The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 6, pp.121-126, Dec. 31, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.6.121
JIIBC 2024-6-18

# 클라우드와 엣지 서버의 실시간 태스크 오프로딩 효율성 분석

# Efficiency Analysis for Real-Time Task Offloading of Cloud and Edge Servers

**권가현*, 반효경****

Gahyeon Kwon*, Hyokyung Bahn**

**요 약** 최근 사물인터넷 시스템의 전력 절감을 위해 실시간 태스크를 클라우드 또는 엣지 서버로 오프로딩하는 기술이 활발히 연구되고 있다. 충분한 컴퓨팅 자원을 갖춘 클라우드 서버로의 오프로딩은 모바일 시스템의 컴퓨팅 부담을 크게 줄일 수 있으며, 안정된 네트워크 망을 통한 근거리 엣지 서버로의 오프로딩을 통해 실시간 태스크의 데드라인 조건을 더욱 세밀하게 충족할 수 있다. 본 논문에서는 모바일 실시간 시스템의 전력 절감을 위해 기존의 프로세서 동적 전압 조절 기술과 저전력메모리 기술에 엣지/클라우드 서버로의 태스크 오프로딩을 함께 적용한 최적화를 수행했을 때 클라우드와 엣지 서버의 효율성을 비교 분석한다. 특히, 태스크 집합의 부하와 네트워크 상황, 원격 서버의 컴퓨팅 능력 차이에 따라 클라우드와 엣지 서버로의 오프로딩이 어떠한 영향을 초래하는지 실험을 통해 비교 분석한다.

**Abstract** Recent research has been actively exploring the technology of offloading real-time tasks to cloud or edge servers to reduce power consumption in Internet of Things (IoT) systems. Offloading to cloud servers, which possess ample computing resources, can significantly reduce the computational burden on mobile systems. More recently, offloading to nearby edge servers via stable network connections has gained attention as a method to more effectively meet the deadline requirements of real-time tasks. This paper compares and analyzes the impact of cloud and edge servers when task offloading is combined with traditional dynamic voltage scaling (DVS) and low-power memory technologies to optimize power-saving in mobile real-time systems. Specifically, we experimentally compare the trade-offs between offloading to cloud versus edge servers, depending on task set load, network conditions, and the computing capabilities of remote servers.

**Key Words** : real-time task, offloading, cloud, edge, optimization, scheduling

*준회원, 이화여자대학교 컴퓨터공학과
**정회원, 이화여자대학교 컴퓨터공학과(교신저자)
접수일자 2024년 8월 31일, 수정완료 2024년 11월 23일
게재확정일자 2024년 12월 6일

Received: 31 August, 2024 / Revised: 23 November, 2024 / Accepted: 6 December, 2024
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

## Ⅰ. 서 론

의료, 국방, 주거, 환경 등 다양한 분야에서 사물인터넷(IoT) 및 인공지능(AI) 기술이 도입되면서, 센서를 통한 데이터 취득 및 전송, AI를 이용한 학습 및 추론 등이 실시간으로 이루어져야 하는 임베디드 시스템이 늘고 있다[1, 2, 3]. 이러한 시스템에서는 배터리 소모를 줄이면서 실시간 워크로드의 데드라인 제약조건을 만족하는 것이 필요하다[4, 5]. 메모리의 전력 소모를 줄이기 위해 저전력 메모리를 이용하여 인기 있는 데이터를 특정 영역에 집중 배치하는 기술이 연구된 바 있으며[6, 7], 태스크의 연산량이 적을 때 CPU의 처리 속도를 낮추는 DVFS(dynamic voltage/frequency scaling) 기술이 널리 연구된 바 있다[8]. 또한, 컴퓨팅 자원이 제한적인 모바일 시스템에서 직접 연산을 수행하는 대신 태스크를 원격의 서버로 오프로딩해서 실행하는 기술 또한 주목받고 있다[9, 10].

충분한 컴퓨팅 자원을 갖춘 클라우드 서버로의 오프로딩은 태스크 집합의 부하가 높은 상황에서 모바일 시스템 내의 CPU 연산량을 크게 줄여 에너지 절감을 얻을 수 있으며, 안정된 네트워크 망을 통한 근거리 엣지 서버로의 오프로딩은 예측 가능한 지연시간을 제공하여 실시간 시스템의 제약 조건을 만족하면서 모바일 시스템의 에너지를 절감하는 것이 가능하다. 한편, 이러한 전력 절감과 성능 개선의 최적화는 정교한 자원 사용 예측을 통해서만 가능하다[11].

본 연구에서는 모바일 실시간 시스템의 전력 절감을 위해 DVFS 기술과 저전력메모리 기술, 태스크 오프로딩 기술을 결합한 최적화 기술에서 클라우드 서버로의 오프로딩과 엣지 서버로의 오프로딩이 최적화의 성능에 어떠한 영향을 미치는지에 대해 광범위한 범위의 실험을 통해 분석한다.

본 논문의 연구 결과에 따르면 태스크 집합의 컴퓨팅 부하가 높은 상황에서는 클라우드로의 오프로딩이 효율적인 반면, 태스크 집합의 부하가 낮은 상황에서는 엣지 서버로의 오프로딩이 모바일 시스템의 에너지 절감 최적화에 효과적인 것으로 분석되었다. 또한, 클라우드/ 엣지 서버로부터 모바일 시스템까지의 네트워크 대역폭차이가 큰지 여부와 클라우드와 엣지 서버의 연산 능력 차이에 따라 두 환경으로의 오프로딩 간에 트레이드 오프 관계가 발생하는 것으로 분석되었다. 본 논문의 연구 결과를 토대로 클라우드와 엣지 서버의 다양한 환경적 특성 및 실시간 태스크 집합의 특성이 주어졌을 때 이에 적합한 태스크 오프로딩의 최적 설계를 수행할 수 있을 것으로 기대된다.

## Ⅱ. 태스크 실행 모델

본 연구에서의 태스크 실행 모델은 종래의 실시간 태스크 모델을 CPU 수행 속도 조절, 메모리 배치, 원격 실행 등의 최적화가 가능하도록 확장하였다[8]. 태스크의 집합은 $T = \{t_1, t_2, \cdots, t_n\}$으로 정의되며, 타겟 모바일 시스템은 DVFS 기능을 가진 CPU와, 저전력메모리인 NVM(non-volatile memory)이 추가된 메인 메모리로 구성된다. 또한, 오프로딩이 가능한 클라우드 및 엣지 서버들이 존재하는 것으로 가정한다. 태스크는 로컬 CPU 또는 원격의 클라우드/엣지 서버에서 실행 가능하며, 이에 따라 태스크 집합 $T$는 두 개의 부분집합 LOCAL과 REMOTE로 분할된다. 이때, 원격의 서버가 모바일 시스템보다 컴퓨팅 파워가 우수하므로 오프로딩 비율을 높이는 것이 일반적으로 더 효과적이지만, 오프로딩된 태스크의 실행결과가 데드라인 이내에 반환되어야 하고, 센서나 엑츄에이터에 커맨드를 내리는 제어형 태스크는 연산형 태스크와 달리 반드시 로컬 CPU에서 수행되어야 하므로 이러한 조건을 고려해서 오프로딩 비율을 결정해야 한다.

한편, 태스크가 2종의 메모리에 위치할 수 있으므로, 태스크 집합 $T$는 두 개의 부분집합 DRAM과 NVM으로 분할된다. 태스크 $t_i$는 $t_i = \langle WCET_i, Period_i, Data_i \rangle$로 정의되며, $WCET_i$는 $t_i$가 로컬 CPU에서 기본 클럭 속도로 실행될 때의 최악실행시간을 나타낸다. $Period_i$는 $t_i$의 실행주기를, $Data_i$는 $t_i$의 데이터 특성을 나타내며 $Data_i = \langle Size_i, Input_i, Output_i, Rd_i, Wr_i \rangle$로 정의된다. 이때, $Data_i$는 $t_i$의 메모리 풋프린트를 나타내며, $Input_i$는 $t_i$의 실행을 위한 입력 데이터, $Output_i$는 실행 후 출력 데이터 크기를 나타낸다. 또한, $Rd_i$와 $Wr_i$는 $t_i$의 실행 시 메모리 읽기와 쓰기 연산의 수를 각각 나타낸다. $t_i \in$REMOTE인 태스크를 수행하기 위해서는 $Size_i$를 먼저 원격의 서버로 전송해야 하며, 수행이 끝난 후 $Output_i$를 원격의 서버로부터 수신해야 한다. 본 연구는 주기적인 실시간 태스크를 다루며, 태스크의 데드라인은 $Period_i$에 의해 결정된다. 모든 태스크 $t_i \in T$에 대해 전체 주기 $Epoch_T$는 개별 태스크의 주기인 $Period_i$의 최소공배수로 정의된다.

본 연구에서는 실시간 시스템을 다루므로 네트워크가 일시적으로 단절되는 경우에도 태스크의 데드라인을 만

족해야 한다. 따라서, 원격 서버의 도움 없이 로컬 CPU를 최대 클럭 속도로 가동시 아래의 이용률 테스트를 만족해야 한다.

$$Util_{CPU} = \sum_{t_i \in T} \frac{WCET_i}{Period_i} \leq 1 \quad (1)$$

식 (1)을 만족할 경우 주어진 태스크 집합에 대해 선점형 스케줄링인 EDF(earliest deadline first) 알고리즘을 통해 가능한 스케줄을 얻을 수 있다. 본 연구는 DVFS를 사용하므로, 각 태스크의 최악실행시간은 CPU 클럭 속도에 맞게 재계산되어 아래의 테스트를 만족해야 한다.

$$Util_{CPU} = \sum_{t_i \in T} \frac{DVFS\left(WCET_i\right)}{Period_i} \leq 1 \quad (2)$$

이때, $DVFS\left(WCET_i\right)$는 DVFS 적용 후의 최악실행시간을 나타낸다. 또한, 본 연구에서는 DRAM뿐 아니라 NVM을 메모리로 사용하므로 아래의 테스트를 만족해야 한다.

$$Util_{CPU} = \sum_{t_i \in \mathrm{DRAM}} \frac{DVFS\left(WCET_i\right)}{Period_i} + \sum_{t_i \in \mathrm{NVM}} \frac{DVFS_{NVM}\left(WCET_i\right)}{Period_i} \leq 1 \quad (3)$$

$$DVFS_{NVM}\left(WCET_i\right) = DVFS\left(WCET_i\right) + \Delta_{Rd} * Rd_i + \Delta_{Wr} * Wr_i \quad (4)$$

이때, $\Delta_{Rd}$와 $\Delta_{Wr}$는 DRAM과 NVM 간의 읽기/쓰기 지연 시간차를 나타낸다. 본 논문의 모델이 원격 서버로의 오프로딩을 지원하므로, CPU 이용률은 또한 아래 식을 만족해야 한다.

$$Util_{CPU} = \sum_{t_i \in \mathrm{LOCAL} \cap \mathrm{DRAM}} \frac{DVFS\left(WCET_i\right)}{Period_i} + \sum_{t_i \in \mathrm{LOCAL} \cap \mathrm{NVM}} \frac{DVFS_{NVM}\left(WCET_i\right)}{Period_i} + \sum_{t_i \in \mathrm{REMOTE}} \frac{Comnd_i}{Epoch_T} \leq 1 \quad (5)$$

이때, $Comnd_i$는 매 $Epoch_T$ 마다 원격의 서버에 배정된 태스크 $t_i$의 오프로딩을 진행하도록 CPU가 네트워크 모듈에 지시하는 커맨드 시간을 뜻한다. 태스크의 실행위치와 CPU 모드가 결정되면 $Epoch_T$가 시작될 때 REMOTE에 속한 태스크들을 CPU가 먼저 오프로딩 지시를 내려 로컬과 원격 태스크 실행의 병렬성을 높인다. 한편, 원격의 서버에서 수행되는 태스크의 데드라인을 보장하기 위해 제안한 기법은 다음 식을 만족해야 한다.

$$\forall t_i \in \mathrm{REMOTE}, \quad Trnarnd_i < Period_i \quad (6)$$

$$Trnarnd_i = Comnd_i + Up_i + Remote\left(WCET_i\right) + Dn_i \quad (7)$$

$$Up_i = \begin{cases} \frac{Size_i}{BW_{up}} & if\ first\ execution \\ \frac{Input_i}{BW_{up}} & otherwise \end{cases} \quad (8)$$

$$Dn_i = \frac{Output_i}{BW_{dn}} \quad (9)$$

$BW_{up}$과 $BW_{dn}$은 각각 업링크와 다운링크의 대역폭을 나타내며, $Remote\left(WCET_i\right)$는 원격 서버에서 실행시의 최악실행시간을 나타낸다. 상기의 태스크 모델은 코어가 하나인 시스템을 위한 기본 모델로 코어가 여럿인 시스템의 경우 부등식 (1), (2), (3), (5)의 우측 항을 1에서 코어의 수로 변경하여 모델링할 수 있으며, 이 경우 EDF가 아닌 다중코어용 P-Fair 부류의 알고리즘을 통해 스케줄링을 수행할 수 있다[12]. 그림 1은 본 논문의 태스크 실행 모델의 기본적인 구조를 보여주고 있다.

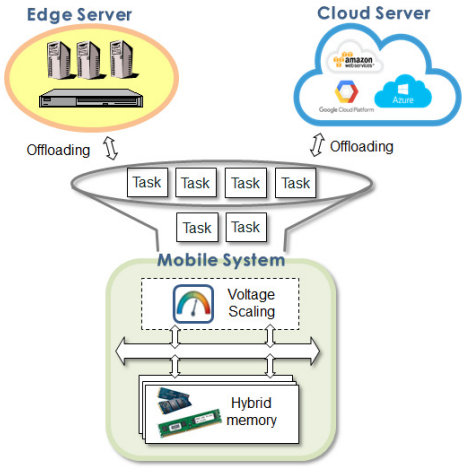



**그림 1. 제안하는 태스크 실행 모델의 기본 구조**
**Fig. 1. Basic structure of the proposed task execution model**

## III. 엣지와 클라우드 오프로딩의 분석

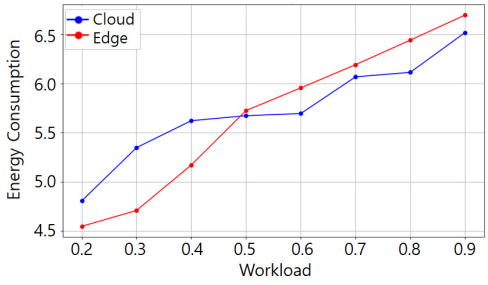
본 논문에서는 유전 알고리즘을 사용해서 실시간 태스크의 실행 위치, CPU의 클럭 빈도, 메모리 배치를 최적화한다. 최적화의 목표 함수는 모바일 시스템의 전력 소모 최소화이며, 제약 조건으로 태스크 집합의 모든 태스크에 대해 데드라인을 만족하도록 하였다. 태스크의 오프로딩은 클라우드 서버 또는 엣지 서버로 할 수 있게 하였고, CPU의 클럭 빈도는 {1, 0.5, 0.25, 0.125} 4가지로 정의하고, 메모리 위치는 {0, 1}로 표시해 DRAM 또는 NVM을 나타내도록 하였다.

유전 알고리즘의 세대를 구성하는 인구의 크기는 100으로 하였으며, 초기 세대는 랜덤하게 구성하였다. 유전 알고리즘에서 전체 인구 중 부모해를 고르는 선택 연산은 목표함수의 값이 우수한 해가 선택될 확률이 열등한 해가 선택될 확률보다 높게 하였으며, 구체적으로는 정규화를 통해 1위의 해가 선택될 확률이 100위의 해가 선택될 확률의 4배가 되도록 하였다. 교차 연산으로는 유전 알고리즘에서 가장 많이 사용하는 1점 교차 연산을 사용하여 임의로 얻어진 교차 지점의 좌우를 서로 다른 부모해로부터 이어받도록 하였다. 넓은 영역의 탐색을 위해 교차 연산 이후 해를 구성하는 특정값을 교란하는 변이 연산을 적용하였으며, 이렇게 얻어진 해를 새로운 세대에 삽입하고 기존 세대에서 가장 열등한 해를 제거하였다. 해집합이 수렴할 때까지 진화를 반복하였으며 각종 실험의 파라미터는 기존 연구의 방식을 따랐다[8].

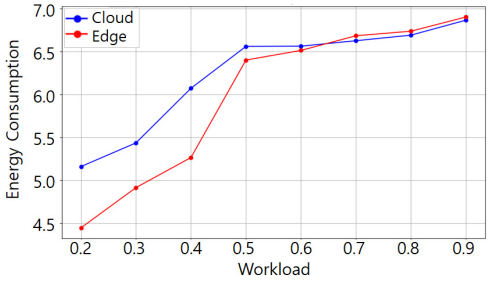
본 논문의 실험에서는 오프로딩이 엣지 서버로 이루어지는 경우와 클라우드 서버로 이루어지는 경우에 대해 태스크 집합의 부하가 변함에 따른 모바일 시스템의 에너지 소모 변화를 조사하였다. 태스크 집합의 부하는 로컬 CPU 이용률의 0.2에서 0.9에 이르는 범위에 대해 실험하였으며, 이때 이용률 1.0은 로컬 CPU에서 처리가능한 최대치를 뜻한다.

그림 2(a)는 클라우드 서버의 컴퓨팅 능력이 엣지 서버의 4배인 환경에서 두 오프로딩 기법을 사용한 경우의 에너지 소모량을 보여주고 있다. 이 실험에서 엣지 서버의 네트워크 대역폭은 기존 연구에서 사용한 80Mbps로 설정했고[8], 클라우드 서버의 대역폭은 50Mbps로 설정하였다. 그림에서 보는 것처럼 워크로드의 부하가 높은 상황에서는 클라우드 오프로딩의 에너지 절감효과가 더 컸으며, 이는 부하가 높은 상황에서 컴퓨팅 능력이 더 우수한 클라우드 서버의 효과를 입증한 것으로 볼 수 있다. 반면, 워크로드의 부하가 0.5보다 낮은 상황에서는 엣지 오프로딩의 에너지 절감효과가 더 큰 것을 확인할 수 있었다. 비록 엣지 서버가 클라우드 서버에 비해 컴퓨팅 능력은 떨어지지만, 부하가 낮은 상황에서는 네트워크 대역폭이 더 우수한 엣지 서버로의 오프로딩으로 충분한 결과를 얻을 수 있음을 의미한다.

그림 2(b)는 클라우드 서버의 컴퓨팅 능력이 엣지 서버의 2배인 환경에서 두 오프로딩의 결과를 보여주고 있다. 본 실험에서는 클라우드의 네트워크 대역폭을 40Mbps로 설정하였다. 그림에서 보는 것처럼 이러한 환경에서는 대부분의 경우 엣지 서버로의 오프로딩이 클라우드 오프로딩보다 에너지 절감효과가 큰 것을 확인할 수 있다. 워크로드의 부하가 0.7 이상으로 매우 높은 경우에는 클라우드 오프로딩이 더 좋은 결과를 나타내었으나 그 차이는 미미했다. 이는 클라우드의 컴퓨팅 능력이 엣지 서버에 비해 월등히 우수하지 않은 상황에서는 네트워크 대역폭이 우수한 근거리 엣지 서버로의 오프로딩이 모바일 실시간 시스템의 스케줄링에서 더 우수한 결과를 얻을 수 있음을 뜻한다.



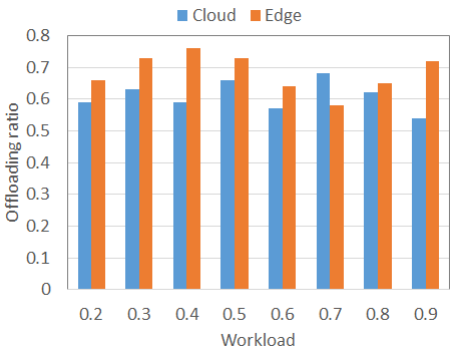

(a) 클라우드의 컴퓨팅 능력이 엣지 서버의 4배인 경우



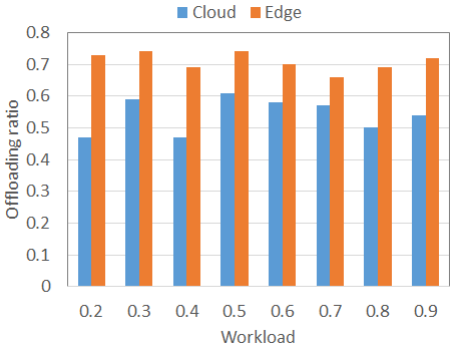

(b) 클라우드의 컴퓨팅 능력이 엣지 서버의 2배인 경우

**그림 2. 클라우드와 엣지 서버로 오프로딩 시의 모바일 시스템 에너지 소모 비교**

**Fig. 2. Comparison of mobile system energy consumption when offloading to cloud and edge servers**




(a) 클라우드의 컴퓨팅 능력이 엣지 서버의 4배인 경우




(b) 클라우드의 컴퓨팅 능력이 엣지 서버의 2배인 경우

**그림 3. 클라우드와 엣지 서버로 오프로딩 시의 모바일 시스템 에너지 소모 비교**

**Fig. 3. Comparison of mobile system energy consumption when offloading to cloud and edge servers**

그림 3(a)는 그림 2(a)의 실험 환경, 즉 클라우드 서버의 컴퓨팅 능력이 엣지 서버의 4배인 경우 클라우드 오프로딩과 엣지 오프로딩 적용 시의 오프로딩 비율을 비교해서 보여주고 있다. 그림에서 보는 것처럼 대부분의 경우 엣지 서버로의 오프로딩 비율이 좀 더 높았으나 클라우드 서버로의 오프로딩 비율이 더 높은 구간도 존재했으며, 두 경우 모두 오프로딩 비율이 공통적으로 높게 나타났다. 이는 모바일 디바이스의 컴퓨팅 파워에 비해 원격 서버의 컴퓨팅 파워가 충분히 크고 그 차이가 네트워크를 통한 태스크 오프로딩 비용을 상쇄하고도 남는다는 점을 시사한다.

그림 3(b)는 그림 2(b)의 실험 환경, 즉 클라우드 서버의 컴퓨팅 능력이 엣지 서버의 4배에서 2배로 줄어든 반면 클라우드로의 네트워크 대역폭은 더 낮아진 환경에서의 결과를 보여주고 있다. 그림에서 보는 것처럼 클라우드 서버로의 오프로딩 비율이 엣지 서버로의 오프로딩 비율에 비해 모든 경우에서 확연히 낮은 것을 확인할 수 있으며, 이러한 성향은 워크로드의 부하가 낮은 경우 더 뚜렷하게 나타났다. 이는 클라우드 서버의 컴퓨팅 능력이 엣지 서버의 2배에 불과한 반면, 네트워크 상황은 엣지 서버보다 나쁘기 때문에 오프로딩을 통한 이득이 엣지 서버에 비해 상대적으로 줄었기 때문으로 분석할 수 있다. 이러한 성향은 워크로드 부하가 낮은 상황에서 더욱 뚜렷이 나타났으며, 이는 클라우드로의 오프로딩이 효과적이기 위해서는 워크로드의 부하가 높고 네트워크 상황이 좋으며 컴퓨팅 능력이 상대적으로 더 우수해야 한다는 것을 의미한다.

## IV. 결 론

본 논문에서는 모바일 환경의 실시간 태스크를 클라우드 서버로 오프로딩하는 것과 엣지 서버로 오프로딩하는 것이 어떠한 차이를 나타내는지 광범위한 실험을 통해 분석하였다. 본 논문의 실험 결과 태스크의 부하가 높은 상황에서는 클라우드로의 오프로딩이 효율적인 반면, 태스크의 부하가 낮은 상황에서는 엣지 서버로의 오프로딩이 모바일 시스템의 에너지 절감 최적화에 더 효과적인 것으로 분석되었다. 또한, 클라우드 서버로부터 모바일 시스템까지의 네트워크 대역폭이 낮거나 클라우드의 연산 능력이 충분히 크지 않은 경우 클라우드보다는 엣지 서버로 오프로딩하는 것이 더 효과적인 것으로 분석되었다. 본 논문의 연구 결과를 토대로 클라우드와 엣지 서버의 다양한 환경적 특성 및 실시간 태스크 집합의 특성이 주어졌을 때 이에 적합한 태스크 오프로딩의 최적 설계를 수행할 수 있을 것으로 기대된다.

## References

[1] 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, no. 4, pp. 1-37, 2021.
DOI: https://doi.org/10.1145/3398209

[2] C. Park, "Analysis of optical characteristics of road surface conditions to select wavelengths for optical remote road sensor of autonomous vehicles," Journal of KIIT, vol. 22, no. 9, pp. 101-110, 2024.
DOI: https://doi.org/10.14801/jkiit.2024.22.9.101

[3] 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

[4] 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

[5] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3774-3785, 2021.
DOI: https://doi.org/10.1109/JIOT.2020.3024223

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

[8] 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

[9] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for IoV using 5G NR-V2X communication," IEEE Internet of Things Journal, vol. 9, no. 13, pp. 10397-10410, 2022.
DOI: https://doi.org/10.1109/JIOT.2021.3121796

[10] D. Lim and K. Sohn, "Task offloading scheme based on the DRL of connected home using MEC," The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), vol. 23, no. 6, pp. 61-67, 2023.
DOI: https://doi.org/10.7236/JIIBC.2023.23.6.61

[11] S. Park and H Bahn, "Combining genetic algorithms and bayesian neural networks for resource usage prediction in multi-tenant container environments," Cluster Computing, vol. 28, no. 2, article 111, 2025.
DOI: https://doi.org/10.1007/s10586-024-04832-6

[12] 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

## 저 자 소 개

**권 가 현(준회원)**

- 2020년 3월 ~ : 이화여자대학교 컴퓨터공학과 학부생
- 주관심분야 : 사물인터넷 시스템, 임베디드시스템, 인공지능

**반 효 경(정회원)**

- 1997년 2월 : 서울대학교 계산통계학과 학사
- 1999년 2월 : 서울대학교 전산과학과 석사
- 2002년 2월 : 서울대학교 컴퓨터공학부 박사.
- 2002년 9월 ~ : 이화여자대학교 컴퓨터공학과 교수.