실시간 내장형 S/W의 성능분석을 위한 Control Flow Graph 추출

*황요섭°, *안성용, *이정아, **심재홍
*조선대학교 컴퓨터공학부,**조선대학교 인터넷소프트웨어 공학부
E-mail :{bluewind°, dis, jeong}@rain.chosun.ac.kr, jhshim@chosun.ac.kr

Control Flow Graph Extraction for Performance Analysis of Real-Time Embedded Software

*Yoseop Hwang°, *Seongyong Ahn, *Jeonga Lee, **Jaehong Shim
*Dept. of Computer Engineering, **Dept. of Internet Software Engineering, Chosun Univ

요 약

최근 반도체 설계 및 생산 공정의 급속한 발달로 내장형 시스템이 대중화되는 추세이고 비용이나 제품 출시 기간에 있어서 내장형 소프트웨어는 중요한 하나의 요소로 대두되고 있다. 내장형 시스템은 일반 PC와는 다르게 메모리 크기, 전력 소비, 신뢰성, 사이즈, 비용 등과 같은 제약사항들을 내포하기 때문에 제한된 자원의 효율적인 이용과 소프트웨어의 최적화를 위해 소프트웨어의 성능을 분석하기 위한 필요성이 대두된다. 본 논문에서는 소프트웨어 성능분석 도구인 'Cinderella'를 확장하기 위하여 현재 가장 널리 사용되고 있는 이진 실행 파일인 ELF파일에서 성능을 측정하기 위한 기본 요소로서 Control flow graph를 추출하기 위한 알고리즘을 제안한다. 본 논문에서 제안한 알고리즘은 향후 ARM기반의 머신에서 ELF 파일의 내장형 소프트웨어의 시간분석에 필요한 요소이다.

1. 서 론

최근 IT 기술은 반도체 공정기술의 발전에 힘입어 내장형 시스템이 저가, 소형화, 고성능화 됨에 따라 제품 경쟁력의 핵심이 H/W 생산 기술에서 S/W 최적화 기술로 이동하는 추세이다.

내장형 시스템은 주어진 시간 안에 작업들을 완료해야 하는 시간 제약성을 가지는데 비용, 신뢰성, 전력 소비, 사이즈 등도 영향을 줄 수 있다. 특히 시스템에 사용되는 소프트웨어는 가장 큰 영향을 미치는데 똑같은 기능을 수행하는 소프트웨어라 하더라도 어떻게 프로그램 했느냐에 따라 코드 사이즈라든지, 성능 면에서 많은 차이가 생겨날 수 있다. 즉 내장형 소프트웨어의 성능을 분석하기 위한 측정 및 예측 도구가 필요하다.

내장형 소프트웨어의 경우에는 이의 성능을 예측하기 위해 소프트웨어 컴포넌트의 성능 수행시간을 정확히 측정하기 위한 고성능 시간 분석에 관한 연구들이 진행되고 있다. 이러한 연구들은 명령어 수준에서 성능을 분석하는 경우가 많다. 마이크로 프로세서에서 Control flow graph를 추출하기 위한 연구는 특정 어플리케이션의 성능을 예측할 수 있게 한다[5,7]. 다중처리가 가능한 내장형 시스템의 성능 분석을 위한 연구는 여러 개의 작업들이 각각 다른 프로세서에서 분산 처리되는 경우에 대하여도 성능분석을 가능하게 한다[6].

위와 같은 소프트웨어 분석을 위한 기술들은 3가지 한계성을 가지고 있다. 첫째, 실제 응용에서 등장하는 복잡한 프로그램은 분석할 수 없다. 둘째, 캐쉬 메모리나 파이프라인을 지원하는 프로세서를 모델링하여 실행시간을 분석하는 어려움이 아직도 있다. 셋째는 새로운 하드웨어 플랫폼을 쉽게 재구성할 수 없다는 제약점을 가지고 있다.

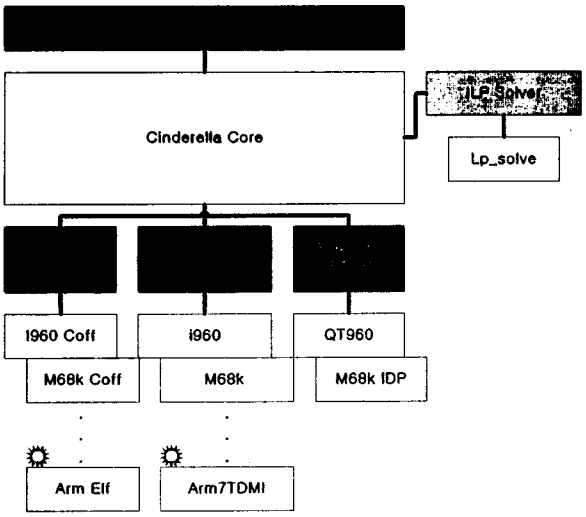
본 논문에서는 내장형 소프트웨어를 분석하기 위한 성능 측정 도구로서 위와 같은 한계를 고려한 'Cinderella'라는 시간 분석도구를 활용하는 방안을 제시한다[4]. 'Cinderella'는 이진 실행 파일을 읽어들여 하드웨어 플랫폼을 선택한 후 목적파일에서 바이트 명령어들을 읽어들여 CFG(Control Flow Graph)를 추출함과 동시에 기본블록과 순환영역(loop bound)의 반복 횟수, 분기명령어의 관계를 노드(node)와 연결선(edge)이라는 정보를 사용하여 어플리케이션의 수행시간을 측정한다. 이 도구는 Unix의 이진 실행 파일 포맷인 COFF(Common object file format)에서 CFG를 추출하여 성능을 측정한다. 하지만 현재 사용되는 대표적인 이진 실행 파일 포맷인 ELF(Executable and Linkable Format)는 지원되지 않는다[2]. 본 논문에서는 ELF파일을 고려한 'Cinderella'의 확장과 ELF파일에서 디버깅 정보(Dwarf)를 이용하여 CFG를 추출하는 방법을 제안하고자 한다[1,3].

본 논문의 구성은 2장에서 소프트웨어 성능분석 도구인 'Cinderella'의 소개 및 구조와 기능에 대해 서술하고, 3장에서 'Cinderella'를 확장하기 위한 ELF파일과 디버깅 정보 파일에 대해, 4장에서 프로그램 경로를 분석하기 위한 ELF에서의 CFG추출 방법을, 5장에서는 결론 및 향후 연구과제를 제시한다.

2. 소프트웨어 성능 분석도구(Cinderella)

'Cinderella'는 ILP(Integer Linear Programming)기반으로 되

어있는 내장형 소프트웨어 성능분석도구로써 정해진 프로세서에서 프로그램이 멈추지 않고 돌아갈 때 실행할 수 있는 수행시간의 최적실행시간(Best case execution time)과 최악실행시간(Worst case execution tim)을 찾는 것을 필요로 한다[8]. 'Cinderella'는 추가적으로 모듈을 첨가할 수 있는 재구성 가능한 프레임웍(retargetable framework)을 고려한 확장성 있는 구조를 가지고 있는데 그림 1에서 보는 것처럼 'Cinderella' 코어 부분, Object file 모듈 부분, Instruction set 모듈 부분, Machine 모듈 부분, ILP Solver 부분, GUI(Graphical User Interface)부분등 크게 6개의 블록으로 구성되어 있다. 확장을 고려할 수 있는 부분들은 Object file 모듈 부분, Instruction set 모듈 부분, Machine 모듈 부분을 수정함으로써 가능하다.

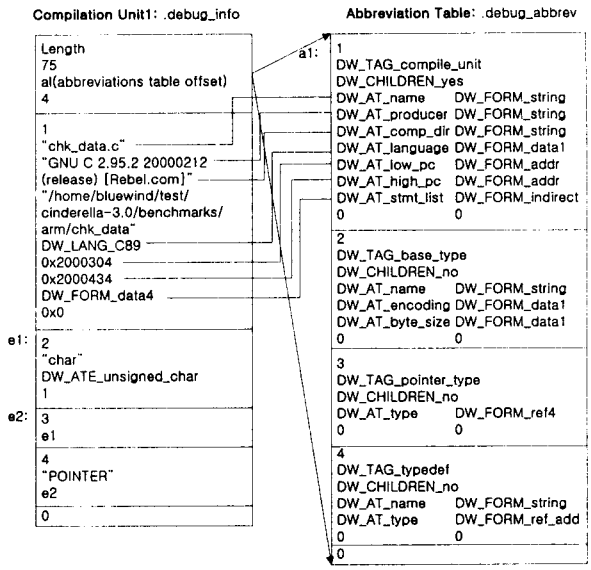


(그림 1) Cinderella 구조의 블록 다이어그램

3. ELF에서 CFG를 추출하기 위한 디버깅 정보(DWARF)

CFG를 이진 실행 파일 즉 목적 파일에서 추출하기 위해 필요한 부분은 바이너리 인스트럭션이 있는 부분인 목적코드(object code)와 소스 파일의 라인 번호나 지역 심볼들의 정보를 가지고 있는 디버깅 정보(debugging info)이다.

ELF는 DWARF(Debugging With Attribute Record Format)라고 하는 디버깅 포맷을 가지고 있다. 이 파일 포맷은 TIS(Tool Interface Standards)에서 디버깅에 관계된 표준을 제시한 것으로, DWARF는 컴파일러, 어셈블러, 링크 에디터에 의한 심볼릭들과 소스레벨 디버깅을 위해 필요한 정보 포맷을 제공한다. 디버깅 정보 포맷은 특정 컴파일러나 디버거들에 편파적이지 않다. DWARF의 목적은 이전 호환성을 유지하면서 다른 언어로 쉽게 확장할 수 있는 형태로, 어떤 디버거든 소스 프로그램의 정확한 모양을 전달할 수 있도록 정보를 제공한다. DWARF DIE들(Debugging Information Entries)은 ELF파일에서 .debug 섹션을 형성한다.(그림2) 고정된 크기의 작은 정보들 대신에 DWARF DIE들은 각각 임의의 길이를 갖는 복잡한 속성들을 포함하고 있으며 영역별로 프로그램 데이타의 트리구조로 씌여져 있다.

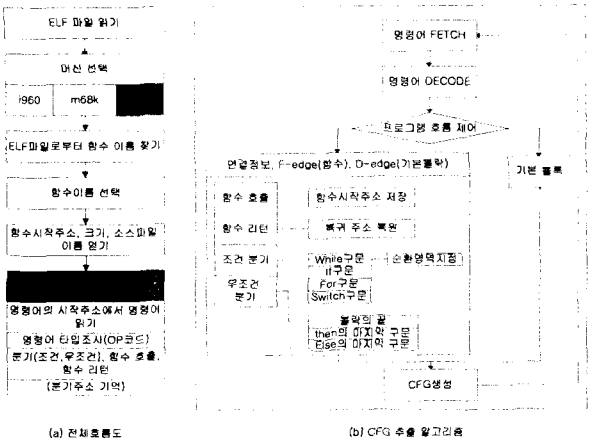


(그림 2) .debug_info와 .debug_abbrev섹션의 구조

4. CFG를 추출하기 위한 알고리즘

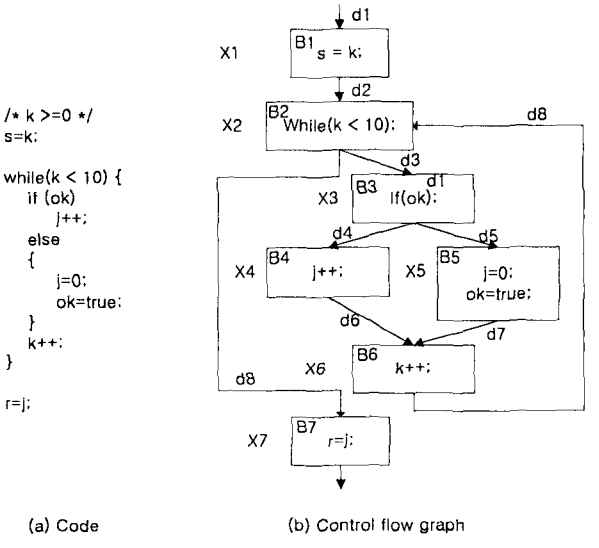
이진 실행 파일에서 CFG를 추출하기 위해 소스 프로그램을 컴파일 할 때 디버깅 옵션을 체크한다. ELF파일에서는 따로 디버깅 정보를 제공하지 않기 때문인데 디버깅 정보는 CFG를 추출할 때 꼭 필요한 내용들을 가지고 있다. CFG를 생성하기 위해 프로그램 명령어 코드가 존재하는 위치를 알아야 한다. 프로그램 명령어 코드들은 섹션헤더의 .text에 존재하게 되는데 .text 섹션에서 함수의 시작주소를 알아야 이진 실행 파일에서 명령어를 파싱함으로써 분기명령이라든지 함수 호출, 함수 반환(function call,function return)같은 제어 흐름을 알 수 있다.

CFG는 분기(조건 분기, 무조건 분기)명령과 함수호출, 함수 반환과 같은 프로그램의 제어 흐름을 생성하기 위한 그래프이다. 디버깅 섹션에서는 여러 가지 정보를 제공하는데 소스 프로그램에서 각 함수들의 이름과 라인번호, 메모리상에 로드되는 가상주소, 파일에서의 위치(offset), 크기등 CFG를 추출하기 위한 세부 정보를 가지고 있다. 이와 관련된 부분은 .debug 섹션들에서 추출할 수 있는데 .debug_info 섹션은 컴파일 유닛들에 대한 내용을 표현하는 섹션으로 offset/name의 형태로 정보들을 표시하며 소스파일의 이름, 컴파일 디렉토리, 라인번호, 프로그램의 시작주소와 끝나는 주소의 정보를 가지고 있다. .debug_abbrev 섹션은 .debug_info에서 생략형으로 되어 있는 부분을 알기 위한 섹션이다(그림 2). ELF파일에서 CFG를 추출하기 위한 전체흐름은 그림 3의 (a)와 같으며, CFG를 추출하는 알고리즘은 그림 3의 (b)와 같다.



(그림 3) 전체흐름도와 CFG를 추출하기 위한 알고리즘

CFG는 명령어를 읽어서 파싱했을때 프로그램 제어문(조건 분기, 무조건분기, 함수 호출, 함수 리턴)이 아니면 기본 블록으로 노드(node)를 형성한다. 명령어가 프로그램 제어일때는 연결정보라는 연결리스트를 사용함으로써 각 노드 간의 연결성을 알 수 있도록 F-edge(함수정보)와 D-edge(블록정보)를 사용하여 표현한다.



(그림 4) 소스코드와 CFG

그림 4는 c소스코드에서 CFG를 표시한 것이다. B는 기본 블록을 의미하고 D는 블록들간의 연결정보, X도 기본블록을 의미하는데 반복횟수를 명시하는 제약조건을 표기한다. X2에서 X7사이의 순환영역이(loop bound) 얼마나 반복될 것인가는 사용자가 직접 최소값과 최대값을 입력함으로써 수행 시간의 최적실행시간과 최악실행시간을 계산할 수 있다.

5. 결론 및 향후 연구방향

본 논문에서는 내장형 소프트웨어의 성능분석을 위하여 현재 가장 널리 사용되고 있는 ELF파일을 DWARF를 이용하여 분석하였으며 실행코드를 기본블록으로 나누고 각각의 블록에 관계된 연결정보를 노드(node)와 연결(edge)로 표현함으로써 CFG를 추출하기 위한 알고리즘을 제안하고 구현하였다.

구현된 도구는 ELF 이진 실행 파일에서 어플리케이션의 성능을 계산하기 위한 요소들을 추출하기 위해 CFG를 생성한다. CFG는 기본블록을 노드로 기본블록을 연결하는 정보들은 연결정보를 사용하여 표시하는데 순환영역의 반복횟수는 사용자가 최소값과 최대값을 입력함으로써 어플리케이션의 최적 실행시간과 최악 실행시간을 계산할 수 있다. 시간 분석을 완료하기 위하여 앞으로는 ARM머신을 위한 명령어 모듈과 머신모듈을 구현하는 작업이 필요하다.

본 논문에서 구현된 CFG추출은 ELF 이진 실행 파일의 내장형 소프트웨어의 성능을 측정하기 위한 방법으로 어플리케이션의 성능을 미리 예측 및 측정함으로써 내장형 시스템의 제약사항들을 만족하면서도 성능을 높일 수 있는 최적의 해결책을 찾는데 활용될 수 있을 것으로 기대된다.

6. 참고문헌

[1] Theiling, H., "Extracting safe and precise control flow from binaries", Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh International Conference on , 2000

[2] TIS Committee, Executable and Linking Format (ELF) Specification Ver.1.2, May 1995

[3] TIS Committee, Debugging Information Format (DWARF) Specification Ver 2.0, May 1995

[4] Seong-Yong Ahn, Jea-Hong Shim, Jeong-A Lee, "A Design and Implementation of a Timing Analysis Simulator for a Design Space Exploration on a Hybrid Embedded System", The KIPS Transactions, Dec., 2002

[5] Kaiyu Chen, Sharad Malik, David I. August, "Retargetable Static Timing Analysis for Embedded Software," Proceedings of the International Symposium on System Synthesis (ISSS), October, 2001

[6] Ti-Yen Yen, Wayne Wolf, "Performance Estimation for Real-Time Distributed Embedded Systems," IEEE Transactions onn Parallel and Distributed Systems, Vol. 9, 11, November 1998

[7] George Hadjiyiannis, Pierro Russo, Srinivas Devadas, "A Methodology for Accurate Performance Evaluation in Architecture Exploration" Design Automation Conference 1999

[8] Yau-Tsun Steven Li, Sharad Malik, "Performance analysis of Real-Time Embedded Software," Kluwer Academic Publishers, 1999