논문 22-47-04-12 The Journal of Korean Institute of Communications and Information Sciences '22-04 Vol.47 No.04
https://doi.org/10.7840/kics.2022.47.4.663

루프 언롤링을 이용한 인터미턴트 컴퓨팅 성능 개선

김 시 현◆, 하 란°

Improving the Performance of Intermittent Computing Using Loop Unrolling

Sihyun Kim◆, Rhan Ha°

요 약

인터미턴트 컴퓨팅은 불안정한 전력 공급 상황에서 소프트웨어가 동작할 수 있는 컴퓨팅 기술이다. 특히 태스크 기반의 인터미턴트 컴퓨팅 기술과 loop continuation 기법은 데이터 비일관성 문제, 비종료 문제를 해결하여 인터미턴트 프로그램이 오류 없이 완료될 수 있게 하였다. 그러나 loop continuation 기법은 루프 변수를 FRAM에 기록하는 작업으로 인하여 에너지를 낭비하게 된다. 본 논문에서는 이러한 오버헤드를 줄이기 위하여 루프 언롤링을 적용한 인터미턴트 컴퓨팅 방법을 제안한다. 루프 언롤링을 적용하여 루프 반복 횟수를 줄여서 FRAM 기록 작업을 줄일 수 있었고 에너지 사용량을 개선할 수 있다. 키워드 검색 프로그램에 대하여 루프 언롤링 적용 모델 재부팅 횟수가 베이스라인 모델보다 5.44% 적은 실험 결과를 얻었으며 루프 언롤링 방법이 인터미턴트 컴퓨팅 성능을 개선했음을 확인하였다.

Key Words : Intermittent computing, Batteryless system, Keyword spotting, Energy harvesting, Loop unrolling

ABSTRACT

Intermittent computing is a technology that allows software to operate in unstable power supply conditions. In particular, task-based intermittent computing technology and loop continuation method solve data inconsistency and non-termination problems so that the intermittent program can be completed without errors. However, the loop continuation mechanism wastes energy due to the operation of writing loop variables to FRAM. In this paper, we propose an intermittent computing method with loop unrolling to reduce such overheads. By applying Loop unrolling, FRAM write operation can be alleviated by reducing the number of loop iterations, and thus energy usage can be improved. By experiments, the number of reboots of the loop unrolling applied model was 5.44% less than that of the baseline model for the keyword spotting program, and it was shown that the loop urolling method improved the intermittent computing performance.

Ⅰ. 서 론

현대 사회는 사물인터넷, 인공지능, 빅데이터, 모바일 네트워크 등의 기술 발전으로 인하여 초연결사회로 진입하게 되었다. 초연결사회란 사람, 사물, 공간 등 모든 것들이 인터넷으로 서로 연결되어 모든 것에 대한 정보가 생성 수집되고 공유 활용되는 사회를 뜻한다. 초연결사회에 부합하는 서비스를 사용자가 원활

◆ First Author : Hongik University Department of Computer Engineering, sihyun0626@naver.com, 학생회원
° Corresponding Author : Hongik University Department of Computer Engineering, rhanha@hongik.ac.kr, 정회원
논문번호 : 202202-017-C-RE, Received January 28, 2022; Revised February 23, 2022; Accepted February 23, 2022

하게 이용하기 위해서 수많은 사물인터넷 관련 기술이 요구된다. 이 때 배터리로 동작하는 사물인터넷의 경우 지속적인 배터리 관리 이슈가 있기 때문에 배터리 없이 빛과 같은 주변의 에너지만을 이용하여 동작하는 배터리리스(batteryless) 기기가 제안되었다. 배터리리스 기기는 주변 에너지를 하베스팅(harvesting)하여 저장한 에너지의 양이 특정 에너지 수준이 되었을 때 동작하고 꺼지는 것을 반복하며 동작한다. 이러한 배터리리스 기기에서 실행 맥락 등을 유지하기 위해서 간헐적인 에너지 공급 환경에서 동작할 수 있는 인터미턴트 컴퓨팅(Intermittent Computing) 기술이 관심을 받고 있다[1-3].

이런 흐름에 맞추어 몇몇 선행 연구는 여러 가지 인터미턴트 컴퓨팅 모델을 제시한다. 그 중 ALPACA(Adaptive Lightweight Programming Abstraction for Consistency and Atomicity) 모델[2]은 인터미턴트 프로그램을 태스크(task) 단위로 수행하게 하여 각 태스크 간의 원자성을 보장한다. ALPACA의 이런 태스크 기반 방법은 인터미턴트 프로그램에서 주로 나타나던 데이터 비일관성(data inconsistency) 문제를 해결한다[2]. 하지만 ALPACA 모델은 태스크 크기에 따라서 프로그램이 비종료 되는 문제가 있다.

SONIC(Software Only Neural Intermittent Computing) 모델[3]은 loop continuation 기법을 사용하여 ALPACA가 많은 반복문을 실행할 때 생기는 비종료 문제를 해결한다. SONIC의 loop continuation 기법으로 인하여 프로그램은 원자성과 관계없는, 유연한 태스크로 구성된다. 이렇게 유연해진 태스크는 프로그램이 태스크 크기와 상관없이 작업을 수행하고 무사히 종료할 수 있게 한다. 이런 SONIC 모델의 장점은 인터미턴트 컴퓨팅 환경에서 효과적인 딥러닝 추론 연산을 가능하게 한다.

기존의 인터미턴트 컴퓨팅 연구는 데이터 비일관성 문제, 비종료 문제를 해결하여 인터미턴트 프로그램을 오류 없이 완료할 수 있게 하였다. 하지만 인터미턴트 딥러닝 모델에서의 추론 연산은 비용이 큰 루프 연산이 지배적이다. 이러한 루프 연산은 잦은 분기 명령어와 조건식, 증감식 연산으로 인하여 인터미턴트 컴퓨팅의 성능을 저하시키는 원인이 된다. 이와 같은 문제를 해결하기 위해서는 루프 연산을 최적화하여 딥러닝 추론 연산을 개선할 필요가 있다. 따라서 본 논문에서는 루프 언롤링을 적용하여 인터미턴트 컴퓨팅에서의 루프 연산을 개선하는 방법을 제안한다.

그림 1의 (a)는 loop continuation을 적용하여 복소수의 크기를 계산하는 루프 코드이며 (b)는 (a)에 대하여 언롤링 인자가 2인 루프 언롤링을 적용한 코드이다. 그림 1의 (a), (b)는 같은 연산을 수행한다. (b)의 루프 반복 횟수는 (a)의 루프 반복 횟수의 절반 정도 크기이고 이러한 특징 때문에 (b)는 (a) 보다 루프 변수를 제어하는 연산량이 줄어드는 효과가 생긴다. 이처럼 루프 변수 제어 연산량을 줄이는 것은 SONIC 기반의 인터미턴트 컴퓨팅의 성능을 개선하는데 중요한 역할을 한다. SONIC의 loop continuation 기법은 그림 1과 같이 루프 변수를 MCU(MicroController Unit)가 직접 접근할 수 있는 비휘발성 메모리(non-volatile memory)인 FRAM(Ferroelectric Random Access Memory) 영역에서 사용하고 이는 잦은 FRAM 연산을 유발한다. FRAM에 접근할 때 발생하는 에너지 소모량은 SRAM보다 크기 때문에 SONIC 모델에서 루프 변수 제어 연산으로 생기는 오버헤드는 전체 시스템 에너지의 14%를 차지한다[3]. 이러한 이유로 루프 언롤링을 통하여 루프의 반복 횟수가 감소되면 loop continuation으로 인한 오버헤드를 개선하여 SONIC 기반의 인터미턴트 프로그램의 성능을 향상시킬 수 있다.

```
// v : loop value
FRAM v ← 0
for i ← v to N do
    m_i ← sqrt(im_i^2 + re_i^2)
    i ← ++v
end for
```

(a) Normal loop code

```
// v : loop value
FRAM v ← 0
for i ← v to N do
m_i ← sqrt(im_i^2 + re_i^2)
m_{i+1} ← sqrt(im_{i+1}^2 + re_{i+1}^2)
 i ← (v ← v + 2)
end for
```

(b) Loop unrolled code

그림 1. 루프 언롤링 예시
Fig. 1. Example of loop unrolling

본 논문에서는 루프 언롤링을 사용하여 루프 연산을 최적화한 LU-SONIC(Loop Unrolled-Software Only Neural Intermittent Computing) 모델을 제안한다. 성능 비교를 위하여 LU-SONIC 모델을 기반으로 간헐적인 에너지 환경에서 동작할 수 있는 키워드 검색(keyword spotting) 시스템을 구축하고 베이스라인 모델로 SONIC 모델 기반 키워드 검색 시스템 또한 구축한다. LU-SONIC 모델과 SONIC 모델에 관해서 각각 소비하는 에너지, 프로그램 수행시간 등의 성능에 대한 비교/분석 실험한다.

본 논문의 구성은 다음과 같다. Ⅰ의 서론에 이어서 Ⅱ에서 본 논문과 관련된 선행 연구를 살펴보고 Ⅲ에서 인터미턴트 컴퓨팅 환경에서 키워드 검색 시스템

을 모델링한다. Ⅳ에서 실험 결과에 대한 분석을 수행하고 Ⅴ에서 본 연구의 결론을 도출한다.

Ⅱ. 관련 연구

사물인터넷 기술의 수요가 증가하면서 배터리 관리 문제가 주요 연구 대상이 되었다. 무겁고 주기적으로 교체해야 하는 등 전지가 가지고 있는 여러 가지 단점들은 사물인터넷 기술의 지속가능한 구현에 대한 장애물이 되었다. 에너지 하베스팅 기술은 전지를 사용하지 않고 태양 에너지 등 외부 에너지원을 이용해서 에너지를 얻을 수 있는 기술이며 이러한 특징이 기존 전지의 문제점을 해결한다. 하지만 에너지 하베스팅으로 얻은 전력은 항상 사용할 수 있을 만큼 충분하지 않고 전력이 간헐적으로 공급되는 특성이 있으므로 정전이 빈번하게 발생한다. 이렇게 불안정한 에너지 환경에서 프로그램이 동작할 수 있도록 하기 위해서 인터미턴트 컴퓨팅 기술에 관한 연구가 활발하게 이루어졌다.

Mementos 기술[1]은 체크포인트 메커니즘을 사용해서 RFID로 에너지를 수확하는 하드웨어가 장시간의 연산을 수행 가능하게 한다. Mementos 모델은 사용가능한 에너지를 측정하여 정전이 임박한 상황이 발생하면 프로그램 위치를 체크포인트로 지정한다. 정전이 발생한 후 프로그램이 재시작되는 상황이 되면 재시작 위치를 사전에 지정한 체크포인트 위치로 프로그램의 제어 흐름을 변경해서 프로그램이 정전 이전의 진행 상태를 반영할 수 있게 한다.

Mementos 와 같은 체크포인트 기반의 인터미턴트 컴퓨팅은 WAR(Write After Read) 의존성을 가진 데이터에 대해서 데이터 비일관성 문제가 발생하는 문제가 있다. ALPACA 모델[2]은 태스크 기반의 인터미턴트 컴퓨팅 기술을 제안해서 데이터 비일관성 문제를 해결한다. 프로그램을 개발자가 정한 태스크 단위의 프로그램으로 변환하여 태스크에 해당하는 프로그램 단위로 원자성을 보장하여 데이터 비일관성을 제거한다.

SONIC 모델[3]은 ALPACA 모델을 이용하여 딥러닝 모델에 적합한 인터미턴트 컴퓨팅 모델을 제안한다. ALPACA 모델은 적절한 태스크 크기를 정하는 것이 어렵다는 문제점이 있다. 태스크의 크기가 작다면 태스크의 전환이 빈번해서 그로 인한 오버헤드가 증가하는 문제가 있다. 반면에 태스크의 크기가 크다면 프로그램은 계속 해당 태스크만 반복하는 비종료 문제를 야기할 수 있다. 왜냐하면 에너지 버퍼가 수용할 수 없는 과도한 에너지를 태스크가 요구한다면 태스크를 완료할 수 없고 결국 다음 태스크로 진행될 수 없기 때문이다. 이러한 문제점은 ALPACA 모델이 많은 루프 연산으로 이루어진 딥러닝 모델에 대하여 적절한 대응을 어렵게 했다. SONIC은 loop continuation 기법을 사용하여 루프 변수를 비휘발성 변수로 사용한다. 따라서 태스크가 재시작될 때 정전 이전의 루프 수행 상황을 반영한다. 이는 태스크의 원자성을 보장하지 못하지만 멱등성을 보장하여 데이터 일관성을 유지한다. Loop continuation으로 인하여 SONIC 모델은 합성곱(convolution) 연산, 완전 연결 계층(fully-connected layer)에서의 연산 같은 반복 연산들의 계산 비용을 줄일 수 있지만 여전히 루프 연산으로 인한 성능 저하 문제는 해결해야할 과제이다.

Ⅲ. 인터미턴트 환경의 키워드 검색 모델링

키워드 검색은 특정 단어 혹은 짧은 문장을 인식하는데 사용하는 음성 인식 기술이다. 키워드 검색 프로그램은 그림 2와 같이 크게 프로그램 초기 설정, 음성 데이터 처리, 딥러닝 추론 연산과정으로 이루어져 있다. 해당 프로그램은 하베스팅으로 얻은 에너지를 사용하기 때문에 에너지 공급이 간헐적이고 정전이 빈번하게 발생한다. 따라서 수행 중인 프로그램의 데이터가 정전으로 인하여 손실되는 것을 막기 위해서 비휘발성 메모리를 사용하여 연산 중간의 결과를 저장한다. 또한 딥러닝 모델에서 각 계층마다 사용되는 가중치와 편향치, 그 외의 데이터로 인해서 많은 양의 비휘발성 메모리가 요구된다. 하지만 MCU의 FRAM 용량 제약 조건으로 인해서 비휘발성 메모리 사용량은 제한된다. 따라서 매개변수 개수를 줄일 수 있는 딥러닝 모델을 채택하여 FRAM 기록 횟수를 절약해야 한다.

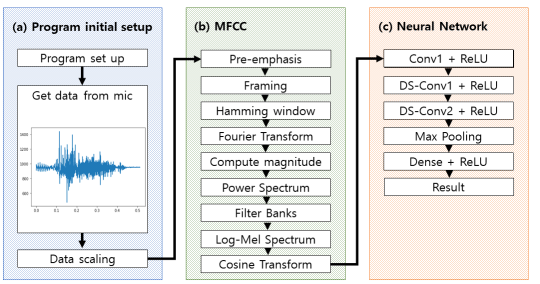


그림 2. 키워드 검색 시스템 모델링 프레임워크
Fig. 2. Framework of Keyword spotting system modeling

마이크에서 얻은 음성 데이터를 처리하는 과정에서는 데이터를 프레임 단위로 잘라서 이산 푸리에 변환, 이산 코사인 변환 등의 계산이 필요하다. 또한 딥러닝 추론 연산은 합성곱 연산, pooling 연산, 완전연결 계층 연산 등의 연산 과정을 수행한다. 이렇게 키워드 검색 프로그램은 대부분의 코드가 루프 연산으로 구성되어 있다. 빈번한 루프 연산은 반복적인 조건식 테스트 연산, 증감 연산, 분기 명령어로 인하여 프로그램의 수행시간이 증가하고 따라서 인터미턴트 프로그램의 정전 빈도가 늘어나는 문제를 일으킨다. 또한 루프 연산에 사용되는 loop continuation은 루프 변수를 FRAM에 기록하는 작업으로 인해 에너지를 낭비하게 한다. 루프 언롤링을 통해서 루프 연산이 개선된다면 FRAM 기록 작업을 줄여서 에너지를 절약하게 되고 정전 횟수가 감소하게 되므로 프로그램 수행시간을 줄일 수 있다.

3.1 하드웨어 구성

하드웨어는 크게 에너지 하베스팅을 위한 디바이스, 음성 데이터를 처리하는 디바이스로 이루어져 있다. 전체 시스템은 에너지 하베스팅으로 얻은 전력을 이용하여 구동되기 때문에 전력을 생산하는 장치와 전력을 저장하는 축전기가 필요하다. 본 논문에서는 빛에너지를 에너지원으로 사용하여 태양 전지 패널을 통해서 전력을 얻는다. 얻어낸 에너지는 태양 전지 패널에 부착된 에너지 관리 보드를 통해서 버퍼에 저장된다.

마이크 PMM-3738-VM1010-EB-R을 사용하여 음성 데이터를 입력받고 MSP430FR5994 보드를 사용하여 여러 가지 연산을 수행한다. MSP430FR5994 보드에서는 음성 데이터의 보정 및 신호처리 연산을 수행하고 이 때 얻은 데이터를 사용하여 딥러닝 모델에서의 추론 연산을 수행한다. MSP430FR5994 보드는 256KB의 FRAM을 사용할 수 있는 메모리 용량 제약 조건에서 동작한다. 따라서 딥러닝 모델의 매개변수 수를 줄여서 비휘발성 메모리 사용량을 절약하는 것이 중요하다.

3.2 음성 데이터

필요한 음성 데이터는 입력으로 사용하는 테스트 데이터와 딥러닝 모델을 훈련하고 검증하기 위한 훈련 데이터 두 종류이다. 실험에서 사용된 키워드는 "on", "off", "stop", "no", "yes"의 다섯 가지이다. 테스트 데이터는 마이크에서 얻으며 샘플 주파수는 7812Hz이고 4000개의 샘플을 사용하여 512ms 만큼의 시간에 해당하는 음성 데이터를 사용한다. 딥러닝 모델을 훈련하고 검증할 데이터로는 구글에서 제공하는 speech commands dataset[4]을 사용한다. 이 데이터셋은 하나의 음성 파일에 대해서 길이가 1초인 22,050 Hz의 샘플 주파수로 이루어져 있다. 따라서 훈련 데이터들은 테스트 데이터에 맞춰서 길이가 512ms인 7812Hz 샘플 주파수의 데이터로 변환한다.

실험에서 사용할 테스트 데이터는 마이크가 받은 음성 아날로그 데이터를 부호 없는 12bit의 PCM(Pulse Code Modulation)를 통해서 변환하여 얻어낸 디지털 값이다. 반면에 훈련 데이터는 -1, 1 사이의 부호가 포함된 값을 가진다. 따라서 식 (1)을 통해서 마이크의 데이터를 부호가 있는 값으로 변환한다. 실험에서 사용하는 테스트 데이터는 샘플 주파수가 7812Hz인 4000개의 샘플이기 때문에 4000개의 샘플에 대해서 식 (1)을 각각 계산한다.

$$y(t) = (x(t) - 950) \times 32, t = 0, 1, ..., 3999 \quad (1)$$

계산된 데이터에 대해서 그림 2의 (b)와 같이 MFCC(Mel Frequency Cepstral Coefficient) 계산을 하여 데이터를 처리한다[5]. MFCC는 음성신호에서 특징을 추출하는 방법을 이용하여 잡음을 제거한다. 특히 음성데이터는 고유한 정보를 얻을 수 있고 잡음에 강한 특성을 보이기 때문에 실험에 많이 사용되고 있다[6].

3.3 딥러닝 모델

MFCC 알고리즘을 적용한 3.2절의 데이터를 딥러닝 모델의 입력으로 사용한다. 본 논문에서는 Tensorflow 2.2.0 버전의 프레임워크를 사용하여 딥러닝 모델을 설계하고 훈련데이터를 학습한다. 이 때 설계한 딥러닝 모델의 각 계층에 대해서 해당 가중치와 편향치를 구하여 SONIC 모델의 입력 음성 데이터에 대한 딥러닝 추론 연산에 사용한다. 모델의 매개변수 수와 연산량을 줄이기 위해서 CNN(Convolutional Neural Network) 계열 모델 중 DS-CNN(Depth-wise Separable Convolutional Neural Network)을 사용한다. DS-CNN은 CNN 연산을 depth 방향과 point 방향으로 나누어서 계산한 모델로 CNN 연산에 비해서 매개변수 수와 합성곱 연산을 줄인 모델이다[7].

Ⅳ. 인터미턴트 환경에서 루프 언롤링 성능 분석

4.1절에서는 루프 변수 제어 연산의 비중과 루프

언롤링으로 인한 에너지 개선 효과 사이의 관계를 실험을 통해서 분석한다. 이어서 4.2절에서는 실험 방법 및 과정, 4.3절에서는 실험 결과를 분석한다. 마지막으로 4.4절에서는 루프 언롤링으로 인해서 발생하는 오버헤드에 관해서 알아본다.

4.1 루프 언롤링과 에너지 개선 효과 분석

Loop continuation 기법이 적용된 루프 연산에서 전체 루프 연산 중 루프 변수를 제어하는 연산의 비중과 그에 따른 각각의 루프 언롤링 효과를 조사할 필요가 있다. 본 논문에서는 31종의 임의 루프 연산에 대하여 각각 루프 변수 제어 에너지 *a*, 총 루프 연산 에너지 *b*, 언롤링 인자가 2인 루프 언롤링을 적용한 루프 연산 에너지 *c*를 측정한다. 에너지 측정 방법은 MSP430FR5994 보드에 코드를 실행해서 MONSOON Power Monitor로 에너지를 측정한다. *b*에 대한 *a*의 비율을 전체 루프 연산 중 루프 변수를 제어하는 연산의 비중으로 정하고 *b*, *c*에서 계산한 에너지를 비교하여 루프 언롤링 효과를 측정한다.

그림 3은 앞서 언급한 방법으로 측정한 루프 변수 제어 연산 비중에 따른 루프 언롤링 효과에 대한 실험 결과이다. 그림 3의 가로축은 루프 연산 에너지 중 루프 변수를 제어하는 연산 에너지의 비중이며 세로축은 루프 언롤링 적용 시 감소하는 에너지 비율이다. 대체로 루프 변수 제어 연산 비중이 클수록 루프 언롤링 효과가 큰 경향을 확인할 수 있다.

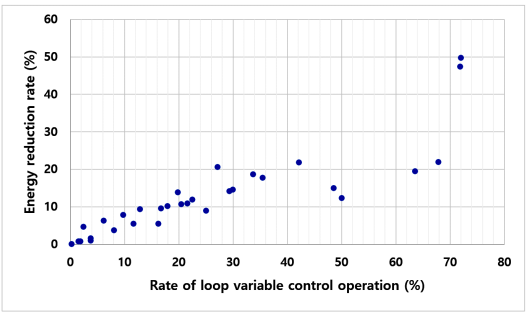


그림 3. 루프 변수 제어 연산 비중에 따른 루프 언롤링 효과
Fig. 3. Loop unrolling effect according to the weight of loop variable control operation

4.2 실험 과정

MSP430FR5994 보드를 사용하여 인터미턴트 컴퓨팅 환경의 키워드 검색 프로그램을 수행하고 프로그램의 재부팅 횟수와 수행시간을 측정한다. 우선 빛에너지를 통해 얻은 전력을 저장하기 위해서 0.1F, 0.33F, 0.47F 용량의 축전기를 각각 사용하여 실험한다. 또한 빛에너지의 밝기에 따른 성능을 확인하기 위해서 각각 445lx, 762lx, 1195lx의 환경에서 실험을 진행한다. 실험을 마친 이후에 SONIC 모델과 LU-SONIC 모델에 대해서 MCU의 재부팅 횟수를 체크하고 수행시간을 비교함으로써 성능을 평가한다.

실험에서 평가할 모델은 베이스라인 모델인 SONIC 모델과 루프 언롤링을 적용한 모델인 LU-SONIC 모델이다. 루프 언롤링 방법은 다음과 같다. 우선 SONIC 모델에 존재하는 루프 연산에 대해서 각 루프 연산 중 루프 변수를 제어하는 연산의 비율을 에너지 측정을 통해 계산한다. 각 루프 연산의 계산한 비율이 미리 정한 임계치보다 작다면 루프 언롤링을 하지 않고 임계치보다 크다면 언롤링 인자를 2로 잡아서 루프 언롤링을 적용한다. 언롤링 인자가 3 이상인 경우 언롤링 인자가 2인 경우보다 루프 안에 코드가 증가하므로 간헐적인 전력 공급 상황에서 중복 계산 횟수가 증가하는 상황이 발생한다. 반면에 언롤링 인자가 3 이상인 루프의 FRAM 접근 횟수는 더욱 줄어드는 장점이 있다. 따라서 언롤링 인자가 3 이상의 경우에 대해서는 언급된 Trade-off 관계에 대한 심도 있는 연구가 차후 필요하다.

그림 3에 따르면 루프 변수 제어 연산의 에너지 비율이 3.77%일 때 루프 언롤링으로 개선되는 에너지 비율이 1.54%이다. 또한 루프 변수 제어 연산 비중이 작을수록 루프 언롤링을 적용하여 개선할 수 있는 에너지 비율이 줄어들어 루프 언롤링 효과가 감소하는 경향을 보인다. 따라서 본 논문에서는 루프 변수 제어 연산의 에너지 비율이 3% 미만일 때 루프 언롤링 효과가 미미하다고 판단하여 실험의 임계치를 3%로 정한다. 실험에서 루프 언롤링을 적용한 루프 연산은 키워드 검색 프로그램의 루프 연산 중에서 데이터 스케일링 연산, pre-emphasis 연산, Hamming window 연산, 이산 푸리에 변환 연산, 이산 코사인 변환 연산, MFCC값 저장 연산, 합성곱 연산, 합성곱 결과 저장 연산, 배열 초기화 연산, 합성곱 계층에서 편향치 연산, 완전 연결 계층의 결과 저장 연산, 완전 연결 계층에서 편향치 연산으로 총 12종이다.

본 논문의 실험은 안정적인 에너지 하베스팅 환경부터 열악한 에너지 하베스팅 환경까지 다양한 환경에서 수행되었다. 실험에서 가장 안정적인 에너지 하베스팅 환경은 1195lx 빛에너지와 0.47F 축전기로 이루어졌으며, 가장 열악한 에너지 하베스팅 환경은 445lx 빛에너지와 0.1F 축전기로 이루어졌다.

4.3 실험분석

하베스팅한 에너지를 사용하지 않고 연속적인 에너지원을 사용하여 끊김없이 전력이 공급되는 상황인 연속적 전력 공급 상황과 하베스팅한 에너지를 이용하는 인터미턴트 컴퓨팅 환경에서 다양하게 실험하였다.

4.3.1 연속적인 전력 공급 상황

표 1은 연속적인 전력 공급 상황에서 SONIC 모델과 LU-SONIC 모델의 주요 연산별 수행시간을 비교한 결과이다. 두 모델에 대해서 각각 10번의 실험을 반복하였고 표 1의 데이터는 평균값이다. SONIC 모델과 LU-SONIC 모델의 수행시간은 각각 52.409s, 50.771s로 LU-SONIC 모델의 수행시간이 SONIC 모델보다 3.12% 감소했다. 표 1을 보면 MFCC 연산이 전체 키워드 검색 프로그램의 지배적인 연산임을 알 수 있다. 표 1에서 알 수 있듯 SONIC 모델과 LU-SONIC 모델의 수행시간을 비교해보면 LU-SONIC 모델에서 개선된 수행시간은 대부분 MFCC 연산의 결과이다. 반면에 DS-CNN에서의 연산 과정에서 루프 언롤링으로 인한 수행시간 개선 효과는 없었다. LU-SONIC 모델에서는 루프 반복 횟수를 확인하는 조건문 연산이 추가되는 요인으로 DS-CNN 연산 과정의 수행시간 개선효과가 미미했다.

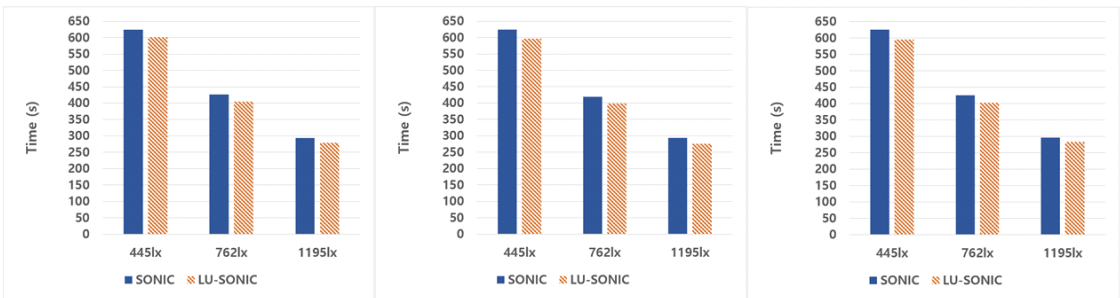
표 1. 연속적인 전력 상황에서 수행시간 비교
Table 1. Comparison of run time in continuous power

Model	MFCC run time (s)	DS-CNN run time (s)	Total run time (s)
SONIC	49.225	3.181	52.409
LU-SONIC	47.586	3.184	50.771

4.3.2 인터미턴트 컴퓨팅 환경

그림 4는 하베스팅한 에너지를 이용하여 인터미턴트 컴퓨팅 환경에서 SONIC 모델과 LU-SONIC 모델의 수행시간을 비교한 결과이다. 그림 4의 (a), (b), (c)는 각각 0.1F, 0.33F, 0.47F 축전기를 사용한 결과이다. 실험에서 사용한 3가지 축전기에 대해서 각각 445lx, 762lx, 1195lx 빛에너지를 에너지원으로 사용하여 총 9가지 환경에서 수행시간을 측정했다. 따라서 총 9가지 실험 환경에 대해서 각각 10번의 실험을 반복하였고 그림 4의 데이터는 평균값이다. 0.1F 축전기를 사용한 경우, 445lx, 762lx, 1195lx 환경에서 LU-SONIC 모델의 수행시간은 SONIC 모델에 비해서 각각 3.67%, 4.86%, 5.06% 감소했다. 다음으로 0.33F 축전기를 사용한 경우, 445lx, 762lx, 1195lx 환경에서 LU-SONIC 모델의 수행시간이 SONIC 모델에 비해서 각각 4.49%, 4.69%, 5.88% 감소했다. 마지막으로 0.47F 축전기를 사용한 경우, 445lx, 762lx, 1195lx 환경에서 LU-SONIC 모델의 수행시간이 SONIC 모델에 비해서 각각 4.79%, 5.45%, 4.20% 감소했다. 그림 4를 보면 수행시간의 관점에서 인터미턴트 컴퓨팅의 성능이 축전기의 용량에 영향 받기보다는 공급되는 에너지의 크기에 영향을 더 많이 받음을 의미한다.

그림 4가 인터미턴트 컴퓨팅 환경에서 SONIC 모델과 LU-SONIC 모델의 수행시간을 비교했다면 그림 5는 인터미턴트 환경에서 재부팅 횟수를 비교한 결과이다. 그림 5의 데이터는 SONIC 모델에 대한 LU-SONIC 모델의 재부팅 횟수 감소 비율이다. 마찬가지로 그림 4에서와 같은 9가지 환경에서 실험을 진행했다. 그림 5의 실험 결과를 분석하면 9가지 환경 모두에서 LU-SONIC 모델이 SONIC 모델에 비해서 재부팅 횟수가 감소하였다. 0.1F, 0.33F 축전기를 사용한 결과를 보면 빛에너지의 조도가 증가할수록 LU-SONIC 모델의 재부팅 횟수 개선 비율이 더 효과



(a) 0.1F capacitor (b) 0.33F capacitor (c) 0.47F capacitor

그림 4. 인터미턴트 컴퓨팅 환경에서 수행시간 비교
Fig. 4. Comparison of run time in intermittent computing environment

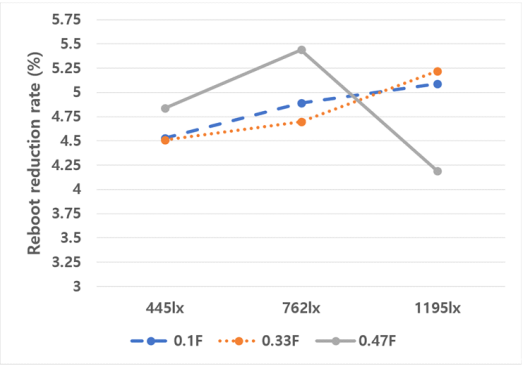


그림 5. 인터미턴트 컴퓨팅 환경에서 재부팅 횟수 비교
Fig. 5. Comparison of the number of reboots in intermittent computing environment

적인 경향이 있는 것을 알 수 있다. 빛에너지 조도가 증가한다는 것은 인터미턴트 컴퓨팅 환경의 에너지 공급이 비교적 원활하다는 것을 의미하고 이는 4.4절에서 언급될 오버헤드가 비교적 적은 상황이다. 따라서 에너지 공급이 원활할수록 루프 언롤링으로 인한 성능 개선이 더욱 효과적인 경향을 보인다. 그러나 0.47F 축전기를 사용한 결과에서 빛에너지의 조도가 증가했음에도 재부팅 횟수 개선 비율이 감소하였다. 이는 루프 언롤링으로 인한 인터미턴트 컴퓨팅 성능 개선의 임계치에 해당하는 환경임을 의미한다.

요약하면, 연속적인 전력 공급 상황에서 LU-SONIC 모델은 SONIC 모델에 비해서 수행시간이 3.12% 감소하였다. 간헐적인 전력 공급 상황에서는 축전기 용량과 빛에너지 세기를 변화하며 9가지 환경에서 실험을 진행하였고, 실험 결과 열악한 에너지 하베스팅 환경부터 안정적인 에너지 하베스팅 환경까지 9가지 환경 모두에서 LU-SONIC 모델이 SONIC 모델에 비해서 수행시간과 재부팅 횟수가 감소했다. 실험을 통하여 인터미턴트 컴퓨팅 환경의 키워드 검색 프로그램에 대해서 LU-SONIC 모델은 SONIC 모델에 비해서 재부팅 횟수가 최대 5.44% 감소한 결과를 얻었다. 재부팅 횟수가 감소한 것은 프로그램의 사용되는 에너지 및 프로그램의 수행시간이 감소했다는 것을 의미하고 실제로 그림 4의 결과는 LU-SONIC의 수행시간이 감소하였다는 것을 입증하였다. 따라서 LU-SONIC 모델에서 의미 있는 에너지 절약을 수행함을 확인할 수 있었다.

특히 가장 열악한 에너지 하베스팅 환경인 445lx 빛에너지와 0.1F 축전기에서 LU-SONIC 모델은 SONIC 모델에 비해서 수행시간과 재부팅 횟수가 각각 3.67%, 4.53% 감소하였다. 이는 열악한 에너지 하베스팅 환경에서도 LU-SONIC 모델이 SONIC 모델보다 더 좋은 성능을 가짐을 보인다.

4.4 루프 언롤링 오버헤드

LU-SONIC 모델은 SONIC 모델에 비해서 연산량이 증가하는 경향이 있다. 설명을 위해서 언롤링 인자가 1인 루프 코드에서 한 번의 반복(iteration)에 해당하는 연산 단위를 블록(block)이라고 정의하자. 이러한 정의를 기준으로 그림 1을 보면 한 번의 반복에서 (a) 코드, (b) 코드 각각의 연산량은 1블록, 2블록으로 (b) 코드가 더 많다. 즉, 루프 언롤링을 적용한 루프 연산은 한 번의 반복 과정에서 연산량이 증가한다. 이렇게 증가된 연산량은 정전이 발생했을 때 중복해서 처리하는 연산량이 증가하는 원인이 된다.

표 2는 루프 반복 횟수와 정전 발생 시 1블록 중복 계산 횟수 p와 2블록 중복 계산 횟수 q를 나타낸 표이다. 이 때 표의 데이터는 4.2절에 언급된 루프 언롤링 적용된 12종의 루프 연산에 대해 조사한 값이다. 표 2를 보면 LU-SONIC 모델과 SONIC 모델의 루프 반복 횟수는 각각 824651, 1660377로 루프 반복 횟수는 LU-SONIC 모델이 SONIC 모델의 절반보다 작음을 알 수 있다. 이는 $2n + 1$ 번의 반복을 수행하는 루프의 경우 LU-SONIC 모델은 루프 언롤링을 적용하여 n 번의 반복을 수행하는 루프로 처리되기 때문이다. LU-SONIC 모델의 루프 반복 횟수가 SONIC 모델보다 줄어드는 특징으로 인해서 LU-SONIC이 SONIC 보다 개선된 성능을 보인다. 하지만 표 2에서 SONIC과 LU-SONIC의 중복 계산되는 총 블록 수 t를 식 (2)를 통해 계산하여 비교해보면 각각 1059블록, 1527블록으로 LU-SONIC이 더 많다. 결과적으로 LU-SONIC 모델은 SONIC 모델에 비해서 중복 계산량이 증가하지만 루프 반복 횟수가 줄게 되어 FRAM 연산량을 개선하였기 때문에 더 효과적인 인터미턴트 컴퓨팅 연산을 가능하게 했다. 따라서 LU-SONIC에서 증가된 중복 계산 횟수를 줄일 수 있다면 전체 처리 시간 및 에너지 사용량을 더욱 효과적으로 단축할 수 있을 것으로 기대된다.

표 2. SONIC 모델과 LU-SONIC 모델의 오버헤드 비교
Table 2. Comparison of overheads between SONIC model and LU-SONIC model

Model	1 block recomputation (p)	2 block recomputation (q)	number of iteration
SONIC	1059	0	1660377
LU-SONIC	549	489	824651

$$t = p + 2 \times q \quad (2)$$

그 외에도 언롤링 인자가 3 이상일 때 루프 언롤링 오버헤드를 생각해 볼 수 있다. 3 이상의 언롤링 인자를 적용한 loop continuation 기반 루프의 경우 루프의 FRAM 접근 횟수는 더욱 줄어드는 장점이 있다. 그러나 루프 안에 블록이 증가하므로 간헐적인 전력 공급 상황에서 중복 계산 횟수가 증가하는 경우가 발생한다. 이러한 점을 고려해서 최적의 루프 언롤링 적용 효과를 가지는 언롤링 인자를 찾는 연구가 차후 필요할 것으로 판단된다.

V. 결론 및 향후 연구

Loop continuation 기법은 루프 변수를 FRAM 변수로 사용하여 인터미턴트 프로그램에서 루프 연산 성능을 개선하였다. 하지만 루프 연산이 지배적인 인터미턴트 프로그램에 대해서 잦은 FRAM 변수의 연산은 프로그램의 성능을 저하시킨다. 본 연구에서는 루프 언롤링을 적용하여 인터미턴트 컴퓨팅 기반의 키워드 검색 프로그램에 대한 재부팅 횟수를 최대 5.44% 개선함을 확인하였다.

앞서 언급된 것과 같이 특정 인터미턴트 컴퓨팅 환경에 알맞는 적정한 언롤링 인자를 찾는 기법이 개발된다면 인터미턴트 컴퓨팅의 성능을 더욱 개선할 수 있을 것으로 기대한다.

References

[1] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," in *Proc. 16th Int. Conf. Architectural Support for Program. Lang. and Operating Syst.*, pp. 159-170, Newport Beach, USA, Mar. 2011.

[2] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent execution without checkpoints," in *Proc. ACM Program. Lang.*, vol. 1, no. 96, pp. 1-30, Oct. 2017.

[3] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proc. 24th Int. Conf. Architectural Support for Program. Lang. and Operating Syst.*, pp. 199-213, Providence, USA, Apr. 2019.

[4] P. Warden, *Speech commands: A public dataset for single-word speech recognition* (2017), Retrieved Feb. 12, 2021, from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz.

[5] F. Haytham, *Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between* (2016), Retrieved Feb. 12, 2021, from https://haythamfayek.com/2016/04/21/speech-processingfor-machine-learning.html.

[6] H. Bae, H. Lee, and S. Lee, "Voice recognition-based on adaptive MFCC and deep learning for embedded systems," *J. Inst. Contr., Robot. and Syst.*, vol. 22, no. 10, pp. 797-802, Oct. 2016.

[7] C. François, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vision and Pattern Recog.*, pp. 1251-1258, Honolulu, USA, Jul. 2017.

김 시 현 (Sihyun Kim)

2022년 2월 : 홍익대학교 컴퓨터공학 학사

<관심분야> 사물인터넷, 인터미턴트 컴퓨팅, 임베디드 시스템

[ORCID:0000-0001-8221-3779]

하 란 (Rhan Ha)

1987년 : 서울대학교 컴퓨터공학 학사

1989년 : 서울대학교 컴퓨터공학 석사

1989년~1990년 : KT 전임연구원

1995년 : University of Illinois at Urbana-Champaign 컴퓨터공학 박사

1995년~현재 : 홍익대학교 컴퓨터공학과 교수

<관심분야> 사물인터넷, 인터미턴트 컴퓨팅, 임베디드 시스템, 모바일 컴퓨팅, 실시간 시스템, SW보안

[ORCID:0000-0001-9861-362X]