TMO기반 프로그램을 위한 소스코드 분석 및 분해

이재석, 신원, 김태완, 장천현
건국대학교 컴퓨터공학과
e-mail:{skybreez, wonjjang, twkim, chchang}@konkuk.ac.kr

Analysis and Classification of Source Code for TMO-Based Programs

Jae-Seok Lee, Won Shin, Tae-Wan Kim, Chun-Hyon Chang
Dept of Computer Engineering, Konkuk University

요 약

실시간 시스템에서는 정해진 시간 안에 작업을 수행해야 하는 것이 가장 중요하다. 때문에 실시간 시스템에서의 응답시간 위반은 물질 또는 인명 피해와 직결된다. 이에 응답시간을 보장하기 위해 실시간 시스템을 분석하는 기법들에 대한 많은 연구가 진행되었다. 그러나 기존의 분석 방법들은 최악실행시간을 도출하기위해 실시간 프로그램의 흐름을 분석하거나 분석을 위한 제약을 생성할 때 부하가 생기는 문제점을 가지고 있다. 이러한 문제를 해결하기 위하여 본 논문에서는 프로그램에서 나타나는 함수 또는 변수 등이 중복해서 사용되는 특성들을 이용하여 실행시간 분석에 대한 부하를 줄일 수 있는 방법을 제안하고, 제안한 방법을 기반으로 실시간 프로그램에서 실행시간을 예측할 때 필요한 기본 자료들을 도출할 수 있는 소스코드 분석 도구를 제안한다.

1. 서론

오늘날 실시간 시스템은 소형 임베디드 시스템에서부터 대형 분산처리 시스템까지 다양한 분야에서 사용된다. 실시간 시스템이란 주어진 작업을 정해진 시간 안에 수행할 수 있는 환경을 제공하는 시스템을 의미한다. 실시간 시스템에서 가장 중요한 것은 제한된 시간 안에 일을 처리해야하는 것이며, 응답시간을 위반하였을 때 인적, 물질적인 피해와 직결되기 때문에 응답시간 보장은 더욱 중요하다. 개발자는 응답시간을 고려하여 실시간 시스템을 개발하지만 위반하는 경우가 많다. 이 응답시간 위반 문제는 개발자의 실시간 개념이 부족하거나 시스템의 시간 지연 요소에 의해 발생한다. 그래서 개발자들은 데드라인을 막기 위해 실시간 프로그램을 분석하여 응답시간을 확인하는 작업에 시간을 낭비한다. 그래서 실시간 시스템은 응답시간을 보장하는 프로그램에 대한 연구가 계속되고 있다. 여러 연구 결과 중에 응답시간을 보장하며, 시간 처리의 자유로움을 보장하는 TMO (Time - triggered Message - triggered Object)모델을 사용하여 실시간 시스템의 안정성을 높일 수 있게 되었다. 그러나 TMO모델도 실행시간의 정의는 개발자에 의해 이루어지기 때문에 정의된 실행시간의 정확성 여부를 확인하는 작업에는 여전히 어려움이 있다. 그러므로 실시간 시스템의 정확한 실행시간의 도출이 필요하다.

이에 본 논문에서는 실시간 프로그램에서 실행시간의 정확성 여부를 확인하기 위한 실행시간 분석기의 기본 자료들을 도출하는 정적 분석 도구를 제안한다. 2장에서는 관련연구인 TMO와 정적 분석과정에 대하여 설명한다. 3장에서는 소스코드 분석 도구인 SCAC (Source code Analyzer & Classification)의 개념과 전체 구조들을 설명한다.

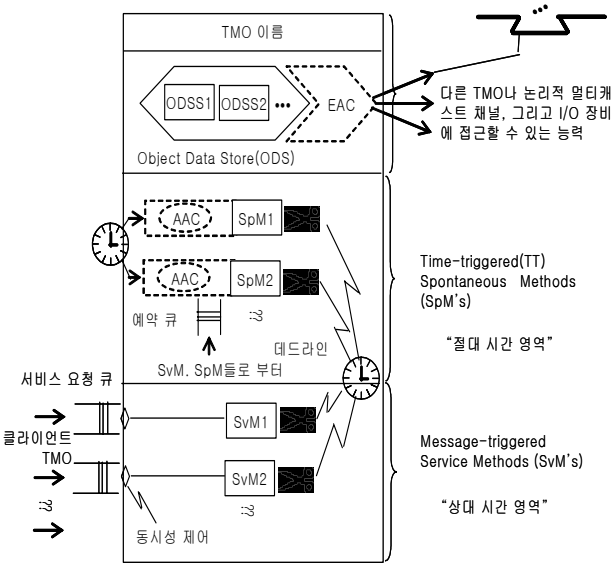
2. 관련연구

2.1 TMO

TMO 모델은 적시성 서비스 기능을 디자인 단계에서부터 보장하고, 실시간 시스템이 가지는 시간적인 행동, 메시지에 의한 기능적인 행동에 대한 추상

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

화 등을 지원한다. TMO모델은 실시간 시스템의 시간적 행동을 추상화하기 위한 SpM(Spontaneous Method), 메시지에 의한 행동을 추상화하기 위한 SvM(Service Method), 그리고 이들이 공유하는 데이터를 세그먼트 단위로 저장하기 위한 ODSS (Object Data Store Segments)등으로 구성되어 있고, 이들을 TMO라는 하나의 객체로 모델링 할 수 있게 해준다. TMO의 시간적 행동은 AAC에서 정의되고, 정의된 주기와 제한시간을 가지고 반복 실행되는 SpM에 의해 SvM이 동작한다. (그림 1)은 TMO객체 모델의 기본 구조를 나타낸다[2].



(그림 1) TMO 객체 모델의 기본 구조

TMO는 적시성보장 컴퓨팅(Timeliness Guaranteed Computing)을 목표로 제안된 실시간 객체 모델로 경성/연성 실시간 응용뿐만 아니라 일반적인 응용 프로그램에도 사용할 수 있으며, 설계 시 시간 보장 개념을 제공한다.

2.2 기존 정적 분석 과정

기존의 실행시간 분석 도구들은 실행시간을 도출하기 위해 프로그램 흐름을 분석한 후 영향요소를 적용하고, 실행시간 분석 기법을 적용하여 실행시간을 도출한다.

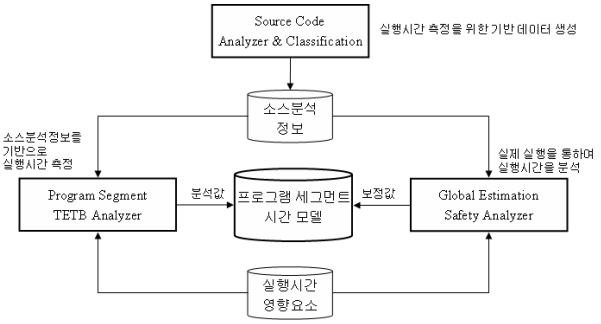
프로그램 흐름 분석 단계는 가능한 프로그램의 실행경로를 결정하는 단계이다. 프로그램 흐름에 대한 정보를 추출하기 위한 방법론으로 기존에는 사용자가 직접 입력하는 방법을 사용하였고, 최근에는 이를 자동화하기 위한 연구가 많이 진행되고 있다.

영향요소 적용 단계는 하드웨어, 운영체제 등 실행시간에 영향을 줄 수 있는 파이프라인, 캐시 등과 같은 실행시간 영향 요소들을 분석하는 단계이다.

실행시간 분석 기법 적용 단계는 위의 두 단계에서 나온 데이터를 기반으로 최악시간을 분석하는 단계로서 경로기반, IPET기반, 트리기반 분석 기법으로 나뉜다. 그러나 기존의 분석기법들은 최악 실행시간 도출을 위해 경로기반 분석 기법에서는 실행가능한 모든 경로를 탐색해야하고, IPET기법에서는 가능한 논리 제약을 생성하기 위한 많은 연산을 수행해야 하기 때문에 많은 부하가 발생한다[3]. 이와 같이 분석과정에서 생기는 부하는 실시간 프로그램의 실제 실행시간 값을 도출할 때 영향을 미쳐 실행시간 분석의 신뢰성을 떨어뜨린다. 또한, 기존의 정적 분석기는 프로그램 흐름 분석과 영향요소, 그리고 실행시간 분석 기법들에 의해서만 실행시간을 도출하기 때문에 분석된 실행시간은 신뢰성이 떨어진다. 이에 본 논문에서는 기존의 정적 분석 과정에서 실행시간 분석 기법들의 단점들을 보완한 신뢰성 분석 도구의 구조가 제안되었다.

2.3 신뢰성 분석 도구

신뢰성 분석 도구는 개발자가 작성한 TMO기반의 프로그램 코드를 입력받아 각각의 작업에 대한 실행시간을 분석하여 개발자에게 알린다. 신뢰성 분석 도구는 크게 3단계로 나뉜다. 실행시간 분석을 하기 위한 기반 정보들을 프로그램 코드를 이용하여 자동으로 생성하기 위한 도구인 SCAC, SCAC의 분석 정보를 기반으로 각각의 작업에 대한 실행시간을 분석하는 PTETBA(Program Segment Tight Execution Time bound Analyzer), 실제 실행을 통하여 PTETBA의 결과인 Program Segment Timing Model에 대한 보정 과정을 진행하는 GESA(Global Execution Safety Analyzer)이다. (그림 2)는 신뢰성 분석 도구의 전체구조를 나타낸다[1].



(그림 2) 신뢰성 분석 도구의 전체구조

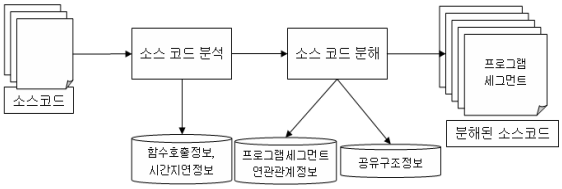
본 논문에서는 소스코드를 분석 및 분해하여 기존의 정적 분석 과정에서 생기는 부하를 방지할 수 있

는 구조인 SCAC에 대하여 기술한다.

3. SCAC

SCAC는 실행시간을 분석하기 위해 기반이 되는 자료를 도출하기 위한 도구로서 기존의 정적 실행시간 분석 단계 중 흐름분석 단계를 자동화한다.

소스코드를 분석하는 단계는 크게 소스코드 분석 단계와 소스코드 분해 단계로 나뉜다. 소스코드 분석 단계에서는 프로그램의 흐름이나 명시된 시간 값 등을 도출하고, 소스코드 분해 단계에서는 소스코드 분석 단계에서 나온 정보를 기반으로 소스코드를 분해 및 재구성한다. (그림 3)은 SCAC 구조도이다.



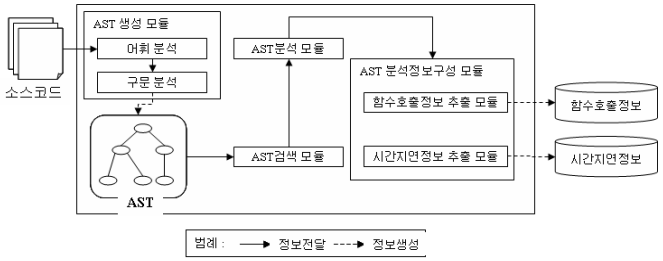
(그림 3) SCAC 전체 구조도

SCAC의 결과물로서 프로그램 세그먼트와 공유구조정보가 있다. 프로그램 세그먼트는 소스코드를 하나의 주기나 함수간의 흐름별로 분리하여 독립적으로 컴파일 할 수 있게 저장한 구조이고, 공유구조정보는 프로그램 세그먼트를 도출하는 과정에서 생기는 공유 함수 및 전역 변수, 공유 데이터 등의 정보를 저장한 구조이다. 이 정보들은 기존의 실행시간 분석 기법이 가지고 있는 분석 과정에서 생기는 부하문제를 해결하기 위해 제안되었다.

소스코드를 프로그램 세그먼트로 분리할 수 있는 것은 TMO 기반 실시간 프로그램의 특성 때문이다. TMO 프로그램은 SpM과 같이 실시간 프로그램 특성을 따라 쓰레드들이 동작할 때 흐름 별로 분리가 가능하며, 시간에 의하여 동작하기 때문에 독립적으로 분석이 가능한 구조이다. 이러한 독립적으로 분석이 가능한 구조에서도 공유 함수와 전역 변수의 정보는 분석 시 중복이 된다. 이러한 중복 분석은 기존의 정적 분석과정에서 나온 부하문제와 동일한 것이고, 이를 방지하기 위해 공유 정보들은 분리한다.

3.1 소스코드 분석기 구조

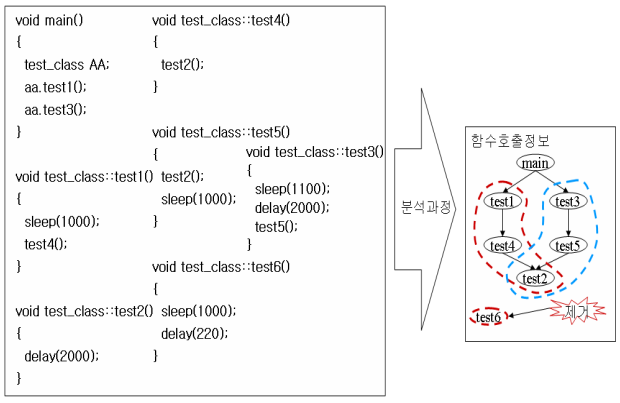
SCAC의 소스코드 분석기는 실행시간 분석 과정에서 기초가 되는 정보인 함수의 호출관계와 소스코드 상에 명시된 시간 정보를 도출하기 위해 제안된 구조이다. 소스코드 분석기의 함수호출정보는 함수간의 호출된 정보이고 시간지연정보는 소스코드 상에 명시된 sleep, delay와 같은 시간이나 TMO의 AAC에서 명시된 시작시간, 종료시간, 실행주기 등의 시간정보이다. (그림 4)는 소스코드 분석기의 구조도이다.



(그림 4) 소스코드 분석기 구조도

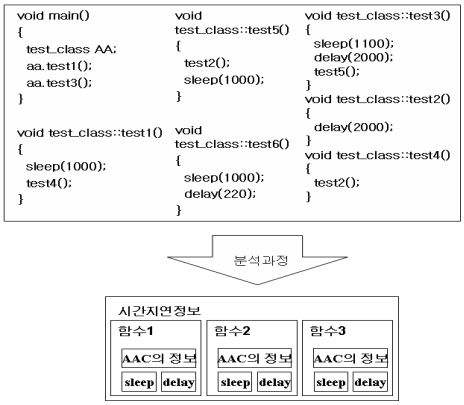
소스코드 분석기는 소스코드를 분석이 쉽고, 분해 및 재구성이 가능한 구조인 AST(Abstract Syntax Tree)형태로 변형한다. AST는 소스코드의 정해진 문법에 맞추어 구성된 트리구조이다.

함수호출정보는 소스코드 상의 명시된 함수정보들을 모두 생성한다. 함수들의 정보는 클래스명, 함수명, 함수를 호출한 정보와 소스의 파일명등이 있으며, 생성된 함수들의 정보는 시간지연정보나 프로그램 세그먼트, 공유구조정보의 기본 자료로 사용된다. 그러나 호출하지 않거나 호출되지 않는 함수들은 프로그램에 영향을 주지 않아 공유구조정보나 프로그램 세그먼트 도출 시에 제거한다. (그림 5)는 함수호출정보의 구성 예이다.



(그림 5) 함수호출정보 구성 예

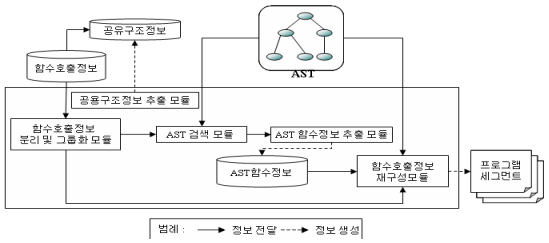
시간지연정보는 소스코드를 분석하는 단계에서 도출된다. 이 시간지연정보는 소스코드 상에 명시된 시간 값인 sleep, delay 시간 값이나 SpM의 AAC 시간 값을 함수호출정보의 함수정보와 결합 시켜 차후 실행시간 분석 도구에서 프로그램의 흐름별로 실행시간을 계산할 때 사용된다. (그림 6)은 시간지연정보의 구성 예이다.



(그림 6) 시간지연정보 구성 예

3.2 소스코드 분해기 구조

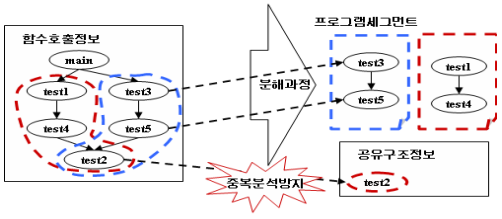
소스코드 분해기는 소스코드 분석기에서 도출한 정보들을 기반으로 프로그램 세그먼트와 공유구조정보를 도출한다. (그림 7)은 소스코드 분해기의 구조도이다.



(그림 7) 소스코드 분해기 구조도

프로그램 세그먼트는 프로그램의 흐름별로 컴파일 될 수 있는 파일의 형태이다. 프로그램 세그먼트의 구성과정은 소스코드 분석기 정보인 함수호출정보를 기반으로 함수정보를 분리한다. 분리한 함수정보 중 함수를 호출한 정보를 이용하여 프로그램 흐름별로 그룹화 한다. 프로그램 흐름별로 묶인 정보들은 함수호출정보에서 생성된 AST 정보들을 이용하여 파일 형태로 생성된다.

공유구조정보는 함수호출정보와 공유 함수명, 전역 변수명 등을 가지고 있다. 공유구조정보의 생성은 함수호출정보의 함수 호출 횟수를 검사하여 하나의 함수가 한번이상 호출하거나 호출되어진 정보들을 이용하여 도출한다. (그림 8)은 프로그램 세그먼트와 공유구조정보의 구성 예이다.



(그림 8)프로그램 세그먼트와 공유구조정보 구성 예

소스코드 분해기에서 재구성된 프로그램 세그먼트와 공유구조정보는 신뢰성 분석 도구의 실행시간을 도출하기 위한 기반 데이터로 사용한다. 프로그램 세그먼트는 프로그램의 흐름별로 실행시간을 도출할 때 사용된다. 그리고 프로그램 세그먼트에서 나온 시간 정보와 시간지연정보의 시간 정보, 공유구조정보의 시간 정보를 결합하여 기존의 실행시간 분석기보다 분석의 부하를 줄이며, 분석할 수 있는 정보들로 사용된다.

4. 결론 및 향후계획

실시간 프로그램은 시간적 정확성을 가져야 한다. 개발자는 시간적 정확성을 고려하여 실행시간을 정의하고 정의한 실행시간을 검사를 수행해야 한다. 응답시간을 보장하는 TMO 모델에서도 개발자의 시간 정보 정의와 실시간성/실행시간의 검증이 필요하고, 이를 위한 실행시간 분석 도구에 대한 연구가 필요하다.

본 논문에서는 TMO의 특성을 이용하여 분석할 수 있는 파일 형태의 구조인 프로그램 세그먼트와 기존의 분석과정에서 생기는 부하문제를 보완하기 위해 공유구조정보 개념을 제안하였다. 또한, 이러한 개념을 이용하여 실시간 프로그램에서 실행시간을 분석 및 측정하기 위한 기본 자료들을 도출할 수 있는 소스코드 분석 도구인 SCAC를 설계 및 구현했다. SCAC를 이용하여 도출한 정보들은 신뢰성 분석 도구의 실행시간 분석과정에서 분석을 위한 기반 데이터로 사용된다.

향후에는 프로그램 세그먼트 연관관계 정보를 구현하기 위한 연구를 진행할 예정이다.

참고문헌

[1] 신원, 김태완, 장천현, “정적 실행시간 분석기의 기반 구조”, 한국 소프트웨어공학 학술대회 논문집 제8권 1호, pp.115-123, 2006

[2] K.H.(Kane) Kim, “A TMO Based Approach to Structuring Real-Time Agents”, In Proc. IEEE ICTAI, pp.165-172, 2002

[3] Jakob Engblom and Andreas Ermedahl, “Comparing Different Worst-Case Execution Time Analysis Methods”, Work-in-Progress session of the 21st IEEE Real-Time Systems Symposium (RTSS 2000), pp.27-30, 2000