Journal of the Korea Industrial Information Systems Research Vol.27 No.6, Dec. 2022 :69-75
http://dx.doi.org/10.9723/jksiis.2022.27.6.069 ISSN:1229-3741

비대칭 멀티코어 모바일 단말에서 SVM 기반 저전력 스케줄링 기법

(SVM-based Energy-Efficient scheduling on Heterogeneous Multi-Core Mobile Devices)

한 민 호[1], 고 영 배[2], 임 성 화[3]*
(Min-Ho Han, Young-Bae Ko, and Sung-Hwa Lim)

요 약 본 논문에서 비대칭 멀티 코어 구조의 스마트 모바일 단말에서 실시간성 보장과 에너지 소비량 절감을 고려한 작업 스케쥴링 기법을 제안한다. 최근 VR, AR, 3D 등 고성능 응용프로그램은 실시간과 고수준 작업이 요구된다. 스마트 단말은 배터리에 의존적이므로 높은 에너지 효율을 위해서 big.LITTLE 구조가 적용되었지만, 이를 제대로 활용하지 못함으로써 에너지 절감효과가 반감되는 문제점이 있었다. 본 논문에서는 big.LITTLE 구조의 단말에서 실시간성과 높은 에너지 효율을 높일 수 있는 비대칭 멀티코어 할당 기법을 제안한다. 이 기법은 SVM 모델을 활용해서 실제 작업의 실행시간을 예측하고 이를 통해서 에너지 소모와 실행시간을 최적화한 알고리즘을 제안한다. 상용 스마트폰에서의 비교실험을 통하여 제안기법이 기존 기법과 유사한 실행시간을 보장하면서 에너지 소비량의 절감을 보였다.

핵심주제어: 기계학습, 비대칭 멀티 코어, CPU 스케줄링, 에너지 절감

Abstract We propose energy-efficient scheduling considering real-time constraints and energy efficiency in smart mobile with heterogeneous multi-core structure. Recently, high-performance applications such as VR, AR, and 3D game require real-time and high-level processings. The big.LITTLE architecture is applied to smart mobiles devices for high performance and high energy efficiency. However, there is a problem that the energy saving effect is reduced because LITTLE cores are not properly utilized. This paper proposes a heterogeneous multi-core assignment technique that improves real-time performance and high energy efficiency with big.LITTLE architecture. Our proposed method optimizes the energy consumption and the execution time by predicting the actual task execution time using SVM (Support Vector Machine). Experiments on an off-the-shelf smartphone show that the proposed method reduces energy consumption while ensuring the similar execution time to legacy schemes.

Keywords: Machine learning, asymmetric multi-core architecture, CPU scheduling, energy saving

* Corresponding Author: sunghwa@nsu.ac.kr
Manuscript received May 26, 2022 / revised July 27, 2022 / accepted December 14, 2022

1) 아주대학교 정보통신연구소, 제1저자
2) 아주대학교 소프트웨어학과
3) 남서울대학교 멀티미디어학과, 교신저자

1. 서 론

최근 모바일 CPU의 트랜지스터 집적도의 발전 속도가 느려지면서, 코어의 수를 늘리는 멀티 코어인 이기종 컴퓨팅 방향으로 발전하고 있다. 그에 따라서 ARM에서는 이기종 다중 처리 시스템 구조 중 하나인 big.LITTLE 구조를 개발했다(ARM technologies, 2022). 이는 상대적

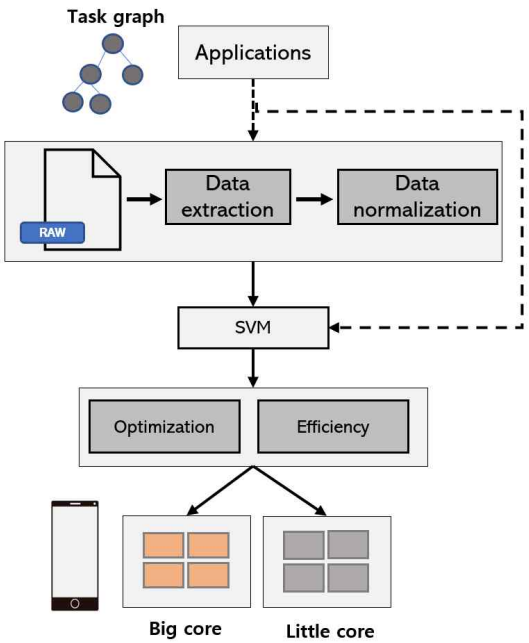

Fig. 1 Proposed model

으로 전력 소모가 적은 저성능인 LITTLE 코어와 전력 소모가 많은 고성능인 big 코어를 함께 탑재하는 구조이다. 최근 CPU는 멀티 코어 형태로 발전했으며, 멀티 코어 형태 중 하나인 big.LITTLE 구조는 엣지 디바이스 장치에 광범위하게 적용되었다. big.LITTLE 구조는 더 나은 성능과 높은 에너지 효율을 위해서 설계된 목적에 반하여, 대부분의 실시간 애플리케이션을 실행하는 동안 LITTLE 코어가 충분히 활용되지 않는다. 현재 애플리케이션은 만족할 만한 에너지 절감효과를 얻지 못하고 있다.

최근 big.LITTLE 구조의 이점을 활용하기 위한 다양한 기법이 제안되고 있다. 선행연구는 LITTLE 코어를 최대한 활용하여 실행시간과 에너지 소비량을 함께 줄이는 방향으로 연구되었다(Yeli Geng et al., 2018; Jie Ren, Lu Yuan et al., 2020; Wonik Seo, Daegil Im et al. 2015). 하지만 이 같은 방법은 작업 시간 단축으로 저부하 작업을 LITTLE 코어로 처리함에 한계가 있다(Donghoon Kim et al., 2020; Sumit K et al., 2019; Wonik Se et al., 2015). 또한, 현재 평균적으로 높아진 CPU 코어의 성능으로 개선된 실행시간 단축은 체감하기 어렵다(Comparison of Mobile Processors (CPU Benchmarks), 2022; healthline, 2020).

본 논문에서는 애플리케이션에서 요구되는 실행시간을 보장하면서 에너지 소비량을 낮추는 기술을 제안하기 위해서 SVM(Support vector machine)을 활용한 알고리즘을 제안한다. 제안하는 알고리즘은 작업 간에 상대적인 실행시간 개념을 활용해서 LITTLE 코어 사용을 최적화하는 방법이다. 이를 증명하기 위해서 기존 기법들과의 성능 비교를 통해서 제안하는 기법의 성능을 검증하였다.

2. 시스템 구성 및 실험

2.1 시스템 구성

본 시스템은 애플리케이션에서 Fig. 2와 같은 의존성이 있는 작업이 동시다발적으로 발생한다고 가정한다. 이 작업은 $T_{n,M}$으로 나타내며, n은 작업 고유 번호이고, M은 직전에 완료한 작업 고유 번호들의 집합이다. 즉, 고유 번호 n은 집합 M에 속하는 모든 작업이 완료된 이후에 실행할 수 있다. 제안하는 모델은 SVM 모델을 활용한다. SVM 모델을 학습시키기 위해서 결과 데이터를 학습 데이터로 재구축하며, k-means을 사용한다. 재구축된 학습 데이터로 SVM 모델을 학습한다. 학습한 SVM 모델의 결과를 활용해서 작업 최적화 및 효율화하기 위한 알고리즘을 제안한다. 제안하는 기법을 통해서 저전력 스케줄링을 진행한다.

2.2 학습 모델

본 논문에서 제안한 알고리즘에 활용된 기법은 SVM이다. 작업 분류를 위한 방법으로 SVM 모델을 적용했다.

SVM 모델의 실험은 선행연구 자료를 기반으로 진행했다(Jie Re et al., 2017; Robert P Dic et al., 1998). 결과 데이터를 학습 데이터로 변형하기 위해서 k-means기법을 활용했다. k-means 결과에 따라서 기존 5개의 그룹으로 나누어진 데이터를 3개의 그룹으로 분류했다. 학습 데이터로 SVM 모델을 학습했다. SVM 기법이 적용된 제안기법 모델은 Fig. 1과 같다.

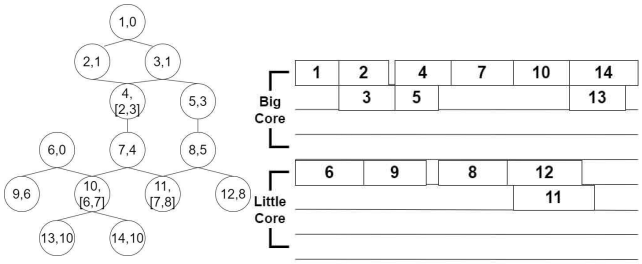

Fig. 2 An example of a task graph and its multicore schedule for an application

2.3 최적화 연산 모델

본 절에서는 $T_{n,M}$의 선행 작업이 2개 이상 있는 상황에서 선행 작업을 LITTLE 코어로 스위칭해도 $T_{n,M}$의 작업 시간에 변함없는 선행 작업을 선별한다. 이 선행 작업을 LITTLE 코어로 스위칭한다. Fig. 2에서 $T_{10,[6,7]}$의 선행작업인 $T_{6,0}, T_{7,4}$ 선행작업의 작업 시간을 비교한다. $T_{6,0}$은 LITTLE 코어로 스위칭한 결과, $T_{10,[6,7]}$의 작업 시작 시간에 변함이 없다. 이와 같은 방식으로 $T_{n,M}$을 LITTLE 코어로 스위칭하는 방안을 제안한다. $t_{n,M}$은 $T_{n,M}$의 big 코어에서 작업 소요 시간을 수치화한 개념으로 아래와 같이 표현할 수 있다.

$$t_{n,M} = \frac{E(G_n)}{E(G)} \times SVM_n \times \gamma \quad \cdots (1)$$

식 (1)에서 γ는 LITTLE 코어에서 처리할 경우, 코어의 성능 차이 값이다. $E(G)$는 총 작업의 평균작업 소요 시간이다. $E(G_n)$는 작업이 속하는 그룹의 평균작업 소요 시간이다. SVM_n은 예측되는 분류된 그룹 번호이다. $Score_{n,M}$는 $T_{n,M}$의 작업 완료 시간을 수치화한 개념이다. $T_{n,M}$의 $Score_{n,M}$는 아래와 같이 표현할 수 있다.

$$Score_{n,M} = \frac{E(G_n)}{E(G)} \times SVM_n \times \gamma + \max_{1 \le m \le |M|} Score_{m,K} \quad \cdots (2)$$

식(2)에서 $\max_{1 \le m \le |M|} Score_{m,K}$는 $T_{n,M}$ 이전 작업인 M 집합에 속한 작업 중에서 가장 값이 큰

ALGORITHM 1. PROPOSED ALGORITHM

Date: *Task graphs, Performance value of CPU core, Learning data*

Result: *Schedule sequence S*

Learning Initialization
  Data Extraction and normalization
  Creating k clusters denoted D_1, D_2, D_3, D_4
  For all learning a SVM model from D_1 do
Function Initialization
  For n to range ($T_{n,M}$):
    $Score_{n,M} = \frac{E(G_n)}{E(G)} \times SVM_n \times \gamma + \max_{1 \le m \le |M|} Score_{m,k}$
Function Optimization
  For n to range ($T_{n,M}$):
    If $1 + \frac{Score_{n',M} - Score_{n,M}}{t_{n,M}} \ge \gamma$:
      $T_{n,M}$ ← LITTLE core
    Elif $1 + \frac{Score_{n,M} - Score_{n',M}}{t_{n',M}} \ge \gamma$:
      $T_{n',M}$ ← LITTLE core
  Update S
Function Efficiency
  $BM = \max Score_{n,M}$
  For n to range ($T_{n,M}$, 0, −1):
    $T_{n,M}$ ← LITTLE core
    IF $Score_{n,M} > BM$:
      $T_{n,M}$ ← big core
      Break
    Break
  Update S

Fig. 3 Proposed algorithm

값이다. 식 (1), (2)을 활용해서 $T_{n,M}$을 LITTLE 코어로 스위칭해도 총 작업 시간이 변함없는 작업을 찾을 수 있다. 아래 식을 만족하는 식은 아래와 같이 표현할 수 있다.

$$1 + \frac{Score_{n',M} - Score_{n,M}}{t_{n,M}} \ge \gamma \cdots (3)$$

식 (3)을 만족하는 $T_{n,M}$은 LITTLE 코어로 스위칭을 한 경우에도 총 작업 시간에는 변함이 없다. 이를 통해서 본 절의 목적을 이루어 낼 수 있다.

2.4 효율화 연산 모델

본 절에서는 작업 완료 시간을 보장하는 상황에서 작업을 LITTLE 코어로 스위칭한다. Fig. 2와 같은 작업 그래프에서 작업 완료 시간에 기준점이 되는 작업 찾는다. 기준 작업의 $Score_{n,M}$값을 밴치마크로 정의한다. 밴치마크보다 일찍 끝나는 작업을 LITTLE 코어로 스위칭하는 방안을 제안한다. LITTLE 코어로 스위칭하는 작업의 $Score_{n,M}$값이 밴치마크값 보다 작다. 이를 위해서 밴치마크가 되는 작업을 찾는다. 이는 아래와 같이 표현할 수 있다.

$$BM = \max Score_{n,M} \quad \cdots (4)$$

밴치마크는 BM로 정의한다. 밴치마크 작업보다 일찍 끝나는 작업을 LITTLE 코어로 스위칭한다. LITTLE 코어로 스위칭한 $Score_{n,M}$값이 BM 보다 낮은 $T_{n,M}$을 찾는다. 이는 아래와 같이 표현할 수 있다.

$$TB \geq Score_{n,M} \cdots (5)$$

식(5)의 조건이 성립되는 상황에서 $T_{n,m}$을 LITTLE 코어로 스위칭함으로 본 절의 목적을 이루어 낼 수 있다.

3. 저전력 알고리즘

본 논문에서 제안하는 알고리즘은 Fig. 3과 같다. 주요 아이디어는 작업 완료 시간에 영향을 미치지 않는 작업을 LITTLE 코어로 스위칭하는 것이다. 이는 SVM 모델 결과를 활용한 알고리즘이며, 각 단계별 설명은 다음과 같다.

본 알고리즘에 필요한 데이터는 작업 그래프, big 코어와 LITTLE 코어의 성능, 학습 데이터 등이다. 제안기법의 각 단계에 대한 설명은 다음과 같다.

- Learning Initialization : SVM 모델을 학습시키기 위한 단계이다. 데이터를 지도학습에 맞는 형태로 변환하기 위해서 정규화, k-means을 적용하며, 이 학습 데이터를 통해서 SVM 모델을 학습한다.
- Function Initialization : 각 작업의 상대적인 작업 시간을 계산하여, $T_{n,M}$의 $Score_{n,M}$ 값을 계산한다.
- Function Optimization : $T_{n,M}$의 집합 M이 원소가 2개 이상 있는 경우, 선행 작업을 LITTLE 코어로 스위칭할 수 있는지 판단한다.
- Function Efficiency : 밴치마크를 기준으로 LITTLE 코어로 스위칭이 가능한 $T_{n,M}$ 을 판단한다.

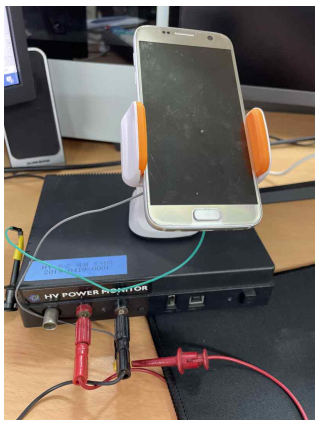

Fig. 4 Test-bed

4. 성능 평가

4.1 실험환경

본 실험은 선행연구 자료 기반으로 실험을 위한 테스트 프로그램을 TGFF(task graphs for free) 기반으로 구축했으며, 작업은 선형함수를 연속으로 실행시키는 방식으로 동작하여 CPU 성능측정에 널리 활용되고 있는 Linpack benchmark를 활용하였다 (Robert P Dick et

al., 1998; Jack J Dongarr et al., 2003). 실험 장비는 갤럭시 S7에서 진행했다(Jisang Park et al., 2017). 갤럭시 S7은 big 코어 4개와 LITTLE 코어 4개로 구성되어 있으며, LITTLE 코어는 1.6GHz으로 작동하며, big 코어는 2.3GHz으로 작동한다. 소비전력 측정은 Monsoon HV power monitor로 측정했다(Monsoon Solutions Inc., 2019). Fig. 4는 실험을 위한 테스트베드의 모습을 나타낸다. 측정값은 에너지 소비량, 작업 실행 시간을 기록했다.

실험을 위한 테스트 프로그램에서 24개의 의존성이 있는 작업을 랜덤하게 발생시켰다. 반복적인 실험으로 병렬성의 정도에 따라 크게 3가지 Case로 분류하였다. Case 1는 작업의 병렬성이 상대적으로 매우 낮게 설정되었다 ($T_{n,M}$ 에서 M의 원소가 1개 있는 경우). Case 2는 작업의 병렬성이 중간 수준으로 설정되었다 ($T_{n,M}$ 에서 M의 원소가 2개 이상 있는 경우). Case 3는 작업의 병렬성이 매우 높게 설정되었다 ($T_{n,M}$ 에서 M의 원소가 4개 이상 있는 경우). 이와 같은 3개의 Case에서 기존 기법들과 제안 기법의 성능을 비교 측정했다. 각 기법에 대한 설명은 다음과 같다.

- Default: 삼성 갤럭시 S7는 작업이 Big 코어에 우선 할당된다. 만약 모든 Big 코어를 사용중인 경우, Little 코어로 작업이 할당된다.
- EE Scheduling: 개별 작업의 목표 실행시간을 보장하며, 개별 작업의 목표 작업 완료 시간에 따라서 해당 작업을 LITTLE 코어로 스위칭한다 (Donghoon Ki et al., 2020).
- Proposing: 본 논문의 제안기법으로써 Fig. 3에서 제안한 알고리즘으로 작동한다.

4.2 결과분석

본 실험은 응용프로그램에서 발생한 에너지 소비량, 작업 완료 시간을 기록했다. Fig. 5는 각 Case별로 주어진 작업 그래프의 최종 완료

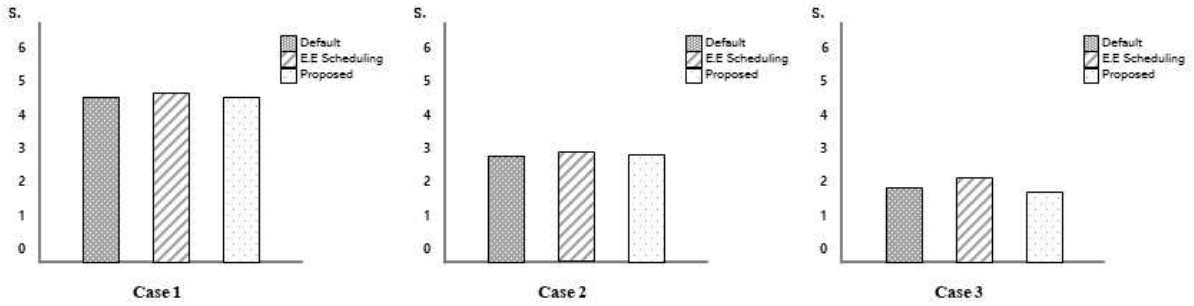

Fig. 5 Comparisons of job completion time for each cases

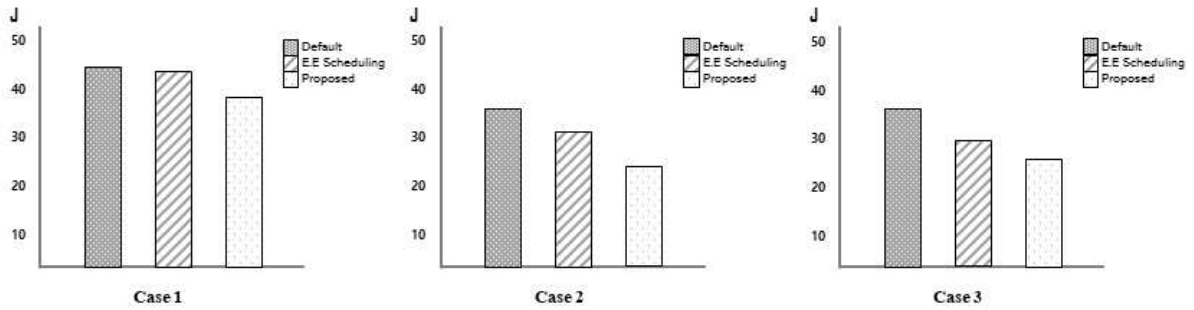

Fig. 6 Comparisons of energy consumption for each cases

시간을 나타내었으며, Fig. 6은 각 Case별로 주어진 에너지 소비량을 제안기법과 기존 기법(Default와 EE Scheduling)들 별로 나타내었다.

Fig. 5에서 알 수 있듯이, 모든 Case에서 작업 시간은 기존 기법들과 비슷한 수준의 작업 완료 시간을 보였지만, 약 1.1% 기존 기법보다 오래 걸렸다. 이는 스위칭 작업으로 인한 오버헤드로 확인했다. Fig. 6에서 나타낸 에너지 소비량은 모든 Case에서 제안기법이 감소함을 보였다. Case 1에서 제안기법의 에너지 소비량은 Default 대비 약 15%가 감소했으며, Case 2에서는 35%가 감소했으며, Case 3에서는 32%가 감소했다. 제안기법은 기존 기법 대비 코어를 최대 2개 더 사용했다. LITTLE 코어 사용으로 저전력을 통한 낮은 에너지 사용량 확인할 수 있었다. 모든 Case에서 Proposed는 Default와 비슷한 수준의 작업 완료 시간을 나타내었다. Case 1에서는 최적화 모델에서는 작업 스위칭이 발생하지 않았으며, 효율화 모델에서 LITTLE 코어로 스위칭이 이루어졌다. Case 2와 Case 3에서는 최적화 연산모델과 효율화 모델 모두에서 LITTLE 코어로 스위칭이 발생했다. 제안기법은 $T_{n,M}$ 집합 M의 원소가 2개 이상인 상황에서 높은 효율을 보였다. 이를 통해서 상대적인 작업 시간 개념으로 작업 완료 시간을 보장하면서 저전력 스케줄링이 가능한 것을 확인했다.

5. 결 론

본 논문에서는 비대칭 멀티 코어 모바일 장치에서 작업을 효율적으로 처리하는 저전력 스케줄링 기법을 제안하였다. 제안 알고리즘은 작업 완료 시간이 기존 방식과(Default) 유사한 상황에서 LITTLE 코어를 활용하는 기법이다. 제안 알고리즘으로 저전력 효과를 보였다. 상용 스마트폰에서 구현실험을 통하여 제안기법이 기존 기법들에 비해 비슷한 실행시간을 보장하면서 최대 35%의 에너지 절감 효과를 보였다. 특히, Case 2, 3과 같이 선행 작업이 2개 이상일수록 높은 효율을 확인할 수 있었다. 향후 모바일 기기의 발열량과 에너지 소비량의 관계를 확인하고 이를 고려한 스케줄링 기법을 연구할 계획이다.

ACKNOWLEDGMENT

본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행되었으며(No. 2021R1A2C1012776), 또한 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음(IITP-2022-2018-0-01431).

References

ARM technologies (2022). https://www.arm.com/technologies/big-LITTLE (accessed on May 1th, 2022)

Comparison of Mobile Processors (CPU Benchmarks) (2022). https:// www.notebookcheck.net/Mobile-Processors-Benchmark-List.2436.0.html (Accessed on Apr. 12th, 2022).

Donghoon Kim, Young-Bae Ko, Sung-Hwa Lim. (2020). Energy-Efficient Real-Time Multi-Core Assignment Scheme for Asymmetric Multi-Core Mobile Devices. 10.1109/ACCESS.2020.3005235

Healthline(2020), How Many Frames Per Second Can the Human Eye See?, https://www.healthline.com/health/human-eye-fps (accessed on May, 1th, 2022)

Jack J Dongarra, Piotr Luszczek, and Antoine Petitet. (2003) The linpack benchmark: past, present and future. doi.org/10.1002/cpe.728, Concurrency and Computation: practice and experience, 15(9):803 - 820, 2003.

Jie Ren, Lu Yuan, Petteri Nurmi, et al. (2020). Camel: Smart, Adaptive Energy Optimization

for Mobile Web Interactions,. IEEE INFOCOM 2020-IEEE Conference on Computer Communications, 202010.1109/INFOCOM41043.2020.9155489

Jie Ren, Ling Gao, Hai Wang, and Zheng Wang. (2017). Optimise Web Browsing on Heterogeneous Mobile Platforms: A Machine Learning Based Approach. 10.1109/INFOCOM41043.2020.9155489

Jisang Park, KIM Kiseong, Hongku Yeo, Jin-Hoo Lee, Jaewoong Chung, Duyeong Choi, and Minhyouk Lee. (2017) Mobile phone, US Patent App. 29/577,834.

Min-Seob Kim, Byoung-Hoon Lee, Jae-Hyuk Park, et al. (2020). Operando Identification of the Chemical and Structural Origin of Li-Ion Battery Aging at Near-Ambient Temperature. https://doi.org/101021/jacs.0c02203

Monsoon Solutions Inc.(2019) https://www.msoon.com(Accessed on Mar. 10th, 2022)

Robert P Dick, David L Rhodes, and Wayne Wolf. (1998). Tgff: task graphs for free. 10.1109/HSC.1998.666245

Sumit K. Mandal, Ganapati Bhat, Chetan Arvind Patil, et al.(2019). Dynamic Resource Management of Heterogeneous Mobile Platforms via Imitation Learning. 10.1109/TVLSI.2019.2926106

Se Won Lee, Donghoon Kim, Sung-Hwa Lim. (2020). Power-Efficient Big.LITTLE Core Assignment Scheme for Task Graph Based Real-Time Smartphone Applications, Mobile Internet Security. 10.1007/978-981-15-9609-4_6

Wonik Seo, Daegil Im, Jeongim Choi, and Jaehyuk Huh. (2015). Big or little: A study of mobile interactive applications on an asymmetric multi-core platform. 10.1109/IISWC.2015.7

Yeli Geng, Yi Yang, and Guohong Cao. (2018). Energy-efficient computation offloading for multicore-based mobile devices. 10.1109/INFOCOM.2018.8485875

한 민 호 (Min-Ho Han)

- 아주대학교 산업공학과 학사
- 아주대학교 AI융합네트워크학과 석사
- (현재) 아주대학교 정보통신연구소 연구원
- 관심분야: big.LITTLE, Location-based services, Machine learning

고 영 배 (Young-Bae Ko)

- 아주대학교 전자계산학과 공학사
- 아주대학교 경영정보학과 MBA
- Texas A&M University (College Station) 컴퓨터공학과 박사
- (현재) 아주대학교 소프트웨어학과 정교수
- 관심분야: Location-based services, Smart Mobility, Edge Intelligence

임 성 화 (Sung-Hwa Lim)

- 정회원
- 아주대학교 정보및컴퓨터공학부 공학사
- 아주대학교 정보통신공학과 석사
- 아주대학교 정보통신공학과 박사
- 남서울대학교 멀티미디어학과 부교수
- 관심분야: 저전력 컴퓨팅, 운영체제, 모바일 컴퓨팅, 실시간 시스템