Loop 코드의 전력 효율성 향상을 위한 C 코딩 가이드라인

이재욱, 김순겸, 홍장의
충북대학교 컴퓨터과학과
e-mail : {jwlee, skkim}@selab.cbnuac.kr, jehong@chungbuk.ac.kr

C Coding guideline to improve energy efficiency of loop code

Jae-Wuk Lee, Soon-Kyeom Kim, Jang-Eui Hong
Dept. of Computer Science, Chungbuk National University

요 약

최근 스마트폰 및 태블릿 PC 와 같은 다양한 모바일 기기의 사용이 증가하고 있다. 이러한 기기들은 배터리 사용으로 인해 전력공급이 제한되어, 소모전력 효율 향상이 요구된다. 이에 따라 최근에는 소프트웨어에 의한 소모전력의 효율성을 향상시키기 위한 연구들이 진행되고 있다. 그러나 소프트웨어의 코드 구조의 분석을 통해 전력 효율성을 향상시키기 위한 연구는 미비하다. 따라서 본 논문에서는 코드 구조의 변경에 따른 소모 전력 효율성을 분석하여, 소프트웨어 개발이나 유지보수 단계에서 전력 소모량을 감소시킬 수 있는 가이드라인을 제시하고자 한다.

1. 서론

최근 스마트폰 및 태블릿 PC 와 같은 다양한 모바일 기기들의 사용이 증가하고 있다. 이러한 모바일 기기들은 배터리의 전력으로 구동되므로 전력 공급에 한계성이 있다. 이러한 환경에서 소프트웨어의 기능과 성능을 유지한 채 운용시간을 늘려 지속성을 유지하는 것은 중요한 이슈이다.

모바일 기기의 전력 효율을 높이기 위한 목적으로 배터리 용량 증가, 하드웨어 부품 최적화 등의 방법들이 연구되어 왔다. 그러나 소프트웨어의 구조변경을 통해 기능은 유지한 채 전력 소모량을 감소시킬 수 있음이 발견되어 소프트웨어에 대한 소모전력 절감 연구들이 진행되고 있다[1, 2].

기존의 소프트웨어 소모전력에 대한 연구들은 코드 분석을 통해 전력 소모량의 분석 및 예측 방법에 주안점을 두고 있다[3, 4, 5]. 그러나 이러한 연구들은 구현된 코드에 대한 소모전력을 분석하는 방법들로써, 소프트웨어 개발 과정에서 저전력 요구사항을 반영하기 어렵다는 단점이 있다. 따라서 본 논문에서는 코드 구조에 따른 전력 효율성을 분석하여, 저전력 요구사항을 만족시킬 수 있는 소프트웨어 구현을 위한 코딩 가이드라인을 제시한다.

본 연구에서는 loop 의 구조에 따른 소모 전력 분석을 수행하였다. 이는 loop 코드는 그 특성상 호출될 때마다 여러 번 수행되기 때문에, 구조 변경에 따른 소모 전력 절감 효과가 클 것으로 판단하였기 때문이다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구들을 소개하고, 3 장에서는 소모전력 절감을 위한 loop 코드 변환 기법을 소개한다. 4 장에서는 3 장에서 제시한 기법에 대한 실험 결과를 소개하고, 이를 기반으로 5 장에서 loop 의 구조 변경을 통한 전력 효율성 향상 가이드라인을 제시한다. 마지막으로 6 장에서는 결론과 향후 연구를 기술한다.

2. 관련연구

Jun 의 연구[3]는 컴포넌트 기반의 소프트웨어 개발에서 소모전력을 분석하였다. 소프트웨어 컴포넌트 아키텍처를 오토마타로 정의하고, 실행시간에 대한 가중치를 부여하여, 오토마타에 대한 시뮬레이션을 통해 소모전력을 예측하였다. 이 연구는 소프트웨어 개발의 초기단계에서 소모전력 절감을 시도하고 있으나, 실제 코드 구현 단계에서 적용하기에 어려움이 있다는 단점이 있다.

Gottschalk 의 연구[4]에서는 전력 소모가 큰 코드 패턴을 식별하여 Energy Code Smell 을 정의하였다. 이 연구에서는 일반적인 code smell 들에 대하여 소모 전력에 미치는 영향을 분석하였고, 이에 대한 Restructuring 을 통해 소모 전력 절감을 유도하였다. 그러나 기법 적용에 대한 가이드라인이 제시되지 않아, 실제 개발 과정에 적용하는데 어려움이 존재한다.

Brnadolese 의 연구[5]는 소스코드의 변환이 전력 소모에 미치는 영향을 분석하였다. 데이터 구조, 인라인 함수, 제어 구조 등에 대한 분석을 통해 소모 전력 절감을 위한 코드 변환 기법들을 제시하였다. 그러나 제시된 기법은 특정 예제에 국한되어, 실제 코드에 적용하기 어렵다는 단점이 있다.

3. Loop 구조 변환 기법

Loop 코드는 그 특성상 종료 조건 검사 연산과 loop 내부의 코드 블록이 반복적으로 수행되기 때문에, 코드 구조 변환을 통한 소모 전력 절감 효과가 누적되어 나타난다. 따라서 본 연구에서는 loop 코드를 대상으로 구조 변환을 통한 소모전력 분석을 수행하였으며, 그 결과 제시하는 기법들은 <표 1>과 같다.

<표 1> Loop 구조 변환 기법

변환 기법	설명
Index Elimination	Loop 내부의 증감변수들을 단일 변수로 통합
Reduce Loop Structure	중첩 구조를 단일 loop 로 축소
Global Variables	Loop 내부의 전역변수를 지역변수로 대체
Extract Expression	동일한 결과를 갖는 연산식을 loop 밖으로 추출

4. 전력 효율성 향상을 위한 loop 코딩 가이드라인

<표 1>에서 제시한 loop 코드 구현 가이드라인을 다음과 같이 제안한다.

G1) Eliminate Index: Loop 가 다수의 증감 변수를 가지며, 증감변수의 범위와 증감 값이 같은 경우 단일 변수로 통합하여 증감 연산 횟수를 감소시킨다.

```
for (i1 = 0, i2 = 0, i3 = 0; i1 < SIZE; i1++)
{
  ...
  a[i2++] = b[i3++];
  ...
}
```

```
for (i1 = 0; i1 < SIZE; i1++)
{
  ...
  a[i1] = b[i1];
  ...
}
```

(그림 1) G1)에 대한 예시

G2) Reduce Loop Structure: 중첩 구조에서 외부 loop 가 내부 loop 이외의 코드를 갖지 않은 경우 단일 loop 로 축소하여 종료 조건 검사 연산의 횟수를 감소시킨다.

```
int a[m][n];

for (i = 0; i < m; i++)
{
  for (j = 0; j < n; j++)
  {
    a[j][i] = 0;
  }
}
```

```
int a[m][n];
int *p = &a[0][0];

for (i = 0; i < size; i++)
{
  *p++ = 0;
}
```

(그림 2) G2)에 대한 예시

G3) Replace Global Variables: Loop 내부의 전역변수는 지역변수로 변환하여 시스템의 오버헤드를 감소시킨다.

G3-1) 전역변수에 저장된 값을 읽어 오는 연산의 경우, 대상 loop 직전에 전역변수의 값을 지역변수에 저장하는 문장을 추가한다.

G3-2) 전역변수에 값을 저장하는 연산의 경우, 대상 loop 직후에 지역변수의 값을 전역변수에 저장하는 문장을 추가한다.

```
int sum;

int func ()
{
  ...
  for (i = 0; i < size; i++)
  {
    sum += a[i];
  }
  ...
}
```

```
int sum;

int func ()
{
  int sum_local;
  ...
  for (i = 0; i < size; i++)
  {
    sum_local += a[i];
  }
  sum = sum_local;
  ...
}
```

(그림 3) G3-2)에 대한 예시

G4) Extract Expression: 동일한 결과값을 내는 연산을 단일 변수로 대체하여 연산 횟수를 감소시킨다.

```
for (i = 0; i < size; i++)
{
  a[i] = x*y + i;
}
```

```
temp = x*y;
for (i = 0; i < size; i++)
{
  a[i] = temp + i;
}
```

(그림 4) G4)에 대한 예시

5. 실험 및 검증

5.1 실험 환경

본 연구에서는 제안한 loop 코드 가이드라인에 대한 소모전력 절감효과를 확인하는 실험을 위해 코드 기반 전력분석 도구인 XEEMU[6]를 사용하였다. <표 2>는 XEEMU 의 플랫폼 환경을 요약한 것이다. 해당 도구는 실측 대비 1.6%~3.0%의 오차율이 있어 정확도 높은 소프트웨어의 소모전력 분석 결과를 얻을 수 있다.

<표 2> XEEMU 의 플랫폼 환경

CPU	Intel XScale 80200 processor, 266 to 733 MHz in 66MHz
Architecture	ARM pipelined RISC
Memory	32 MByte Micron SDRAM, 100MHz
Compiler	GCC 4.1.1
Measurement tap	CPU core current, IO current, System peripherals

실험을 위해 먼저 4 장에서 제시한 가이드라인에 대한 기본 코드를 구현하여 소모전력 효율 변화를 분

석하였다. 이를 기반으로 제시한 가이드라인이 실제 소프트웨어 개발에 적용되었을 때의 효용성을 검증하기 위한 목적으로 오픈소스를 대상으로 가이드라인 적용 실험을 수행하였다. 이를 위해 C++로 구현된 Huffman code 를 대상으로 가이드라인을 적용하여 전력 소모량 변화를 측정하였다. Huffman code 는 데이터의 무손실 압축에 쓰이는 알고리즘이다[7]. 이는 압축 소프트웨어 및 MPEG 등의 기본 알고리즘으로 많이 사용되고 있으며, loop 코드의 비중이 높아 본 연구의 실험에 적합할 것으로 판단하였다. 실험을 위한 입력 파일은 10KB 의 텍스트 파일을 사용하였다.

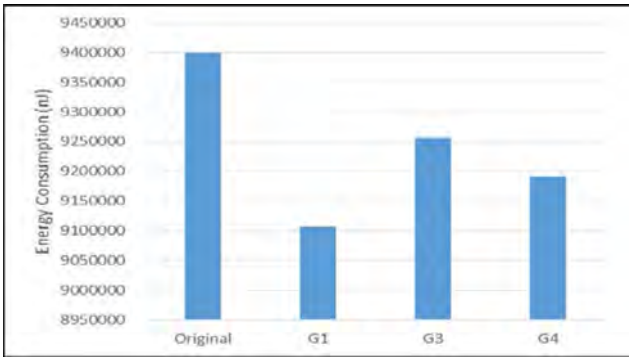
5.2 실험 결과 및 분석

먼저 4 장에서 제시한 loop 코드 가이드라인의 적용에 따른 전력 소모량은 <표 3>과 같다. 해당 실험은 제시한 가이드라인의 검증을 위한 기본 코드를 구현하여 수행하였다. 실험 결과 제시된 가이드라인을 적용하면 전력 소모 효율이 향상됨을 알 수 있으며, 특히 Reduce Loop Structure 의 경우 20%의 절감 효과가 있는 것으로 나타났다.

<표 3> Loop 코드 가이드라인 기법 적용 결과

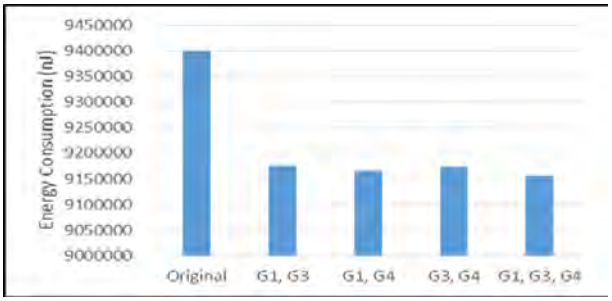
Techniques	Energy (nJ)		Difference (%)
	Before	After	
Index Elimination	4939	4819	2.43
Reduce Loop Structure	945508	753183	20.34
Global Variables	6967	6590	5.41
Extract Expression	5973	5548	7.12

Huffman Code 에 대한 분석 결과 Queue 클래스에 가이드라인 G1, encoder 클래스에 G2, heap 생성 클래스에 G4 을 적용할 수 있음을 발견하였으며, 각 가이드라인을 적용한 경우의 전력 소모량은 (그림 5)와 같다. 실험 결과, 각각의 가이드라인을 적용한 경우 소모전력량이 감소하는 것으로 나타났다.



(그림 5) Huffman Code 에 대한 가이드라인 별 소모전력 실험 결과

가이드라인을 동시에 적용했을 경우의 전력 소모량 변화를 분석하기 위해, 가이드라인 조합에 따른 소모전력 분석 실험을 수행하여 (그림 6)과 같은 결과 값을 얻었다. 각 가이드라인은 프로그램 코드의 서로 다른 부분에 적용되었으나, 가이드라인을 개별 적용한 경우의 절감 값의 합이 아닌 평균 값에 가까운 절감 효과를 얻는 것으로 분석되었다.



(그림 6) Huffman Code 에 대한 가이드라인 조합 별 소모전력 실험 결과

6. 결론

소프트웨어의 loop 코드를 대상으로 코드 구조 변환에 따른 소모 전력을 분석하고, 이를 기반으로 loop 코드 구현 과정에서 전력 효율 향상을 위한 가이드라인을 제시하였다. 제시된 loop 코드 구조 변환 기법과 가이드라인에 따라 소프트웨어 코드 구현 과정에서 저전력 요구사항을 반영할 수 있을 것으로 기대된다.

향후 연구에서는 제시된 기법 이외에 전력 효율 향상 효과를 얻을 수 있는 기법을 추가로 연구하고, 다양한 loop 코드 패턴 분석을 통해 전력 효율 향상을 위한 loop 리팩토링 연구를 수행하여, 본 논문에서 제시한 기법과 가이드라인의 효용성을 높이고자 한다.

Acknowledgment

이 논문은 정부(교육부)의 재원으로 한국연구재단-일반연구자지원사업의 지원을 받아 수행된 연구임 (No. NRF-2014R1A1A4A01005566).

참고문헌

[1] T. K. Tan. Et al., “Software Architectural Transformation: A New Approach to Low Energy Embedded Software,” Proceeding of Design Automation & Test in Europe, 2003.

[2] 김동승, 최성운, 윤성로, “저전력 정렬 알고리즘 설계”, 제 35 회 한국정보처리학회 춘계학술대회 논문집 제 18 권 1 호, pp. 123-126, 2011.

[3] H. Jun, et al., “Modeling and Analysis of Power Consumption for Component-Based Embedded Software,” Proc. Embedded Ubiquitous Computing Workshops 2006, pp. 795-804, 2006.

[4] Gottschalk, Marion, et al. "Removing Energy Code Smells with Reengineering Services," Proc. of GI-Jahrestagung. 2012.

[5] Brandolese, Carlo, et al. "The impact of source code transformations on software power and energy consumption," Proc. of Journal of Circuits, Systems, and Computers, 2002.

[6] Z. Herczeg, D. Schmidt, and et al., “Eergy simulation of embedded XScale systems with XEEMU,” Journal of Embedded Computing, pp. 209-219, August, 2009.

[7] David A. Huffman, “A method for the construction of minimum redundancy codes,” Resonance, Vol. 11, Issue 2, pp. 91-99, 2006.