| Oral Session IX : Big Data, Smart Energy ICT, Smart Information

좌장 : 신춘성 (전남대)

유사 서비스 함수를 위한 코드 모듈들의 구조 내 저전력 연구

윤예동, 문소영, 김영철
홍익대학교 소프트웨어융합학과
e-mail : yedong@mail.hongik.ac.kr, {whit2, bob}@hongik.ac.kr

Low Power Consumption Research in Code Module Structures for a Similar Service Function

Yedong Yoon, S. Y. Moon, R. Young Chul Kim
SE Lab, Dept. of Software and Communications Engineering, Hongik University

요 약

앞으로의 4차 산업혁명 시대에서 사물인터넷 (Internet of Things, IoT) 기기는 증가할 것으로 전망된다. 특히 스마트 센서와 같은 IoT 장치들은 지역적으로 넓게 흩어져 있으며 수량이 많다. 그로 인하여 각 장치에 안정적으로 전력을 제공할 방법이 제한된다. 전력원이 제한되기 때문에 IoT 장치는 전력 효율성을 높이는 것이 주요 관심사다. 여러 분야에서 이를 해결하기 위한 전력 최적화 방법이 연구되고 있지만, 개개인의 개발자들이 소스 코드를 작성할 때 적용할 수 있는 지표, 방법, 방법론, 도구 등의 연구는 매우 미비한 상태이다. 이를 해결하고자 소스 코드 수준에서 전력 최적화하는 방법을 제안하고자 한다. 본 논문을 통해 저전력 코드를 적용할 부분의 식별 방법을 제안한다. 실험 방법으로 같은 서비스를 제공하는 알고리즘들 사이에서 전력 소모량이 적은 알고리즘 연구 및 지표를 제시한다. 이를 통해 소스 코드 구조 안에서 전력 소모를 줄이는 방법을 분석하고자 한다.

키워드: 저전력, IoT, 리팩토링, 전력 최적화 코드

1. 서론

4차 산업 이후 IoT device들의 사용이 증가하고 있다. 많은 수의 흩어져 있는 각 장치에 안정적인 전력을 공급하기 위한 설비를 갖추려면 비용이 많이 든다. 이러한 비용을 줄이기 위하여 IoT 장치들은 배터리와 같은 제한된 전력 공급 설비를 종종 이용한다. 한정된 전력량 안에서 더 오랜 시간 서비스하기 위해서는 전력 소모를 최소화하는 것이 중요한 이슈이다. 전력 소모를 최소화하기 위하여 여러 분야에서 연구가 진행되고 있다. 하지만 다양한 분야에 기법들을 개발자가 습득하여 전력 최적화된 코드를 작성하는 것은 어렵다. 이를 해결하기 위해 비슷한 서비스를 제공하는 소스 코드 구조 안에서 전력 소모를 줄이는 방법을 연구 중이다. 실험을 위해 Sorting이라는 서비스를 제공하면서 시간 복잡도 측면에서 비슷한 효율성을 가지는 Quick, Merge, Heap Sort 알고리즘을 선정하였다. 각 알고리즘을 구현한 소스 코드들 사이에서 소비하는 전력량을 비교한다. 본 논문은 2장에서 관련 연구, 3장에서 실험 구성, 4장에서 실험 결과, 5장에서 결론 및 향후 연구 방향에 대해 언급한다.

2. 관련 연구

Antonio Vetro는 소스 코드수정을 통해 전력 소모를 줄일 수 있는 패턴을 찾아 Energy Code Smell을 정의했다.[1]

$$P = V * I \quad \text{.....(수식 1)}$$

$$E = Vcc * Icc * \triangle t \quad \text{.....(수식 2)}$$

수식1은 Energy Code smell에서 코드 패턴에 따른 전력 소모 비교에 사용한 수식이다. 하지만 수식1은 소스 코드의 실행 시간이 고려되지 않았다. 즉 소비 전력이 비슷한 두 코드 사이에서 알고리즘의 차이로 실행 시간이 다른 코드 사이에 올바른 소비 전력량의 비교가 이루어지지 않는다. 본 논문은 소스 코드의 실행 시간을 포함하여 소모한 전력량을 계산하는 수식2를 사용할 것을 제안한다. 수식2는 소모 전력량을 구하기 위해 사용된 수식이다. E는 전기 에너지 소모량(소비 전력량)이다. $\triangle t$는 소스 코드의 실행 시간이다. Vcc는 $\triangle t$ 동안에 하드웨어에 공급된 평균 전압이고, Icc는 $\triangle t$ 동안에 소모된 평균 전류이다.

(표 1) 알고리즘들의 시간, 공간 복잡도

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quick Sort	$\Omega(n\log_2(n))$	$\theta(n\log_2(n))$	$O(n^2)$	$O(\log_2(n))$
Merge Sort	$\Omega(n\log_2(n))$	$\theta(n\log_2(n))$	$O(n\log_2(n))$	$O(n)$
Heap Sort	$\Omega(n\log_2(n))$	$\theta(n\log_2(n))$	$O(n\log_2(n))$	$O(1)$

표1은 Quick, Heap, Merge Sort 알고리즘의 시간 복잡도와 공간 복잡도이다[3]. 알고리즘들은 Best와 Average case에서 같은 시간 효율성을 가지며 각기 다른 공간 복잡도를 지닌다. Quick, Merge Sort는 분할 정복 전략을 취하고 있으며 Heap Sort는 변형 정복 전략을 사용한다. 시간 공간 복잡도를 통해 세 알고리즘을 비교한다면 Heap Sort가 가장 우수한 알고리즘이다.

3. 실험 구성

3.1 실험 장비

그림 1 MCBSTM32F400(Evaluation board)와 Ulink plus

그림 1은 측정에 사용한 Keil 사의 Evaluation board와 Ulink plus가 연결된 모습이다. 컴파일 과정에서의 변인을 통제하기 위하여 컴파일러 최적화 옵션을 해제하였다.

(표 2) Evaluation board의 사양

CPU	ARM CortexTM-M4
Architecture	1MB Flash & 192KB RAM
on chip Memory	8MB NOR Flash, 512MB NAND Flash, 2MB SRAM, 8KB I2C EEPROM with NFC interface
Compiler	arm compiler 5 Version

표2는 측정에 사용한 Evaluation board의 사양이다.

3.2 실험 순서

(1)Keil uVision5 IDE에서 Evaluation board에 맞는 프로젝트 생성한다. (2) 전력을 측정하고자 하는 소스 코드(C 언어)를 입력한다 (3) 소스 코드를 빌드한 후 보드에 upload한다. (4) 디버깅 모드를 통하여 소스 코드의 실행 시간과 소비한 전력 등의 데이터를 수집한다.

4. 실험 결과

(표 3) 알고리즘과 데이터 사이즈 별 측정 결과표

Algorithm type	Run Time(ms)	Average Current(mA)	Average Voltage(V)	Energy consumption(μJ)	Data Size
Quick Sort	2.21	7.07	3.28	51.2	100
	5.21	7.27	3.27	123.9	200
	8.74	7.28	3.27	208.1	300
Heap Sort	4.67	7.23	3.27	110.4	100
	9.95	7.4	3.27	240.8	200
	17.2	7.63	3.27	429.1	300
Merge Sort	3.63	7.25	3.27	86.1	100
	8.96	7.47	3.27	218.9	200
	14.22	7.5	3.27	348.7	300

표3은 각 알고리즘이 각기 다른 크기의 데이터 세트를 정렬하기 위해 소모한 전력량을 측정한 결과이다. 시간, 공간 복잡도 비교법에서는 Heap Sort가 가장 뛰어난 알고리즘이지만 실험 결과 Quick Sort가 가장 전력을 적게 쓰는 것으로 나타났다. 이 실험 결과를 통해 시간 공간 복잡도 비교법을 통한 전력 소모를 예측하는 방식에는 한계가 있다.

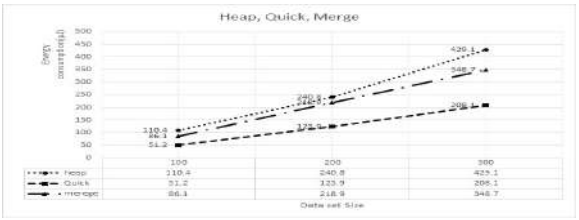

그림 2 Data Size와 Heap, Quck, Merge Sort의 소비 전력량

그림2는 표3의 결과를 비교하기 위한 전력 소모량 비교 그래프이다. 데이터들의 크기에 따른 실행 시간의 증가량은 시간 복잡도를 따름을 알 수 있다. 하지만 같은 시간 복잡도를 지니는 알고리즘들 사이에서 Quick Sort가 빠르게 동작하는지는 알 수 없다. 이는 시간 복잡도 비교법이 현대 컴퓨터 아키텍처를 충분히 반영하지 않았기 때문이다. 컴퓨터가 정렬을 수행하기 위해서는 비교할 데이터가 CPU 내부에 Cache로 위치해야 한다. 만약 Cache 내부에 비교할 데이터가 없으면 메모리로부터 비교할 데이터를 가져와야 하고 이는 소스 코드의 수행 시간이 길어지게 되는 원인이다. Quick sort 알고리즘은 다른 알고리즘들에 비해 참조의 지역성이 뛰어나다[4]. 즉 Quick Sort 알고리즘이 다른 알고리즘들에 비해 Cache에 필요한 데이터를 가져오는 작업이 적어 그림 2와 같이 적은 실행 시간을 가기에 전력 효율적이다.

5. 결론 및 향후 연구

기존에 널리 사용되는 알고리즘 비교에 사용되는 시간, 공간 복잡도는 소스 코드의 실행 시간 관점에 소모하는 전력량을 예측하는데 적절하지 않다. 향후 연구로써 소프트웨어를 전력 최적화를 위한 코드, 설계 패턴 및 개발 프로세스에 적용할 수 있는 도구와 방법론을 연구할 예정이다.

Acknowledgement

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R1I1A305040711, No. 2021R1I1A1A01044060)을 받아 수행된 연구임.

참 고 문 헌

[1] Antonio Vetro, Luca Ardito, Giuseppe Procaccianti, Maurizio Morisio, “Definition, Implementation and Validation of Energy Code Smells: an Exploratory Study on an Embedded System”The 3rd Int’l Conf. on Smart Grids, Green Comm. and IT Energy-aware Tech,2013. pp.34-39,

[2] Vivek Tiwari, Sharad Malik, Andrew Wolfe, "Compilation Techniques for Low Energy: An

Overview", IEEE Symposium on Low Power Electronics 1994. pp.38-39.
[3] Anany Levitin, "Introduction to the Design and Analysis of Algorithms", ISBN-13(978-0132316811)
[4] Anthony LaMarca, Richard E. Ladner, "The Influence of Caches on the Performance of Sorting", Journal of Algorithms Volume 31, Issue 1 1999. pp.66-104.