항공기 소프트웨어에서 원자성위배를 치유하기 위한 블록화 롤백기법

(Blocked Rollback Technique for Healing Atomicity Violation in Airborne Software)

이 동 수†, 최 으 뜸†, 전 용 기†‡*
†국립 경상대학교 정보과학과, ‡국립 경상대학교 항공우주 및 소프트웨어학과
(Dong-Su Lee, Eu-Teum Choi, Yong-Kee Jun)
(†Department of Informatics, Gyeongsang National Univ.,
‡Department of Aerospace and Software Engineering, Gyeongsang National Univ.)

Abstract : Airborne software that uses multi thread with developer's wrong estimation may occurs atomicity violation that affects reliability of Airborne software. In this paper, we propose Blocked Rollback Technique that uses blocked code and rollback recovery for healing of atomicity violation. The proposed Technique has a 30% increase in performance through experiments.

Keywords : rollback recovery, atomicity violation, airborne software

Ⅰ. 서론

항공기 시스템의 기능에 대한 요구가 커지고 있어 항공기 소프트웨어의 복잡도가 증가하고 있다. 멀티스레드를 사용하는 항공기 소프트웨어는 개발자의 적절하지 않은 추정에 인해 원자성위배가 발생할 수 있다. 동시성오류의 한 종류인 원자성위배는 원자성을 가진 영역이 다른 스레드의 실행에 영향을 받아 원자성이 깨지는 것을 의미한다. 원자성위배는 비결정적 결과로 항공기의 신뢰성 보장을 어렵게 만들어 반드시 해결해야 된다. 치유는 대상 프로그램을 실행하며 오류발생 시 복구기법을 사용하여 오류를 해결하는 기법이다.

본 연구에서는 원자성위배를 치유하기 위한 블록화 롤백치유기법을 제안한다. 블록화 롤백치유기법은 식별단계, 삽입단계, 컴파일단계의 총 세 가지로 단계로 구성된다. 식별단계는 복구영역의 최소화를 위해 분기문단위로 블록화하고, 원자성영역을 식별한다. 삽입단계는 코드변환으로 롤백명령을 삽입한다. 컴파일단계는 변환된 코드를 사용해 실행파일을 생성한다. 그리고 실험을 통해 기존의 롤백기법과 실행시간을 비교하여 효율성을 검증한다.

* 교신저자(Corresponding Author)
이동수, 최으뜸: 경상대학교
전용기: 경상대학교 정보과학과, 항공우주및소프트웨어학과, 항공기부품기술연구소, 공학연구원

Ⅱ. 연구배경

항공기 시스템에 다양한 기능이 요구되고 있다. 이 요구들은 항공기 소프트웨어의 복잡도를 증가시키고 이에 따른 코드도 증대된다. 항공기 소프트웨어에서 멀티스레드를 사용할 때 개발자의 적절하지 않은 추정으로 원자성위배가 발생할 수 있다. 원자성위배가 발생하면 항공기 소프트웨어의 신뢰성 보장이 어려워진다.

동시성오류의 한 종류인 원자성위배[2]는 멀티스레드 프로그램에서 원자성을 가진 영역이 다른 스레드의 실행에 영향을 받아 원자성이 깨지는 것을 의미한다. 원자성위배는 Therac 25 치료기기의 오작동 사례처럼 비결정적인 결과로 인명과 재산피해를 발생시킬 수 있다. 따라서 항공기 소프트웨어의 안정성과 신뢰성 보장을 위해 동시성오류는 반드시 치유되어야 한다.

치유는 대상 프로그램을 실행하면서 오류발생 시 복구기법으로 해결하는 기법이다. 그 중 롤백기법[2]은 오류발생 시 지정된 체크포인트로 돌아가는 방법으로 스레드의 실행순서를 제한하지 않아 항공기에 적용하기 적합하다. 하지만 복구영역에 오류와 연관이 없는 실행을 포함하기 때문에 시간 오버헤드가 발생할 수 있다. 본 논문은 코드 블록화를 이용해 성능을 개선한 롤백 치유기법을 제시한다.

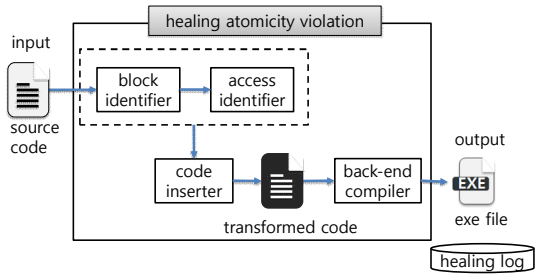

그림 1. 제안한 기법의 디자인
Fig. 1. The Design of Proposed Technique

Ⅲ. 블록화 롤백 치유기법

본 논문에서 제시한 기법은 블록화를 사용하여 코드를 분할하고 접근사건간의 연관성을 이용해 원자성 영역을 식별하고 코드변환을 사용해 롤백명령을 삽입하고 오류발생 시 롤백을 이용해 치유한다. 제시한 기법은 식별단계, 삽입단계, 그리고 컴파일 단계의 세 단계 프로세스를 통해 진행된다.

식별단계는 블록과 접근을 식별하는 두 개의 모듈로 구성된다. 블록식별자는 복구영역을 최소화하기 위하여 소스코드를 입력받아 분기문단위로 블록화를 한다. 복구영역을 지정하기 위한 접근식별자는 전달받은 블록정보를 이용해 블록내부에서 각 공유변수 접근사건부터 해당 접근사건에 연관되어 공유접근사건에서 참조되는 변수의 최초 접근사건까지 역방향으로 추적하여 복구영역으로 지정한다.

삽입단계에서는 식별단계에서 지정된 복구영역의 정보를 입력받아 해당위치에 롤백 명령어와 치유여부 확인을 위한 로그를 생성하는 코드를 삽입하여 소스코드를 변환한다. 컴파일 단계에서는 변환된 소스코드를 입력받아서 back-end compiler를 통해 대상 환경에 맞는 실행파일을 생성한다.

제시된 기법은 정적분석과 코드생성을 수행하는 LLVM 프레임워크[3]를 사용해 구현했다. 심볼로 구성되는 LLVM bitcode를 생성하여 분기문정보를 읽어서 블록화를 진행하고, 정적분석 기능을 사용하여 접근사건을 획득하고 역추적 과정을 통해 복구영역을 지정하며, setjmp/longjmp를 사용해 롤백코드를 삽입했다.

Ⅳ. 실험

실험은 LLVM 3.9.1과 SIMA[4] 환경에서 합성 프로그램을 사용해 제안한기법과 기존의 연구 중 멱등(idempotent)영역을 이용한 기법을 비교했다.

표 1. 시간 오버헤드 측정결과 (단위:ms)
Table 1. Measurement result of Time overhead

결과 / 기법	실행시간(평균)	실행시간(최악)
멱등성 치유	135.05	200.01
블록화 롤백	88.34	128.45

SIMA(Simulated Integrated Modular Avionics)는 IMA[1]를 위한 프로그래밍 규격인 ARINC-653 기반 응용프로그램들을 실행할 수 있는 환경을 제공해준다.

실험을 통해 제안한 기법은 기존기법에 대비하여 시간 오버헤드가 약30% 감소했다. 원자성위배를 치유하는데 있어 블록화를 이용한 복구영역축소와 역추적을 사용한 기법의 효율성을 검증했다.

Ⅴ. 결론

본 논문에서는 블록화와 롤백을 이용하여 원자성위배를 치유 할 수 있는 기법을 제시하였다. 실험을 통해 시간개선과 가능성을 확인하였다. 향후에는 OFP을 대상으로 실제 응용프로그램을 이용한 실험을 진행할 계획이다.

참 고 문 헌

[1] S. H. VanderLeest,, "ARINC 653 hypervisor", 29th Digital Avionics Systems Conference, pp. 5.E.2-1-5.E.2-2, October 2010

[2] D.Deng, G. Jin, M. K, A. Li, B. Liblit, S. Lu, S. Qi, J. Ren, K. Sankaralingam, L. Song, Y. Wu, M. Zhang, W. Zhang, and W. Zheng, "Fixing, preventing, and recovering from concurrency bugs", Science China Information Sciences, 2015

[3] C. Lattner, and V. Adve, “LLVM: A compilation framework for lifelong program analysis & transformation", Proc. of the international symposium on Code generation and optimization, 2004

[4] G.M. Tchamgoue, O. Ha, K. Kim, and J. Jun, “A framework for on-the-fly race healing in ARINC-653 applications”, International Journal of Hybrid Information Technology, Vol. 4, No. 2, pp. 1-12,