레이션 실험을 통해 제안하는 기술이 실시간 시스템의 전력 소모를 크게 줄임을 보인다.

Ⅱ. 제안하는 기법

전압조절이 가능한 CPU와 메모리 스왑을 지원하는 실시간 IoT 시스템의 태스크 집합을 $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ 라고 할 때, 각 태스크 τ_i는 $\langle C_i,\ T_i,\ M_i \rangle$로 정의되며, C_i는 태스크 τ_i의 최악실행시간, M_i는 τ_i의 메모리 사용량, T_i는 태스크의 실행주기를 나타낸다. 본 논문의 시스템 모델은 실시간 태스크 스케줄링 관련 기존 연구들이 채택하고 있는 가정에, 선별적 스왑 등 새로운 기능을 바탕으로 다음과 같이 가정한다.

가정1. 각 태스크는 독립적이며, 다른 태스크의 결과값에 영향을 미치지 않는다.

가정2. 스케줄링 시 문맥전환, 즉 CPU가 다른 태스크로 이양되는 오버헤드는 무시할 수 있다.

가정3. CPU의 클럭 빈도가 결정되면, 공급 전압은 그에 맞게 조정된다.

가정4. 스왑용 스토리지의 접근시간은 예측 가능하다. 즉, 스왑아웃된 태스크의 크기를 알면 메모리로 적재하는 데에 걸리는 최악 시간은 미리 예측할 수 있다.

한편, 태스크의 최악실행시간 C_i는 CPU의 전압 조절과 메모리 스왑을 사용할 경우 늘어난 실행시간을 고려하여 가장 긴 시간 경로에 기반해 그 값이 재설정되어야 한다. 본 논문에서는 이를 함수 f를 통해 정의한다. 이러한 모델에 기반해 실시간 태스크의 스케줄 가능 여부는 시스템의 이용률 U가 다음의 조건을 만족하면 된다.

$$U = \sum_{i=1}^{n} \frac{f(C_i)}{T_i} \quad \leq 1 \qquad (1)$$

이러한 조건을 만족할 경우 EDF(earliest deadline first) 알고리즘을 적용하여 주어진 태스크 집합에 대한 유효한 스케줄을 얻을 수 있다[3]. EDF는 태스크 집합에서 가장 데드라인이 임박한 태스크를 가장 먼저 CPU에

표 1. 태스크 집합의 예
Table 1. An example of task sets

태스크	실행시간	주기
τ_1	1	10
τ_2	1	14
τ_3	2	16

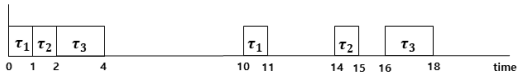

그림 1. 태스크 집합의 EDF 스케줄링 결과
Fig. 1. A scheduled task set by EDF

서 실행하는 방법이다.

본 논문이 제안하는 기법의 설명을 위해 표 1의 간단한 예제를 살펴보자. 총 3개의 태스크 τ_1, τ_2, τ_3 로 구성된 태스크 집합에 대해 시스템의 이용률을 구하면 U = 1/10 + 1/14 + 2/16 = 0.296 〈 1이 되어 스케줄이 가능함을 알 수 있다. 그림 1은 표 1의 예에 대한 EDF 스케줄링 결과를 보여주고 있다. 그림에서 보듯이 모든 태스크의 데드라인을 만족하지만, 유휴구간이 많이 발생하는 것을 알 수 있다. 이러한 유휴구간은 CPU의 전압을 낮추어 줄일 수 있다. 예를 들어, τ_2와 τ_3를 정상 상태 대비 0.5의 CPU 클럭 빈도로 수행할 경우, 최악실행시간 $f(\tau_2)$와 $f(\tau_3)$은 각각 1에서 2로, 2에서 4로 늘어나게 된다. 이에 따라, U = 1/10 + 2/14 + 4/16 = 0.493으로 증가하지만, 여전히 U 〈 1을 만족하여 스케줄링이 가능함을 알 수 있다. 그림 2는 이를 적용한 EDF 스케줄링 결과로, 유휴구간이 그림 1에 비해 크게 줄었음을 알 수 있다. 이에 따라 CPU의 전력 소모 또한 줄어들게 된다.

한편, 실시간 시스템에서는 스토리지를 사용할 경우 태스크의 최악실행시간 예측이 어려우므로 가상메모리스왑 기능을 사용하지 않고 모든 태스크를 메모리에 올려놓는 쉐도잉(shadowing) 방법을 사용한다. 이는 스토리지 접근이 매우 느리고 스토리지의 상태에 따라 접근 시간이 달라져 예측을 어렵게 하기 때문이다[5]. 하드 디스크의 경우 헤드의 이동시간 및 디스크 스케줄링 방식에 따라 요청 데이터를 읽어오는 시간이 달라지며, 플래시 메모리의 경우 내부 상태에 따라 가비지 컬렉션 수행 등으로 접근 시간의 변동폭이 크게 달라질 수 있다[6, 7, 8]. 본 논문은 최근 새롭게 등장한 고속 스토리지가 빠르고 예측가능한 접근 시간을 가진다는 점에 착안하여 태스크의 일부를 스토리지에 두고 해당 태스크가 실행될 때에만 메모리로 올리는 선별적 스왑 기술을 사용한다. 이는

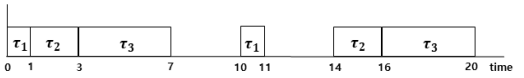

그림 2. CPU 전압조절 기법에 의한 스케줄링 결과
Fig. 2. A scheduled task set by voltage scaling

전체 시스템의 DRAM 크기를 줄여 전력 절감에 기여하게 된다. 즉, 모든 태스크를 모든 시간에 메모리에 올려놓는 것이 아니라, 태스크의 일부를 스토리지에 내려놓고 필요시에만 DRAM에 한시적으로 올리는 방식으로, 스토리지 접근에 부가적인 시간이 소요되어 태스크의 최악실행시간이 그림 3과 같이 늘어나게 된다. 하지만, 그림 2에서 보는 것처럼 스케줄링 결과 여전히 유휴구간이 존재하는 경우 스왑을 일부 태스크에 선별적으로 적용하면 앞뒤 빈 슬롯을 더 채워 U 〈 1을 만족하면서도 가능한 스케줄을 찾을 수 있다.

선별적 스왑에서 스왑 비율을 높이면 DRAM 크기 절감으로 메모리 전력 소모를 줄일 수 있지만, 스왑 시간이 늘어 태스크의 최악실행시간은 증가하게 된다. 스왑 비율을 낮추면 태스크의 최악실행시간이 적게 늘어나지만 전략 절감 효과는 줄어든다. 한편, 스왑 비율의 조절과 CPU의 전압 조절은 서로 상충관계에 있다. 즉, 스케줄링 가능한 범위에서 스왑 비율을 더 높일 경우 CPU의 전압 조절 가능성은 낮아질 수 있어 이 두 가지를 조합하여 전력 절감을 극대화하는 방법이 필요하다. 또한, 전압 조절 기법과 스왑은 중첩 실행이 가능하므로 이를 고려한 두 기법의 최적 적용이 필요하다. 예를 들어, 전압 상태가 4가지이고 스왑 비율이 5가지일 경우 모든 태스크에 대해 가능한 CPU와 메모리의 상태는 총 20가지가 존재한다. 태스크의 수가 n인 경우 20^n가지 경우의 수가 존재하며, 이를 모두 살펴보는 것은 NP 하드 문제에 해당한다. 이에 본 논문은 유전 알고리즘을 이용하여 각 태스크의 실행 전압 상태와 메모리 스왑 비율을 최적화한다.

두 기법 적용에 의한 태스크 τ_i의 최악실행시간 $f(t_i)$는 다음과 같이 정의된다.

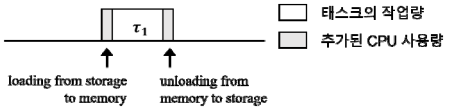

그림 3. 선별적 스와핑 기법에 의한 최악 실행 시간
Fig. 3. A worst execution time by partial swapping

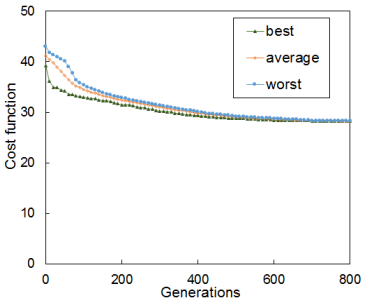

그림 4. 유전 알고리즘의 해집합 수렴 그래프
Fig. 4 Convergence of the solutions as time progresses

$$f(t_i) = \max\{f_{DVFS}(t_i)*f_{swap}(t_i)\}+\mu_i \quad (2)$$

이때 $f_{DVFS}(t_i)$와 $f_{swap}(t_i)$는 각각 DVFS와 스왑에 의해 재설정된 최악실행시간을 나타내며, 이들은 중첩실행이 가능하므로 더 느린 쪽에 의해 최악실행시간이 결정된다. μ_i는 스왑 사용을 위한 CPU의 입출력 커맨드 수행에 소요되는 지연상수를 나타낸다. $f_{DVFS}(t_i)$와 $f_{swap}(t_i)$는

$$f_{DVFS}(t_i) = t_i/\epsilon \quad (3)$$

$$f_{swap}(t_i) = t_i + 2*latency_{swap}(r_i*s_i) \quad (4)$$

로 정의되며, ϵ은 CPU의 전압 모드 파라미터이고, r_i는 스왑 비율, $latency_{swap}$는 데이터 크기에 따른 스왑 스토리지 접근 시간을 나타낸다.

III. 유전 알고리즘을 이용한 최적화

유전 알고리즘은 최적화 문제에서 집단 유전학의 자연진화 원리를 모방하는 확률적 알고리즘이다. 본 논문에서는 각 태스크 별 CPU와 메모리의 상태를 전력 소모 관점에서 최적화하기 위해 steady-state 유형의 유전 알고리즘을 이용한다[9]. 유전 알고리즘에서는 초기 해집합 생성 후 부모해 선택 및 교차, 변이 연산 등으로 새로운 해집합 구성을 반복하여 최적의 수렴된 해집합을 생성한다.

해의 표시는 각 태스크 별로 CPU의 전압 상태와 메모리의 스왑 비율을 나타내는 2개의 문자열로 구성되며, 문자열의 길이는 총 태스크의 수와 같다. 본 논문에서는 CPU의 전압 상태를 클럭 빈도에 근거해 {1, 0.5, 0.25, 0.125} 4개로 정의하고 이를 각각 0, 1, 2, 3의 2비트 값으로 표시하였으며, 스왑 비율은 {0, 0.125, 0.25, 0.5}의 4가지로 정의하고 2비트로 표시하였다. 해의 적합도 평가는 스케줄링 결과 발생하는 전력 소모량으로 하였으며, 해당 해의 이용률이 1을 초과하여 스케줄링이 불가능할 경우 패널티 값을 부여하였다.

해집합의 크기는 100으로 하였으며, 초기 해집합은 랜덤하게 발생시켰다. 진화를 위한 부모해의 선택은 적합도 값을 바탕으로 해집합 내에서 가장 좋은 해가 선택될 확률이 가장 나쁜 해가 선택될 확률의 4배가 되도록 하는 정규화 방법을 사용하였다[9]. 2개의 부모해 선택 후 교차 연산은 유전 알고리즘에서 흔히 사용하는 1점 교차 연산을 사용하여 교차점 좌우를 서로 다른 부모해에서 계승하도록 하였다. 교차 연산 후 스트링 내의 일부값을 확률적으로 교란시키는 변이 연산을 수행하였으며, 유전 알고리즘에서 흔히 사용하는 방식인 스트링 내 임의 지점의 값을 교란시키는 방법을 사용하였다. 생성된 자식해는 새로운 해집합에 삽입하며 대신 기존 해집합에서 적합도가 가장 낮은 해를 제외하였다. 세대 진화의 반복 횟수는 해집합 내의 다양성이 일정 수준 이하로 떨어져 해들이 수렴할 때까지로 하였다. 그림 4는 유전 알고리즘의 수렴성을 입증하기 위한 그래프로 세대 진행에 따른 해집합 내 최적해, 최악해, 전체 해의 평균 적합도 값을 보여주고 있다.

IV. 성능 평가

본 논문에서 제안한 기법의 성능 평가를 위해 다양한 부하의 실시간 태스크 집합에 대해 시뮬레이션을 수행하였다. 성능 비교를 위해 DVFS나 스왑을 사용하지 않는 방식(Original)과 DVFS만 사용하는 방식(DVFS), 스왑만 사용하는 방식(Swap)을 비교 대상으로 하였으며, DVFS와 Swap 역시 유전 알고리즘을 통한 최적화 후 EDF로 스케줄링하였다.

그림 5는 실시간 태스크의 작업 부하에 따른 Original, DVFS, Swap, DVFS-Swap의 에너지 소모량을 보여준다. 그래프에서 y축은 Original의 값을 기준으로 정규화하여 표시하였다. 그림에서 보는 것처럼 DVFS-Swap은 모든 경우에 있어 가장 좋은 결과를 나타내었다. DVFS-Swap의 에너지 절감은 Original 대비 평균 23.8%, 최대 45.0%였으며, 특히 부하가 낮은 상황에서 더욱 우수한 결과를 나타내었다. 이는 부하가 낮을수록 유휴 구간을 활용한 전력 절감 가능성이 높기 때문이다. DVFS, Swap과의 비교 결과는 각각 9.3%, 15.3%의 평균 에너지 절감을 나타내었다.

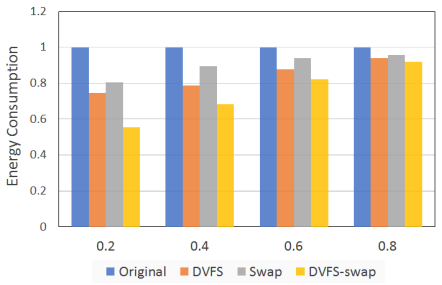

그림 5. 총 에너지 소모 비교
Fig. 5 Comparison of total energy consumptions.

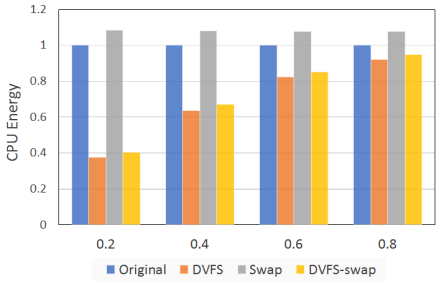

그림 6. CPU 에너지 소모 비교
Fig. 6 Comparison of CPU energy consumptions.

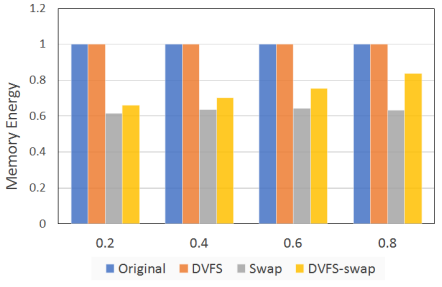

그림 7. 메모리 에너지 소모 비교
Fig. 7 Comparison of memory energy consumptions.

그림 6과 7은 4가지 기법에 대해 CPU와 메모리의 에너지 소모량을 구분하여 보여주고 있다. 그림에서 보듯이 전압조절 기법을 사용한 DVFS와 DVFS-Swap이 CPU의 에너지 소모량을 크게 줄임을 확인할 수 있다. 또한, 태스크의 부하가 증가할수록 전압조절 기법의 효과가 줄어드는데, 이는 전압조절을 통해 유휴 구간을 활용할 수 있는 가능성이 줄어들기 때문이다. 그림 7에서 보듯이 스왑 기술을 활용한 Swap과 DVFS-Swap이 그렇지 않은 기법들에 비해 메모리 전력 소모를 줄임을 확인할 수 있다. 이는 선별적 스왑을 통해 DRAM의 크기를 줄여 불필요한 유휴 전력 소모를 줄였기 때문이다. 한편, 스왑을 이용할 경우 태스크를 고속 스토리지와 메모리 사이에 올리고 내리는 데에 추가적인 CPU의 사용이 필요하여 CPU의 전력소모는 다소 늘어나는 것을 확인할 수 있다. 그러나 그림 6에서 보듯이 이러한 현상은 Swap에서만 나타나는 것을 확인할 수 있으며, DVFS-Swap을 통해 DVFS를 함께 사용할 경우 그 오버헤드는 거의 나타나지 않는다는 것을 알 수 있다.

V. 결 론

본 논문에서는 IoT용 실시간 시스템에 고속 스토리지를 탑재하여 선별적 메모리 스왑 기능을 지원함으로써 DRAM 메모리의 전력 절감을 크게 줄이는 기술을 제안하였다. 또한, CPU의 전압 조절 기법과 결합하여 태스크의 데드라인을 어기지 않으면서 CPU와 메모리의 전력 소모를 최소화하는 스케줄링을 유전 알고리즘으로 최적화하였다. 다양한 부하 환경에서의 실험을 통해 제안하는 기법이 태스크의 CPU 전압조절을 최적화한 기술과 메모리 스왑 비율을 최적화한 기술 대비 각각 9.3%와 15.3%의 전력 절감을 나타내었으며, 두 기법을 적용하지 않은 경우보다 평균 23.8%의 전력 절감 효과를 얻는 것을 확인할 수 있었다.

References

[1] J. Park and E. Park, "Performance evaluation of IoT cloud platforms for smart buildings," Journal of the Korea Academia-Industrial cooperation Society(JKAIS), vol. 21, no. 5, pp.664-671, 2020.
DOI: https://doi.org/10.5762/KAIS.2020.21.5.664

[2] J. Kim, "An implementation of massive IoT emulation testbed," The Journal of KIIT, vol. 19, no. 9, pp. 41-47, 2021.
DOI: https://doi.org/10.14801/jkiit.2021.19.9.41

[3] 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, 2019.
DOI: http://doi.org/10.3390/mi10060371

[4] 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: http://doi.org/10.1109/ACCESS.2019.2937852

[5] D. Kim, E. Lee, S. Ahn, and H. Bahn, "Improving the

storage performance of smartphones through journaling in non-volatile memory," IEEE Trans. Consumer Electronics, vol. 59, no. 3, pp. 556-561, 2013.
DOI: http://doi.org/10.1109/TCE.2013.6626238

[6] J. Park, H. Lee, S. Hyun, K. Koh, and H. Bahn, "A cost-aware page replacement algorithm for NAND flash based mobile embedded systems," Proc. ACM EMSOFT Conference, pp. 315-324, 2009.
DOI: https://doi.org/10.1145/1629335.1629377

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

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

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

저 자 소 개

윤 수 지(비회원)

- 2019년 3월 ~ : 이화여자대학교 컴퓨터공학과 학사과정
- 주관심분야 : 운영체제, 스토리지 시스템, 임베디드 시스템

박 희 진(비회원)

- 2019년 3월 ~ : 이화여자대학교 컴퓨터공학과 학사과정
- 주관심분야 : 운영체제, 스토리지 시스템, 임베디드 시스템

조 경 운(정회원)

- 1995년 2월 : 서울대학교 계산통계학과 학사
- 1997년 2월 : 서울대학교 전산과학과 석사
- 2012년 2월 : 서울대학교 컴퓨터공학부 박사
- 2000년 ~ 2016년: ㈜클루닉스 연구소장
- 2016년 4월 ~ : 이화여자대학교 임베디드소프트웨어연구센터 수석연구원/연구교수

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 research was funded by the ICT R&D program of MSIP/IITP (2019-0-00074, developing system software technologies for emerging new memory that adaptively learn workload characteristics) and (2020-0-00121, Development of data improvement and dataset correction technology based on data quality assessment).