실시간 센서 운영체제를 위한 함수 아웃라이닝 기법

이상호[O1] 민홍[1] 김봉재[1] 김석현[1] 조유근[1] 홍지만[2]
서울대학교 컴퓨터공학부[1]
숭실대학교 컴퓨터학부[2]
{shyi[O],hmin,bjkim,shkim,cho}@os.snu.ac.kr[1], jiman@ssu.ac.kr[2]

Function Outlining for Real-time Sensor Operating Systems

Sangho Yi[O1] Hong Min[1] Bongjae Kim[1] Surkhyun Kim[1] Yookun Cho[1] Jiman Hong[2]
School of Computer Science and Engineering, Seoul National University[1]
School of Computing, Soongsil University[2]

요 약

컴퓨터 시스템의 등장 이래로, 함수 인라이닝 기법은 함수 지향형 프로그래밍 언어에서 코드 크기의 증가와 함께 실행 시간을 감소시키는 하나의 기법으로 사용되어왔다. 이에 반하여, 함수 아웃라이닝 기법은 실행 시간을 증가시키지만 코드 크기의 감소를 가져온다. 기존 범용 컴퓨터 시스템은 코드를 저장하는 저장장치의 크기에 큰 제약이 없었기 때문에, 함수 아웃라이닝 기법은 그리 널리 쓰이지 않았다. 그러나 최근의 무선 센서 네트워크 분야의 연구를 통하여 범용 컴퓨터 시스템에서는 찾아보기 어려웠던 코드 영역의 자원 제약이 심화되었고, 이에 따라 함수 아웃라이닝 기법이 보다 의미를 갖게 되었다. 특히, 실시간 응답을 요구받는 센서 운영체제는 코드 공간의 제약 하에서 실시간 작업 처리를 수행할 수 있어야 한다. 본 논문에서는, 함수 아웃라이닝 기법을 통하여 실시간 센서 운영체제의 여유 시간(laxity time)을 활용하면서 코드의 크기를 감소시키는 방법을 제안한다. 이를 통하여, 코드 공간의 제약 상황을 보다 완화할 수 있다.

1. 서론

최근에 등장한 무선 센서 네트워크는, 각 컴퓨팅 단말인 센서 노드를 저비용으로 설계하도록 강제하고 있다. 그림 1은 센서 노드의 제약 상황을 보인다. 이러한 센서 노드와 범용 컴퓨팅 환경과 비교하자면, 메모리 공간의 측면에서는 약 1,000,000배 이상의 차이가 존재한다. 다시 말해, 프로그램을 탑재하는 코드 영역의 자원 제약이 상당해지고, 이 결과로 기존에 연구되지 않던 함수 아웃라이닝 기법이 쓸모 있는 기술로 탈바꿈하게 된다.

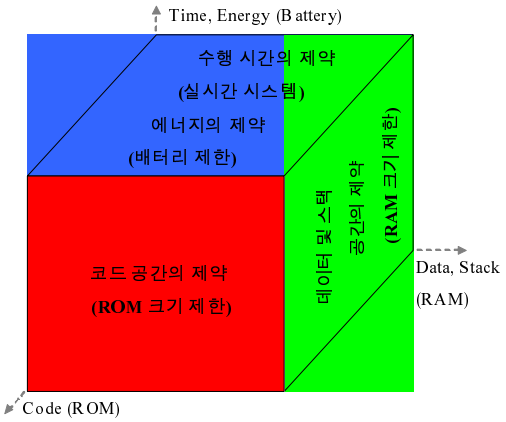

그림 1. 무선 센서 노드상의 자원 제약

무선 센서 네트워크는 상당수가 실시간 응답을 필요로 한다. 예를 들면, 산불 감지 시스템, 교량 진동 감지 시스템, 주거 환경 탐지 시스템 등의 다양한 실시간 응답이 중요한 센서 응용 사례가 존재하며, 이러한 경우에는 센서 노드에 탑재된 센서 운영체제는, 각 작업에 대한 실시간 작업 처리를 보장해야 한다. 함수 아웃라이닝[1] 기법의 사용을 통한 실행 시간의 증가 역시 적절한 수준에서 제한되어야 할 것이다.

본 논문에서는, 자원 제약하의 실시간 센서 운영체제를 위한 함수 아웃라이닝 기법을 제안한다. 이 기법은 실시간 센서 운영체제의 여유 시간(laxity time)을 활용하면서 코드의 크기를 감소시킨다. 제안한 기법을 사용하면, 각 센서 노드에서 큰 제약 요소가 되는 코드 공간에 대한 사용량을 보다 줄일 수 있다.

2. 실시간 센서 운영체제를 위한 함수 아웃라이닝 기법의 설계

본 논문에서는 위의 수행 시간 및 코드 크기에 대한 요구사항을 단 하나의 파라미터로 나타낸다. 먼저, 하나의 함수를 아웃라이닝 한다고 가정한다. 이때, 수행 시간의 증가가 적다면 이 함수는 아웃라이닝 되기에 좋은 함수가 된다. 또한, 코드 크기의 감소가 크다면, 이것 역시 아웃라이닝의 대상이 된다. 따라서 다음의 파라미터 D로 이 코드 크기와 수행 시간의 관계를 나타낼 수 있을 것이다.

D = 코드 크기 감소량 / 수행 시간의 증가량;

그림 2는 본 논문에서 사용한 WHIRL 트리 구조[2]의 설명을 위하여, 예제 코드와 이때의 WHIRL 트리를 간략히 보인다. WHIRL은 ORC(Open Research Compiler) 프로젝트 팀에서 사용하는 중간 언어 표현용 자료 구조이다.

WHIRL은 실제로 C컴파일러 등을 구현할 때에 사용되는 언어 기술용 자료 구조의 하나이다. 이 그림에서, switch문, case문, if문, while문, 그리고 그 외의 코드들이 WHIRL 트리에 추가되어있는 모습을 볼 수 있다.

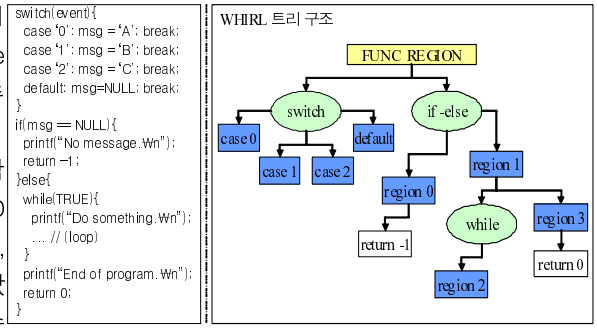

그림 2. 예제 코드에 대한 WHIRL 트리 구조

제안 기법은, 앞 절에서 표기한 파라미터 D를 바탕으로 각 코드 루틴의 아웃라이닝을 할 것인가를 결정한다. 파라미터 D는 아웃라이닝을 수행하였을 때의 코드 크기 감소량이 크면서, 동시에 수행 시간의 증가량이 작은 코드 루틴에 대하여 큰 값을 갖게 된다. 다음의 알고리즘 1은 본 논문에서 사용한 함수 아웃라이닝 기법의 수행 절차를 보인다.

알고리즘 1. 실시간 시스템을 위한 함수 아웃라이닝 기법

– C언어 소스코드의 컴파일 시점에 다음의 과정을 수행–

```
1 Create WHIRL tree
2 Calculate Di for all terminal node i in WHIRL tree
3 Sort Di in descending order
4 for all Di (i=1 to n)
5    Do Function Outlining
6    if laxity_time <= 0 then
7       break;
8    end if
9 end for
```

본 논문에서 제안하는 함수 아웃라이닝 기법은 먼저, WHIRL 트리로 각 코드 루틴을 분석한 후, 각 코드 루틴에 대한 D를 계산한다. 코드 크기의 감소량 및 수행 시간의 증가량은, 컴파일 후의 바이너리를 분석하거나 사전에 수행시켜보면 알 수 있다. 본 논문에서는 이와 같은 방식으로, 시스템의 여유 시간을 최대한 활용하면서 동시에 코드 크기를 감소시키는 방법을 적용하였다.

3. 성능 평가

다음의 그림 3과 4는 제안 기법을 사용하였을 때와, 그렇지 않을 때의 수행 시간 및 코드 크기의 변화를 보이고 있다.

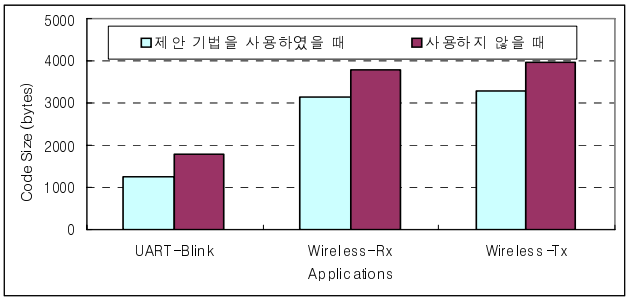

그림 3. 제안 기법을 사용하였을 때의 코드 크기의 변화

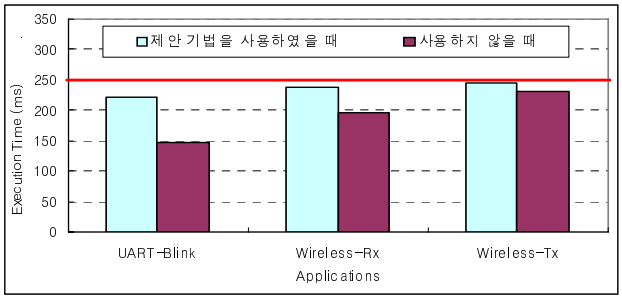

그림 4. 제안 기법을 사용하였을 때의 수행 시간의 변화

그림 3의 결과를 바탕으로, 제안 기법을 사용하면 약 30% 정도의 코드 크기의 절감 효과를 얻을 수 있음을 확인하였다. 이는 각 예제의 코드 구성에 따라 달라지는데, 동일한 연산을 자주 수행하는 경우에는 아웃라이닝이 코드 크기의 절감을 위하여 큰 효과를 얻을 수 있었다. 그림 4에서 적색선으로 표시한 250ms는 본 논문의 실험에서 가정으로 사용한 각 응용의 마감 시한이다. 이 결과에서 보이는 바와 같이, 제안 기법을 사용하면 각 주기별 응용의 실행 시간이 마감 시한을 넘기진 않지만 기존 기법에 비해 약간 더 사용하는 것을 볼 수 있다. 실시간성을 중요시하는 무선 센서 네트워크 및 시스템에서는 마감 시한 내에서 최대한 자원을 효율적으로 사용하는 것이 중요하므로, 제안한 기법은 코드 영역의 사용량을 줄여야 하는 실시간 임베디드 시스템 등에 적용될 수 있을 것이다.

5. 결론

본 논문에서는, 함수 아웃라이닝 기법을 통하여 실시간 센서 운영체제의 여유 시간을 활용하면서 코드의 크기를 감소시키는 방법을 제안하였다. 본 논문의 성능 평가 결과를 통하여, 제안한 기법을 사용하면 센서 운영체제의 실시간 응답을 보장하면서, 동시에 코드의 크기를 상당히 감소시킬 수 있음을 보였다.

[참고문헌]

[1] P. Zhao, J. N. Amaral, Function Outlining and Partial Inlining, The 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'05), 2005.

[2] WHIRL 트리 - ORC, http://ipf-orc.sourceforge.net/