자바 기반 멀티스레드 소프트웨어 검증 자동화 방법

김형락 ○ 이욱진

한양대학교 컴퓨터공학과

koliaok@hanyang.ac.kr, scottlee@hanyang.ac.kr

Automated Verification Method for Java-based Multithreaded Software

Hyunglak Kim○ Scott Uk-Jin Lee

Department of Computer Science & Engineering, Hanyang University

요 약

현재 오픈소스 소프트웨어는 특정 소프트웨어 및 하드웨어의 종속성을 탈피하고, 높은 신뢰성을 바탕으로 빠르게 소프트웨어 산업의 중심으로 성장하고 있다. 특히 자바 언어는 Thread-Safe 라이브러리로 안전한 멀티스레딩 환경을 지원함으로써 대표적인 오픈소스 소프트웨어의 개발 언어로 활용되고 있다. 하지만 멀티스레드의 비결정성에 따른 동시성 오류는 예기치 못한 경쟁 조건, 교착상태 등을 야기하는 문제가 있으며, 멀티스레드를 모델링하고 검증하는데 많은 시간과 노력이 필요하다. 따라서 본 논문에서는 이런 문제를 해결하기 위해 자바 기반 오픈소스 소프트웨어의 멀티스레드 검증 자동화 방법을 제안하여 오픈소스 소프트웨어의 생산성과 안전성을 높이는데 기여하고자 한다.

1. 서 론

오픈소스(Open-Source) 소프트웨어란 소스코드를 공개해 누구나 특별한 제한 없이 그 코드를 보고 사용할 수 있는 오픈 소스 라이선스를 만족하는 소프트웨어를 말한다. 현재 오픈소스 소프트웨어 개발은 라이센스 비용절감과 표준화 프로토콜을 활용함으로써 유연한 연동성을 제공해 빠르게 소프트웨어 산업의 중심으로 성장하고 있다. 또한 기존에 개발된 소스코드를 활용하여 개발기간을 단축시킴으로써 소프트웨어 생산성을 향상 시키고 있다.

이런 오픈소스 소프트웨어 개발을 위해서 다양한 프로그래밍 언어들이 활용되고 있다. 그 중 자바는 객체지향적 모델링과 Thread-Safe 라이브러리를 제공함으로써 오픈소스 소프트웨어 개발에 필요한 재 사용성, 안전성, 생산성을 모두 갖추고 있다. 따라서 자바는 오픈소스 소프트웨어 개발에 가장 많이 활용되는 기본 프로그래밍 언어로 현재까지 각광 받고 있다[1].

멀티코어 및 멀티 프로세서 프로그램을 설계하는 것이 보편화 되기 시작하면서 자바에서도 멀티스레드 개발로 인한 병렬 프로그램의 복잡성이 증가하는 문제가 존재한다. 특히 멀티스레드 프로그램의 수행 중에 발생하는 자료 경합과 원자성 위배, 교착상태 등과 같은 동시성 오류는 소프트웨어의 안전성과 신뢰성을 떨어지는 문제로 이어진다. 이와 같은 동시성 오류 문제를 검증하고자 다양한 자바 멀티스레드 검증 도구가 개발 되었으며 대표적인 검증 도구로 JPF(Java PathFinder)[2], ConTest[3], Jinx[4] 등이 있다.

하지만 기존 검증 도구들은 많은 노력과 시간을 필요로 하는 정형적인 명세 및 모델링을 이용하여 검증을 수행하고 있다. 결국 동시성 오류와 추가적인 모델링 시간 소요는 소프트웨어의 생산성과 안전성을 저해하는 원인 중 하나가 되고 있다[5].

따라서 이런 문제를 해결하기 위해 자바 기반 오픈소스 소프트웨어의 멀티스레드 코드를 추출하고, 자동으로 검증할 수 있는 방법을 제안하여 위와 같은 문제점을 해결함으로써 소프트웨어 산업의 중심으로 성장하고 있는 자바 기반 오픈소스 소프트웨어의 생산성과 안전성을 높이고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 자바 멀티스레드 검증과 관련된 배경지식을 소개하고, 3장에서는 오픈소스 소프트웨어 멀티스레드 검증과 안전성 문제를 분석한다. 4장에서는 이런 문제점을 개선할 수 있는 멀티스레드 자동화 검증 방법을 제시한다. 5장에서는 개선된 검증 방법의 장점을 활용하여 향 후 자바 기반 오픈소스 소프트웨어 자동화 검증 연구의 방향성을 제시한다.

2. 배경지식

대부분의 멀티스레드 소프트웨어 검증 방법들은 특정한 문제가 예상되는 시스템을 모델링하여 검증하는 방법을 사용하는데, 검증 시에 전체 상태를 모두 구현하여 조사하기 때문에 생기는 상태 폭발(state explosion) 문제가 있다. 이를 해결하기 위해 1995년 Bell 연구소에서 오픈소스 기반의 LTL 모델 검증 도구를 개발하였다. 이는 기존 방식과는 다르게

on-the-fly 방법을 적용해 각 상태에서 필요한 상태만 생성하기 때문에 기존 상태 폭발 문제와 메모리의 효율적인 사용을 가능하게 하였다.

이 방법은 통신 프로토콜, 침입탐지 메커니즘, 동시성을 가진 프로세스 등의 검증 연구에 활발하게 활용되고 있다. 하지만 위 검증 방식은 PROMELA라는 모델링 언어와 모델을 검증하는 SPIN이 라는 도구를 사용하기 때문에 이를 능숙하게 다를 수 있는 인력과 충분한 모델링 시간을 필요로 하다[6].

따라서 현재까지 제안된 대부분의 검증 방식은 소스코드에서 모델을 추출하여 모델링 한 결과를 검증하는 방식이며, 이는 많은 시간과 비용 및 전문성을 요구한다.

3. 오픈소스 소프트웨어 멀티스레드 검증 문제점 분석

앞선 연구에서 제안한 방식은 멀티스레드 검증을 위한 모델링 과정이 필요하여 추가적인 비용과 시간이 필요한 문제가 있다. 따라서 멀티스레드 코드를 자동으로 추출해 검증하는 방법이 꼭 필요하다. 따라서 실제 자바 기반 오픈소스 소프트웨어의 멀티스레드 구현으로 발생하는 다양한 문제에 대하여 분석하고, 이에 따라 자바 기반 오픈소스 소프트웨어를 위한 자동화 검증의 필요성에 대해 설명한다.

버그 저장소를 이용해 다양한 멀티스레드 검증 요소를 평가한 자료를[7] 바탕으로 현재 자바 기반 오픈소스 소프트웨어로 가장 많이 사용되는 Apache Tomcat과 Spring Framework 프로그램 각 3개를 검증한 결과를 비교했다. 검증할 결함으로써 (1)원자성 위반, (2)경쟁 상태, (3)잘못된 자바 라이브러리 사용, (4)잘못된 최적화 등 항목별로 나누었고, 검증 도구는 ConTest[3]와 Jinx[4]를 이용했다. 각 오픈소스 소프트웨어 프로그램의 결함에 대한 검증 결과를 아래와 같이 [표 1]과 [표 2]로 재정리 하였다.

[표 1] 오픈소스 소프트웨어 검증 결과[7]

도구	프로그램	잘못된최적화	원자성위반	경쟁상태	잘못된라이브러리사용	탐색율(%)
ConTest	Tomcat 1			X	X	70
	Tomcat 2			X		80
	Tomcat 3				X	90
	Spring 4	X	X	x	X	20
	Spring 5			X		70
	Spring 6				X	80

[표 2] 오픈소스 소프트웨어 검증 결과[7]

도구	프로그램	잘못된최적화	원자성위반	경쟁상태	잘못된라이브러리사용	탐색율(%)
Jinx	Tomcat 1			X		60
	Tomcat 2				X	40
	Tomcat 3	X	X	x	X	10
	Spring 4			X	X	90
	Spring 5				X	30
	Spring 6			X		90

위의 [표 1]과 [표 2]에서 Contest와 Jinx의 탐색율(%)은 전체 검증된 항목 중 검증할 결함 4가지 항목을 탐지하는 확률을 뜻한다. 또한 각 항목의 X표시는 각 결함을 탐지 하지 못한 것을 표현한다. 따라서 [표 1]과 [표 2]의 각 프로그램마다 검증된 항목의 개수가 달라지기 때문에 4가지 항목을 탐지하는 확률이 각각 다르다. 예를 들어 Contest 도구를 사용하여 Tomcat 2 프로그램을 검증했을 때 탐지되어진 항목이 5개이고, 검증할 결함 항목 중 경쟁 상태 하나를 탐지 못했다면 탐색율은 80%이다.

위의 결과에 의하면 Apache Tomcat 오픈소스 프로그램 3개와 Spring Framework 오픈소스 프로그램 3개를 각기 다른 2개의 멀티스레드 검증 도구로 검증 하였고, 각 도구 마다 67% 확률로 경쟁 상태를 탐색하지 못했다. 즉 멀티스레드 코드를 추출해서 자동화 검증 도구로 검증을 하여도 경쟁 상태와 같은 잘 알려진 멀티스레드 결함을 모두 탐색하지 못하는 경우가 발생한다.

현재 다양하 자바 기반 오픈소스 소프트웨어가 개발되고 있는 상황에서 경쟁 상태와 같은 기본적인 멀티스레드 문제들이 발견되지 않는 점은 병렬 프로그램에서 동시성에 의한 문제들을 검증하기가 난해하고 복잡하기 때문이다.

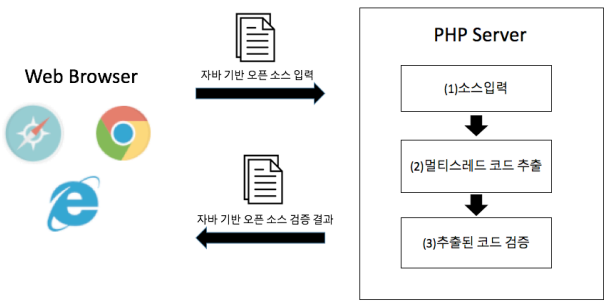
따라서 자바 언어를 사용하는 전문적인 멀티스레드 검증 도구를 이용하여 정확한 멀티스레드 동시성 문제를 검증해야한다. 또한 멀티스레드 코드를 자동으로 추출해서 모델링 과정없이 자동화 하는 방법이 필요하다.

4. 자바 기반 멀티스레드 소프트웨어 검증 자동화 방법

분석된 문제점을 개선하기 위해 자바 기반 오픈소스 소프트웨어 멀티스레드 검증 도구를 제시한다. 검증 도구의 실행 환경은 웹을 기반으로 서버사이드 언어인 PHP를 사용하고, 자동화 검증 도구를 (1)소스입력, (2)멀티스레드 코드추출, (3)추출 된 코드 검증 3단계로 구성한다.

멀티스레드 코드 추출 단계에서는 클래스간의 연관성 추적을 위해서 오픈소스인 JFMCGG(Java Fast Method Call Graph Generator)[8]를 사용한다. 그리고 추출 된 코드 검증 단계에서는 멀티스레드 모델 검증 도구인 JPF(Java PathFinder)[2]를 검증 도구로 사용한다.

JPF을 사용하는 이유는 멀티스레드 코드만 추출해서 컴파일한 바이트코드를 JPF-Core로 검증 할 수 있어 JDK가 설치된 환경이라면 운영체제에 종속되지 않고 JPF-Core를 통하여 모든 자바 소스를 검증 할 수 있기 때문이다.



[그림 1] 멀티스레드 자동화 검증 도구

(1) **소스입력**: 위의 [그림1]에서 보는 것 처럼 PHP를 통해서 자바 기반 오픈소스 소프트웨어의 경로를 지정하면 모든 프로젝트 소스들을 서버에 저장한다.

(2) **멀티스레드 코드 추출**: 첫 단계에서 서버에 저장되어 있는 모든 .java파일에서 자바 스레드 상속을 위한 implements Runable과 extend Thread 2가지 키워드를 가진 클래스들을 추출한다. 두 번째 단계에서는 추출된 클래스의 run() 메소드에 있는 인스턴스 변수들의 클래스와 이 클래스에 run()이 수행되는 메소드만 추출한다. 세 번째 단계에서 스레드를 생성하여 start() 호출과 스레드 객체를 매개변수로 넘기는 코드 부분을 추출하여 Main클래스를 생성한다. 네 번째 단계에서 필요한 자바 라이브러리를 import하여 정상적으로 컴파일 되는지 확인을 하고, 컴파일 되지 않으면 에러 부분의 인스턴스를 JFMCGG[8]를 이용하여 연관 클래스들을 역 추적 한다.

(3) **추출된 코드 검증**: JPF(Java PathFinder)[2]를 PHP의 exec함수를 이용하여 위의 코드에서 추출된 .java 파일을 추가하고, .jpf 파일을 생성해 추출된 .java 파일을 target으로 검증을 수행한다. 검증이 끝나면 자료 경합, 교착상태, 원자성 위반 등의 오류를 리스트화 시켜 사용자가 수정할 수 있도록 웹 브라우져의 화면에 출력한다.

위의 방법을 통해서 기존에 모델링을 직접 해야 했던 문제를 멀티스레드 코드 추출로 해결 할 수 있다. 또한 멀티스레드 검증 도구인 JPF를 이용해서 전문가가 아니더라도 쉽게 웹을 통해서 검증하고자 하는 프로그램을 검증 하고, 분석 할 수 있다.

5. 결론 및 향후 연구방향

현재 자바 기반 오픈소스 소프트웨어는 Spring Framework와 Hadoop과 같은 오픈소스 프레임워크로 다수의 사용자를 확보할 수 있는 완성도 높은 소프트웨어로 개발되고 있으며, 소프트웨어 산업에서 중심으로 성장하고 있다. 이에 따라서 소프트웨어의 안전성과 생산성을 향상 시키기 위해 멀티스레드 검증이 필요한 시점이다.

본 논문은 기존에 Thread-Safe 라이브러리를 제공 하는 자바에서 예측 하지 못하는 동시성 문제가 발생 할 수 있고, 이런 문제를 검증 할 때 모델링이 어려웠던 점을 개선하기 위해 자바 기반 오픈소스 소프트웨어 멀티스레드 검증 자동화 방법을 제안하여 소프트웨어의 안전성과 생산성을 향상 시키고자 한다. 따라서 본 연구의 결과는 자바 기반의 오픈소스 소프트웨어 멀티스레드의 범용적 자동화 검증 방법을 제공하여 안전성과 생산성을 향상 시키는데 기여하리라 생각된다.

향 후 연구로는 본 논문에서 제시한 검증 방법 중 JPF가 Target코드의 2000라인 이하의 코드만 검증이 가능한 문제가 존재한다. 따라서 추출된 멀티스레드 코드의 최적화를 수행하여 대규모 시스템에서 발생할 수 있는 2000라인 이상의 멀티스레드 추출 코드를 축소시켜 해결한다. 또한 다양한 자바 기반 오픈소스 소프트웨어를 테스트하고 비교하기 어려운 문제가 있다. 이를 해결하기 위해 제시한 방법을 고도화시켜 테스트 할 자바 기반 오픈소스 소프트웨어를 선정한 후 비교분석 및 평가를 수행한다. 이 후 멀티스레드 자동화 검증 방법의 효율성을 입증하고, 자바 기반인 Android의 멀티스레드를 검증하는 도구로 제시한 방법을 확장하고자 한다.

사 사

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음.
[R0601-16-1063, ICT 장비용 SW 플랫폼 구축]

참고문헌

[1] Haidar Osman, Manuel Leuenberger, Micea Lungu, “Tracking Null Checks in Open-Source Java Systems”, *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*, Vol. 1. pp. 304-313, 2016

[2] Havelund, Klaus, and Thomas Pressburger. "Model checking java programs using java pathfinder", *International Journal on Software Tools for Technology Transfer* 2.4, pp. 366-381, 2000

[3] ConTest: Testing and Debugging Concurrent Software, 2008.[Online].Available:https://www.research.ibm.com/haifa/projects/verification/contest/index.html

[4] Jinx: concurrency debugger, November 2012.[Online]. Available: https://en.wikipedia.org/wiki/Jinx_Debugger

[5] Baklanova, Nadezhda, Martin Strecker, and Louis Féraud. "Resource sharing conflicts checking in multithreaded Java programs", *Informal Proceedings FAC12*, 2012

[6] Gerard J. Holizmann, “the Model Checker SPIN” *IEEE Transactions on Software Engineering*, Vol.23, No.5. pp.279-295, 1997

[7] Schimmel, Jochen, Korbinian Molitorisz, and Walter F. Tichy, "An evaluation of data race detectors using bug repositories" *IEEE Federated Conference on Computer Science and Information Systems, pp.1361-1364, 2013*

[8] JFMCGG: Java Fast Method Call Graph Generator, 2011. [Online].Available:https://sourceforge.net/projects/jmfcgg