Journal of the Korea Industrial Information Systems Research Vol.26 No.5, Oct. 2021 :11-19
http://dx.doi.org/10.9723/jksiis.2021.26.5.002 ISSN:1229-3741

모바일 디바이스를 위한 마일리지 기반 비대칭 멀티코어 스케줄링+

(Mileage-based Asymmetric Multi-core Scheduling for Mobile Devices)

이 세 원[1], 이 병 훈[2], 임 성 화[3]*
(Se Won Lee, Byoung-Hoon Lee, and Sung-Hwa Lim)

요 약 본 논문에서는 각 코어의 마일리지를 기반으로 하는 비대칭 멀티코어 프로세서의 스케줄링 기법을 제안한다. 저전력을 소비하며 일반성능을 갖는 LITTLE 코어와 고성능을 갖춘 대신 고전력을 소비하는 big 코어로 구성된 big-LITTLE 멀티코어 프로세서 구조를 고려하였다. 시스템에 태스크가 도착하여 처리해야 할 때, 프로세서는 태스크를 처리할 코어 유형(big 또는 LITTLE)을 먼저 결정한 다음 유휴 중인 코어들 가운데서 마일리지가 가장 작은 코어를 조사하여 해당 작업을 코어에 할당한다. 비대칭 멀티코어 할당을 위한 마일리지 기반 밸런싱 알고리즘을 개발하였으며 제안한 스케줄링 기법이 시스템 관리 관점에서 기존 방식보다 더 비용 효율적임을 보인다. 또한 시뮬레이션을 수행하여 제안한 알고리즘의 성능을 평가한다.

핵심주제어 : 멀티코어 스케줄링, 비대칭 멀티코어 구조, 마일리지 기반 스케줄링, 모바일 기기

Abstract In this paper, we proposed an asymmetric multi-core processor scheduling scheme which is based on the mileage of each core. We considered a big-LITTLE multi-core processor structure, which consists of low power consuming LITTLE cores with general performance and high power consuming big cores with high performance. If a task needs to be processed, the processor decides a core type (big or LITTLE) to handle the task, and then investigate the core with the shortest mileage among unoccupied cores. Then assigns the task to the core. We developed a mileage-based balancing algorithm for asymmetric multi-core assignment and showed that the proposed scheduling scheme is more cost-effective compared to the traditional scheme from a management perspective. Simulation is also conducted for the purpose of performance evaluation of our proposed algorithm.

Keywords : Multi-core scheduling, Asymmetric multi-core, Mileage-based scheduling, Mobile devices

* Corresponding Author : sunghwa@nsu.ac.kr
+ 이 논문은 부경대학교 자율창의학술연구비(2020년)에 의하여 연구되었음
Manuscript received September 24, 2021 / revised October 15, 2021 / accepted October 19, 2021
1) 부경대학교 경영학부, 제1저자
2) 수성대학교 ABC(AI, Blockchain & Big data, Cloud)학과
3) 남서울대학교 멀티미디어학과, 교신저자

1. 서 론

전자제품의 산업 성장을 이끌어 온 스마트폰, 태블릿 PC 등과 같은 스마트 모바일 단말은 뛰어난 성능과 높은 전력 효율성을 제공할 수 있는 비대칭 멀티코어 구조를 널리 탑재하는 추세이다. ARM에서 제공하는 big.LITTLE 코어 구조는 이러한 비대칭 멀티코어 구조에서 가장 널리 사용되는 제품 중 하나이다 (Greenhalgh, 2013).

스마트 모바일 단말은 1) 사용자와 인터넷과의 매개체 역할을 하고, 2) IoT(Internet of Things) 환경에서는 여러 무선 센서와 웨어러블 장치들에 대한 network sink 나 sensor sink의 역할도 감당하고 있으며, 3) 그 외에도 사용자에 대한 전자 개인 비서로서의 다양한 역할을 담당하며 실생활에서 그 중요성이 날마다 더해지고 있다.

이러한 스마트 모바일 단말이 정지되거나 기능에 이상이 생길 경우 서비스 전반에 걸친 연쇄적인 파급효과가 발생하여 서비스의 질과 사용자 만족도가 급감할 수 있다. 그러므로 스마트 모바일 단말의 가용성과 신뢰성은 매우 중요하게 지켜져야 할 요소 중 하나가 되었다 (Canalys, 2012).

프로세서는 스마트 모바일 단말에서 가장 중요한 하드웨어 모듈 중 하나이다. 이러한 프로세서의 결함은 크게 soft error와 hard error로 나뉜다 (Srinivasan et al., 2004). Soft error는 일반적으로 발열 또는 전자기적 노이즈에 의해 발생하며 일시적(transient)이다. 그러므로 soft error는 프로세서에 대한 하드웨어적 결함을 발생시키기보다는 계산상의 오류나 데이터 오류를 유발한다. 비대칭 멀티코어 구조에서 프로세서의 soft error 극복에 대한 연구는 그동안 많은 연구자들에 의해 이루어져 왔다 (Rosa et al. , 2017; Baldassari et al., 2017; Lim, 2017; Naithani et al., 2017; Naithani et al., 2018). Hard error는 하드웨어적 결함이나 노후화로 야기되는 영구적인 결함을 의미한다. Hard error는 extrinsic failure와 intrinsic failure로 분류할 수 있다. Extrinsic failure는 설계 또는 제작 결함으로 야기되며, 결함 발생률은 시간이 지날수록 줄어드는 특징을 갖는다. Intrinsic failure는 노후화로 야기되며 시간이 지날수록 결함 발생률이 증가한다. 본 논문에서는 프로세서의 hard error 중 intrinsic failure를 다룬다.

멀티코어의 코어 개수가 증가할수록 시스템 전체의 신뢰성은 급격히 감소하는 것으로 보고되고 있다 (Lu, 2005; Tang and Tan, 2016). 더구나 요즈음의 스마트 모바일 단말은 24/7로 중단 없이 동작하는 특성을 가지므로, 프로세서의 노후화가 시스템 결함에 끼치는 영향은 기존보다 매우 커지고 있다. 왜냐하면 멀티코어 중 한 개의 코어에만 결함이 발생해도 시스템 전체의 운영이 중단되거나 급격한 성능 저하가 발생하기 때문이다.

멀티코어 구조에서 각 코어는 각각 수명이 있는데, 이는 수행할 수 있는 명령(instructions)의 총 개수로 표현할 수 있다. 따라서, 동일한 운영시간 동안이라도 더 많은 명령을 수행한 프로세서가 다른 프로세서들에 비해 더 빨리 노후화가 진행된다.

전통적인 비대칭 멀티코어 스케줄링 기법에서는 수행할 태스크를 할당할 코어 유형(즉, big 또는 LITTLE)이 결정되면, 해당 타입의 코어들 중 유휴 상태인 코어 중 하나에 할당하게 된다. 이 경우 유휴 상태인 해당 타입의 코어가 두 개 이상 존재한다면, tie-break를 위한 정책이 필요하다. 이러한 tie-break가 발생하면 통상적으로 인덱스 번호가 낮은(또는 높은) 코어에 태스크를 할당하는 식의 정책이 사용된다. 그런데, 일반적으로 CPU의 사용률이 다른 장치들에 비해 낮기 때문에 멀티코어의 모든 코어가 동시에 사용되는 경우는 많지 않다. 그러므로, tie-break를 위하여 특정 코어가 먼저 사용되는 상황이 계속 반복될 가능성이 매우 높다. 이런 상황이 지속된다면 자주 사용되는 코어는 다른 코어들보다 노후화가 가속되어 intrinsic failure가 발생할 확률이 높아지게 된다.

Fig. 1은 위에서 설명한 멀티코어의 비대칭 할당 문제를 나타내고 있다. 그림의 예에서는 코어 유형이 결정되면 해당 코어 유형의 유휴 코어들 중 인덱스 번호가 낮은 코어에 우선 할당된다고 가정한다. 이 경우 프로세서 사용률이

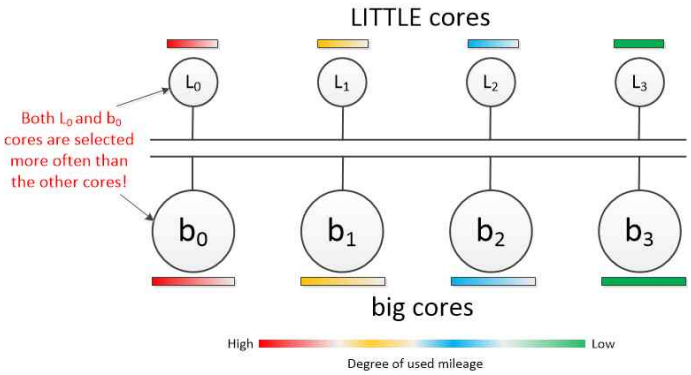

Fig. 1 An illustration of an unbalanced multi-core assignment problem (Lee et al. 2020)

높지 않다는 가정 하에서, L_0 또는 b_0의 사용률은 다른 코어들보다 월등히 높게 되며, 결국 노후화에 따른 intrinsic failure의 가능성이 다른 코어들보다 상대적으로 높아진다.

특정 코어의 노후화로 인한 결함 발생이 다른 코어들보다 사전에 허용된 값 이상 빠를 것으로 예상된다면, tie break 상황에서 해당 코어의 우선순위를 낮추어 스케줄링함으로써 코어의 intrinsic failure로 인한 시스템 결함을 줄일 수 있을 것이다.

이를 위해서는 코어마다 수행한 누적 명령 개수를 저장 및 관리할 필요가 있다. 단, 마일리지 저장 및 관리 작업이 시스템 처리 성능에 영향을 주어서는 안 된다. 이러한 마일리지 정보를 활용하여, tie-break 상황에서 스케줄링 대상이 되는 코어들 중 마일리지가 가장 낮은 코어를 우선적으로 할당하는 마일리지 기반 부하분배(load balancing) 할당기법에 대한 연구가 필요하다.

멀티코어 구조에서 hard error 중 intrinsic failure를 줄이려는 연구(즉, 프로세서의 수명 연장)는 다른 연구들에 비해 상대적으로 매우 적게 이루어져 왔다. Simevski et al.(2014)는 멀티코어의 수명을 높이기 위하여 가장 수명이 많이 남은 코어를 우선적으로 선택하는 YFRR (Youngest First Round Robin) 기법을 제안하였다. 제안 기법은 수명 측정을 위한 추가적인 하드웨어 장치가 탑재되어야 하며, 특히 일반 멀티코어 서버환경을 대상에서 라운드 로빈 방식을 기반으로 하였으므로, 비대칭 멀티코어 구조의 모바일 환경에서는 적합하지 않다.

비대칭 멀티코어 구조에서 태스크 스케쥴링에 대한 기존 연구는 대부분 성능향상 또는 전력소비량 절약에 초점이 맞춰져 있다. Bui et al.(2015)는 기존의 big.LITTLE 스케쥴링이 대부분의 태스크를 big 코어에 할당시키는 단점을 개선하기 위하여 애플리케이션 별로 수행할 태스크의 중요도에 따라 수행할 코어 타입을 지정하여 에너지 소모를 줄일 수 있는 AAS (Application Assisted Scheduling) 기법을 크롬 애플리케이션을 대상으로 제안하였다. Kim et al.(2020)은 하나의 애플리케이션 수행을 위하여 여러 개의 태스크가 의존성을 갖고 연결되는 점을 고려하여 태스크 그래프 기반의 big.LITTLE 코어 스케쥴링 기법을 제안하였고, 특히 실시간 환경에서 실시간 태스크의 마감시간을 보장하면서 에너지 소비를 줄일 수 있음을 보였다. 그러나 기존의 비대칭 멀티코어 환경에서 프로세서의 수명을 고려한 스케쥴링 기법에 대한 연구는 아직 이루어지지 않았다.

본 논문에서는 비대칭 멀티코어 구조의 스마트 모바일 단말에서 프로세서의 명령 수행 마일리지 정보를 기반으로 태스크를 할당함으로써 특정 코어의 급격한 노후화로 인한 시스템 결함을 방지하기 위한 부하분배 기반 태스크 할당 기법을 제안한다. 시뮬레이션을 통해 제안기법이 기존의 방식에 비해 시스템 결함 시간을 지

연할 수 있음을 보였다.

본 논문은 본 연구팀에서 국제학술대회에 발표한 Lee et al.(2020)의 확장판으로 다음의 추가 및 개선사항을 갖는다.

- 비대칭 멀티코어 구조에서 각 코어들의 노후화로 인한 결함발생시점을 와이블 분포를 이용한 확률적 모델링을 통하여 제시하였다.
- 마일리지 기반 밸런싱 기법을 개선하였고 시스템 성능의 영향을 최소화하는 지연된 방식의 마일리지 저장·관리 알고리즘을 추가하였다.
- 본 논문에서 제시하는 와이블분포 기반의 수학적 모델링을 활용하여 시뮬레이션 결과를 도출하였다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 가정하는 시스템 모델과 개발에 필요한 수학적 모델을 소개한다. 3장에서는 본 논문에서 제안하는 멀티코어의 마일리지를 활용한 부하분배 기반 태스크 할당 기법을 설명한다. 4장에서는 제안 기법의 검증을 위한 시뮬레이션 결과를 보이고 분석한다. 마지막으로 5장에서 결론을 맺는다.

2. 시스템 모델

2.1 주요 개념과 기본 가정

본 절에서는 분석의 기초가 되는 이론적 배경과 대상시스템에 대해 설명한다. 모바일 디바이스의 비대칭 멀티코어 구조는 다음과 같이 모델링이 가능하다.

모델링을 위한 주요 개념들을 먼저 정의하자. 본 연구에서는 코어 설계 시 제시된 수명을 보장수명이라고 하고, 이를 수행할 수 있는 명령의 개수로 표현한다. 보장수명이 다 되어도 즉시 해당 코어의 동작이 정지되는 것은 아니며, 이후 시간이 지남에 따라 동작 정지 확률이 급등하도록 변경된다. 그러므로 보장수명이 끝나는 시점으로부터 노후화로 인한 동작정지까지 수행한 명령의 개수를 추가수명으로 본다.

기본 가정들은 다음과 같다:

(1) 시스템은 고성능(big), 일반성능(LITTLE)의 코어를 종류별로 4개씩 가지고 있으며 각각의 코어들은 설계 시 제시된 수명(보장수명) 만큼의 생존이 보장된다.

(2) 각각의 코어들은 명령을 수행할수록 잔여수명이 줄어든다. 즉, 본 논문에서 잔여수명은 보장수명에서 누적 명령수행 개수이다.

(3) 각 코어들의 보장수명은 같으나 추가수명은 이산 와이블분포를 따른다.

(4) 시스템에 도착하는 작업(task)의 길이는 명령의 개수로 표현가능하며 도착과정은 발생률 λ의 포아송과정(Poisson process)을 따른다.

2.2 이산 와이블 분포의 난수 생성

본 연구에서는 각 코어의 고장시간(MTTF, mean to time failure) 모델링을 위해 Balkouch et al.(2014)의 이산 와이블 분포를 사용한다. 왜냐하면 코어의 수명을 수행 가능한 명령의 개수로 정의하였기 때문이고, 또 다른 이유는 시간에 따른 고장률이 일정한지, 증가 또는 감소하는지 정확히 알려져 있지 않더라도 와이블 분포가 강건하고 유연하게 다양한 고장률을 표현할 수 있기 때문이다. 와이블분포에 대한 자세한 내용은 Ross(2020)의 14장에서 찾아볼 수 있다.

이산 와이블 분포의 확률질량함수(probability mass function) $p(x)$와 누적확률분포(cumulative probability distribution) $F(x)$는 각각 다음과 같다.

$$p(x) = q^{x^\beta} - q^{(x+1)^\beta}, \quad (x = 0,1,2,\ldots) \qquad (1)$$

$$F(x) = \sum_{n=0}^{x} p(n) = 1 - q^{(1+x)^\beta} \qquad (2)$$

여기서, $0 < q < 1$, $\beta > 0$이다.

식 (2)의 누적확률분포와 폐구간 [0,1]사이의 균일분포(uniform distribution)와의 관계를 이용하면 이산 와이블 확률변수 X의 값들을 다음과 같이 구할 수 있다.

$$X = \lceil -1 + \left(\frac{\ln(1-U)}{\ln q} \right)^{\frac{1}{\beta}} \rceil \quad (3)$$

여기서, $U \sim Uniform(0,1)$, $\lceil x \rceil$ 는 x를 올림한 수이다.

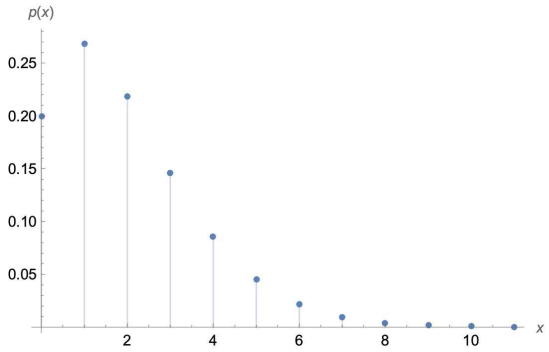

Fig. 2 Probability mass function of discrete Weibull distribution (for $q = 0.8$, $\beta = 1.5$)

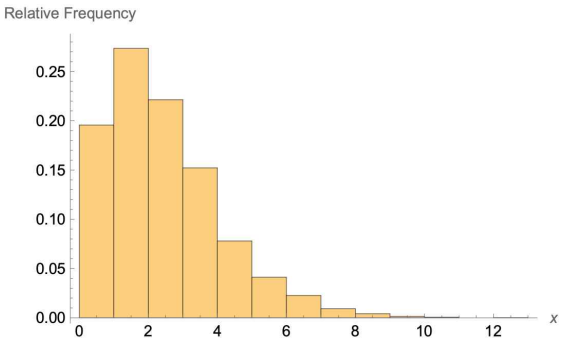

Fig. 3 Relative frequency histogram from 10,000 generated discrete Weibull random numbers

Fig. 2는 식 (1)에서 $q = 0.8$, $\beta = 1.5$인 이산 와이블 분포의 pmf를 나타낸 것이고, Fig. 3은 같은 파라미터 하에서 식 (3)으로부터 생성한 10,000개의 이산 와이블 확률변수의 값들로 상대도수 히스토그램을 그린 것이다. Fig. 2와 Fig. 3으로부터 식 (3)은 성공적으로 이산 와이블 확률변수들을 생성하고 있음을 확인할 수 있다.

이산 와이블 분포 난수생성에 대한 일부 연구에서는 올림 대신 내림을 사용하는데 이렇게 되면 와이블 확률변수 값의 비음조건을 만족하지 못하는 경우가 생긴다. 따라서 본 연구에서는 올림(ceiling)을 사용하여 비음조건을 충족시키고, 다양한 파라미터 하에서 이론함수와 난수발생을 통해 얻은 데이터들의 분포가 일치함을 확인하였다. 이산시간시스템의 분석과 시뮬레이션은 Lee(2020), Lee(2016) 등에서 찾아 볼 수 있다.

3. 마일리지 기반 밸런싱 기법

본 절에서는 비대칭 멀티 코어 프로세서 구조의 시스템 수명을 최대화하기 위해 마일리지 기반의 밸런싱 알고리즘을 제안한다(Algorithm 1). 제안한 알고리즘은 기존의 모든 비대칭 멀티코어 스케줄러에 적용할 수 있다. 알고리즘의 주요 절차와 필수 기능은 다음과 같다.

- 주요 절차: 들어오는 작업(task)을 수신하면 스케줄러는 먼저 *bigLITTLESelect*() 함수를 호출하여 작업에 대한 코어 유형(big 또는 LITTLE)을 결정한다. 코어 유형이 결정되면 스케줄러는 선택된 유형의 코어들 중 마일리지가 가장 작은 코어를 선택한다.

Algorithm 1: Mileage-based balancing algorithm for asymmetric multi-core assignment

```
1  Input: incoming Task T
2  repeat
3      coreType ← bigLITTLESelect(T)
4      S_avail ← getAvailableCoreList(coreType)
5  until S_avail is not empty
6  minIndex ← 0
7  min ← MAX_VAL
8  for k ← 0; k < length of S_avail; ++k do
9      if mileage of S_avail[k] < min then
10         min ← mileage of S_avail[k]
11         minIndex ← k
12     end
13 end
14 Assign task T to S_avail[minIndex]
15 mileage of S_avail[minIndex]
       ← recordMileage(T, S_avail[minIndex])
```

Fig. 4 Proposed Algorithm

- *bigLITTLESelect*(task *T*): 이 함수는 시스템 고유의 멀티코어 스케줄러를 활용하여 작업 *T*를 할당할 코어 유형의 값을 반환한다. 선택한 코어 유형이 big이면 0을, LITTLE이면 1을 반환한다.
- *getAvailableCoreList*(*coreType*): 이 함수는 두 개의 *coreType*(big 또는 LITTLE) 중 사용 가능한 코어 목록을 반환한다.
- *recordMileage*($task_i$, $core_{type\_num}$): 이 함수는 코어 $core_{type_num}$에서 $task_i$ 실행 시 수행된 명령의 개수를 반환한다. 단, 본 함수의 작동은 전체 시스템 성능에 영향을 주어서는 안 되므로, 실시간으로 진행하지 않고 프로세서의 유휴 시간에 실행된다+.

4. 성능평가

본 논문에서 제안한 비대칭 멀티코어의 마일리지 기반 부하 스케줄링 기법의 성능을 평가하였다. 시뮬레이션 툴은 MATLAB을 이용하였으며, 하나의 태스크에 20×10^6의 명령(instruction)이 존재한다고 가정하고, 각 코어의 보장수명은 10,000 태스크로 하여 2.1절의 이산 와이블 확률변수 길이를 고려한 코어의 결함발생시점(즉, 보장수명과 추가수명의 합)을 적용하였다. 전체 코어 중에 하나라도 결함발생시점에 도달하면 시스템 동작이 정지되는 것으로 간주하여 그 때까지 처리한 전체 태스크의 수로 성능을 측정하였다.

성능평가를 위하여 기존의 기법(Legacy)을 본 논문에서 제안하는 기법(Proposed)의 에너지 소비량과 전송 시간을 비교하였다. 각 기법에 대한 설명은 다음과 같다.

- Legacy: 안드로이드에서 스케줄러로 사용되는 CFS (Completely Fair Scheduler) (Love, 2010) 방식을 적용한다. big/LITTLE 코어 결정이 끝나면 Tie break을 위해 동일한 타입의 유휴 코어 중 인덱스 값이 작은 코어에 우선적으로 할당을 시킨다.
- Proposed: 본 논문에서 제안하는 알고리즘(Fig. 4)에 따라 코어 할당을 한다. Tie break을 위해 동일한 타입의 유휴 코어 중 마일리지 값이 가장 낮은 코어에 우선적으로 할당한다.

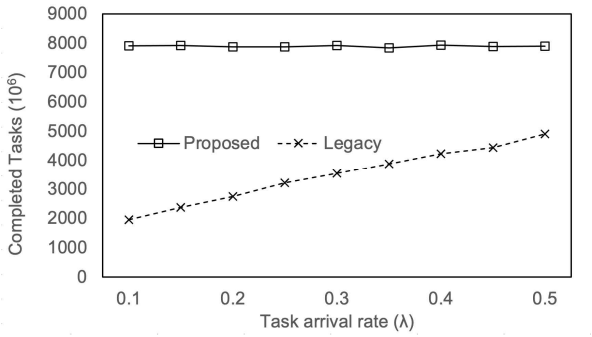

Fig. 5 Completed tasks for varying task arrival rate

Fig. 5는 big 코어 4개, LITTLE 코어 4개일 때, 식 (3)의 $q=0.8$, $\beta=2$로 설정하여 태스크의 도착률 λ에 따른 성능을 나타낸다. 제안 기법의경우 기존 기법(legacy)과 비교하였을 때 태스크를 각 코어에 최대한 균등하게 배정함으로써 발생 빈도가 높아져도 비교적 일정하게 우수한 성능을 보이고 있다. 반면 기존기법은 태스크의 도착률이 낮으면 특정 코어로 태스크 할당이 집중되어 그 코어의 수명이 빠르게 소진되므로 전체적인 성능이 떨어짐을 확인할 수 있다.

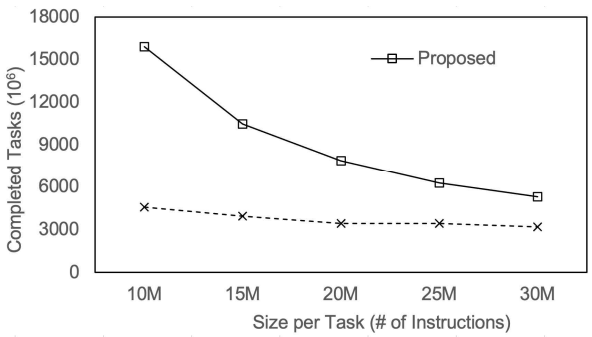

Fig. 6 Completed tasks for varying size per task

Fig. 6은 도착률 $\lambda=0.3$으로 고정한 상태에서

+ 마일리지 정보는 누적 수행 시간 또는 명령 갯수의 누적값으로 구성되는데, 마일리지 업데이트 시 메모리 접근 등으로 인한 시간 지연을 초래할 수 있으므로, 업데이트 작업은 실행 완료 후 유휴 상태에서 비동기 및 비실시간적으로 수행한다.

Fig. 5와 동일한 조건 하에서 태스크의 크기에 따른 성능을 측정하였다. 태스크의 크기가 클수록 하나의 태스크를 처리하는 데 수행할 명령의 개수가 많아짐을 의미한다. 제안 기법이 전체적으로 우수하지만 특히 태스크의 크기가 작을수록 성능이 더욱 좋음을 알 수 있다. 태스크의 크기가 크면 클수록 모든 코어가 동작 중일 가능성이 높아져 제안 기법과 기존 기법과의 차이가 작아지기 때문이다.

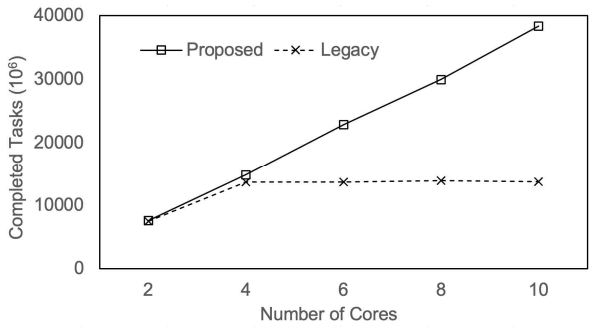

Fig. 7 Completed tasks for varying number of cores

Fig. 7은 Fig. 6과 같은 조건 하에서 코어의 개수에 따른 성능을 보여준다. 제안 기법은 코어 마다 태스크를 최대한 균일하게 할당함으로써 코어 수에 비례하여 성능이 증가함을 알 수 있다. 반면에 기존 기법은 특정 코어에 집중적으로 태스크가 할당되므로 해당 코어가 빠르게 결함발생시점에 도달해 다른 여분의 코어의 수명이 많이 남아 있음에도 불구하고 시스템이 정지됨으로써 코어 수가 많아져도 성능이 더 이상 좋아지지 않음을 확인할 수 있다. 단, 코어 개수가 2-4인 구간에서는 기존 기법도 성능 향상을 보여주는데, 이는 코어 개수가 작으면 모든 코어가 동시에 사용될 경우가 많아지므로 강제적으로 부하분배 효과가 발생하기 때문이다.

5. 결론 및 추후 연구과제

산업정보화 측면에서 스마트 모바일 단말의 역할과 그 중요성은 점점 더 증가하고 있다. 이러한 디바이스들은 하드웨어 측면에서 눈부신 발전을 거듭해 오고 있으며 효율적인 운영을 위한 소프트웨어적 접근 또한 다양하게 이루어지고 있다.

본 논문에서는 비대칭 멀티코어 구조의 스마트 모바일 단말에서 특정 코어의 균등하지 않은 결함 발생으로 전체 시스템의 성능과 가용도가 저하되는 문제를 다루었다. 프로세서의 명령 수행 마일리지 정보를 기반으로 태스크를 할당함으로써 특정 코어의 급격한 노후화로 인한 시스템 결함을 방지하기 위해 부하 분배(load balancing) 기반의 태스크 할당 기법을 제안하였다. 또한, 시뮬레이션을 통해 제안 기법이 기존의 멀티코어 할당기법에 비해 시스템 결함 발생시점을 늦출 수 있음을 보였다.

추후 연구로서 실제 테스트베드에 적용하여 제안 기법의 실용성을 검증할 예정이다.

References

Baldassari, A., Bolchini, C. and Miele, A. (2017). “A dynamic reliability management framework for heterogeneous multicore systems,” *2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Cambridge, 1-6.

Balkouch, H.S., Jazi, M. A. and Nadarajah, S. (2014). “A new discrete distribution,” *Statistics*, 48(1), 200-240.

Bui, D. H., Liu, Y., Kim, H., Shin, I. and Zhao, F. (2015). “Rethinking energy performance trade-off in mobile Web page loading,” in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 14-26.

Canalys (2012). “Smart phones overtake client PCs in 2011,” Canalys, Singapore, Tech. Rep. 2012/021.

Greenhalgh, P. (2013). “big.LITTLE technology: The future of mobile,” ARM Limited

White Paper.

Kim, D.-H., Ko, Y.-B. and Lim, S.-H. (2020). "Energy-Efficient Real-time Multi-core Assignment Scheme for Asymmetric Multi-core Mobile Devices," *IEEE Access*, 8, 117324-117334.

Lee, S.W. (2016). "Workload Analysis of Discrete-Time BMAP/G/1 queue under D-policy," *Journal of the Korea Industrial Information Systems Research*, 21(6), 1-12.

Lee, S.W. (2020). "Analysis of Discrete-Time Geo/G/1 Queues under Workload Control and Multiple Vacations," *Journal of the Korea Industrial Information Systems Research*, 25(1), 89-99.

Lee, S.W., Lee, B.-H. and Lim, S.-H. (2020). "Mileage-based Asymmetric Muli-core Scheduling," *2020 International Workshop on Smart Info-Media Systems in Asia (SISA 2020)*, Seoul, Korea, 161-163.

Lim, S.-H. (2017). "Dynamic Dependability Level Switching Strategies by Utilizing Threat Predictions," *Journal of the Korea Industrial Information Systems Research*, 22(2), 15-25.

Love, R. (2010). *Linux Kernel Development*, London, U.K.: Pearson.

Lu, Charng-Da (2005). "*Scalable diskless checkpointing for large parallel systems*," Ph.D. Thesis, University of Illinois at Urbana-Champaign.

Naithani, A. and Eyerman, S. and Eeckhout, L. (2017). "Reliability-Aware Scheduling on Heterogeneous Multicore Processors," *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Austin, Texas, USA, 397-408.

Naithani, A., Eyerman, S. and Eeckhout, L. (2018). "Optimizing Soft Error Reliability Through Scheduling on Heterogeneous Multicore Processors," *IEEE Transactions on Computers*, 67(6), 830-846.

Rosa, F., Ost, L., Reis, R., Davidmann, S. and Lapides, L. (2017). "Evaluation of multicore systems soft error reliability using virtual platforms," *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, Strasbourg, France, 85-88.

Ross, M. S. (2004). *Introduction to Probability and Statistics for Engineers and Scientists(3rd ed.)*, San Diego, USA, Elsevier.

Simevski, A., Kraemer R. and Krstic, M. (2014). "Increasing multiprocessor lifetime by Youngest-First Round-Robin core gating patterns," *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Leicester, UK, 233-239.

Srinivasan, J., Adve, S. V., Bose P. and Rivers, J. A. (2004). "The Case for Lifetime Reliability-Aware Microprocessors," *2004 International Symposium on Computer Architecture (ISCA'04)*, Munchen, Germany, 276-287.

Tang, X. and Tan, W. (2016). "Energy-Efficient Reliability-Aware Scheduling Algorithm on Heterogeneous Systems," *Scientific Programming*, 2016, 1-13.

이 세 원 (Se Won Lee)

- 종신회원
- 성균관대학교 산업공학과 학사
- 성균관대학교 산업공학과 석사
- 성균관대학교 산업공학과 박사
- 부경대학교 경영대학 경영학부 부교수
- 관심분야: 대기행렬이론, 확률모형, 시스템 최적화

이 병 훈 (Byoung-Hoon Lee)

- 충북대학교 컴퓨터공학과 학사
- 충북대학교 컴퓨터공학과 석사
- 아주대학교 정보통신공학과 박사
- 수성대학교 공학계열 ABC과 조교수
- 관심분야: 머신러닝, 음성인식, 사물인터넷

임 성 화 (Sung-Hwa Lim)

- 정회원
- 아주대학교 정보및컴퓨터공학부 공학사
- 아주대학교 정보통신공학과 석사
- 아주대학교 정보통신공학과 박사
- 남서울대학교 공과대학 멀티 미디어학과 부교수
- 관심분야: 모바일 컴퓨팅, 결함허용 시스템, 임베디드 시스템, 분산 시스템