



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0145764
(43) 공개일자 2024년10월07일

(51) 국제특허분류(Int. Cl.)
G06F 11/07 (2006.01) G06F 11/277 (2006.01)
G06F 11/36 (2006.01) G06F 9/30 (2018.01)
G06F 9/38 (2024.01)
(52) CPC특허분류
G06F 11/0793 (2013.01)
G06F 11/0739 (2013.01)
(21) 출원번호 10-2023-0040652
(22) 출원일자 2023년03월28일
심사청구일자 2023년03월28일

(71) 출원인
경상국립대학교산학협력단
경상남도 진주시 진주대로 501 (가좌동)
(72) 발명자
전용기
경상남도 진주시 내동면 순환로 425-61, 111동
2004호 (남강휴먼빌아파트)
최으뜸
경상남도 진주시 진주역로96번길 15, 102동 902호
(가좌동, 포레나 신진주)
배창희
경상남도 진주시 가좌길44번길 11-1, 107호 (가좌
동, 선비고을)
(74) 대리인
김종선

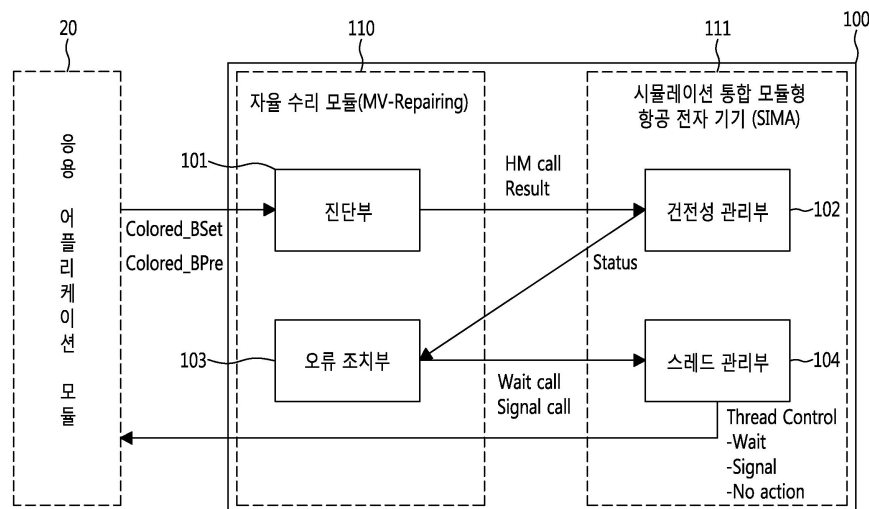
전체 청구항 수 : 총 14 항

(54) 발명의 명칭 항공기 건전성 관리시스템의 다중변수 원자성 위배 자율 수리 장치 및 방법

(57) 요약

본 발명은 항공기 건전성 관리시스템의 다중변수 원자성 위배 자율 수리 장치 및 방법에 관한 것으로 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치는 항공기 제어를 위한 응용 어플리케이션 모듈로부터 공유변수 상관관계 정보를 수집하고, 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단하는 진단부, 상기 진단에 따라 검출된 오류를 수집하는 건전성 관리부, 상기 건전성 관리부로부터 수신된 오류를 복구하기 위한 조치 명령을 결정하는 오류 조치부 및 상기 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어하는 스레드 관리부를 포함할 수 있다.

대표도



(52) CPC특허분류

G06F 11/0781 (2013.01)

G06F 11/277 (2013.01)

G06F 11/3664 (2013.01)

G06F 9/3009 (2013.01)

G06F 9/3851 (2023.08)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711188939
과제번호	2021R1A2C1014163
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	중견연구자지원사업
연구과제명	항공기 소프트웨어의 건전성 관리를 위한 개방형 시스템
기 여 율	1/2
과제수행기관명	경상국립대학교
연구기간	2021.03.01 ~ 2024.02.29

이 발명을 지원한 국가연구개발사업

과제고유번호	1415185979
과제번호	20005378
부처명	산업통상자원부
과제관리(전문)기관명	한국산업기술평가관리원
연구사업명	산업기술혁신사업
연구과제명	중소형 항공기급 개방형 항공전자 시스템 아키텍처 및 소프트웨어 개발
기 여 율	1/2
과제수행기관명	엘아이지넥스원(주)
연구기간	2019.04.01 ~ 2023.12.31

명세서

청구범위

청구항 1

항공기 제어를 위한 응용 어플리케이션 모듈로부터 공유변수 상관관계 정보를 수집하고, 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단하는 진단부;

상기 진단에 따라 검출된 오류를 수집하는 건전성 관리부;

상기 건전성 관리부로부터 수신된 오류를 복구하기 위한 조치 명령을 결정하는 오류 조치부; 및

상기 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어하는 스레드 관리부를 포함하는 다중변수 원자성 위배 자율 수리 장치.

청구항 2

제1항에 있어서,

상기 제1 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 사전에 의도된 프로그램 공유변수 접근 순서를 포함하는 것인, 다중변수 원자성 위배 자율 수리 장치.

청구항 3

제2항에 있어서,

제1 접근 순서 정보는 상기 응용 어플리케이션 모듈의 다중변수 원자성 위배를 진단하기 위해 요구되는 상기 공유변수 상관관계 정보를 포함하는 것인, 다중변수 원자성 위배 자율 수리 장치.

청구항 4

제1항에 있어서,

상기 제2 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 실제 공유변수 접근 순서를 포함하는 것인, 다중변수 원자성 위배 자율 수리 장치.

청구항 5

제1항에 있어서,

상기 진단부는,

미리 설정된 동기화 과정없이 복수의 스레드가 서로 상관관계를 가지는 복수의 공유변수에 접근하고, 상기 스레드 중 적어도 어느 하나에 쓰기 작업이 포함된 경우 발생하는 다중변수 원자성 위배 오류를 검출하되,

상기 복수의 공유변수에 대하여 상기 제1 접근 정보와 상기 제2 접근 정보의 일치 여부에 기초하여 상기 다중변수 원자성 위배 오류를 검출하는 것인, 다중변수 원자성 위배 자율 수리 장치.

청구항 6

제5항에 있어서,

상기 진단부는,

상기 제1 접근 순서 정보의 사전에 의도된 프로그램 공유변수 접근 순서와 상기 제2 접근 순서 정보의 실제 공유변수의 접근 순서를 비교하여 상기 다중변수 원자성 위배 오류를 검출하는 것인, 다중변수 원자성 위배 자율 수리 장치.

청구항 7

제5항에 있어서,

상기 오류 조치부는,

검출된 오류와 관계된 스레드를 지연시키기 위한 조치 명령을 각 오류마다 결정하고,

상기 스레드 관리부는,

상기 오류별로 결정된 조치 명령에 기초하여 상기 오류와 관계된 스레드를 지연시키는 것인, 다중변수 원자성 위배 자율 수리 장치.

청구항 8

(a) 항공기 제어를 위한 응용 어플리케이션 모듈로부터 공유변수 상관관계 정보를 수집하는 단계;

(b) 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단하는 단계;

(c) 상기 진단에 따라 검출된 오류를 수집하는 단계;

(d) 상기 진단성 관리부로부터 수신된 오류를 복구하기 위한 조치 명령을 결정하는 단계;

(e) 상기 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어하는 단계를 포함하는 다중변수 원자성 위배 자율 수리 방법.

청구항 9

제8항에 있어서,

상기 제1 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 사전에 의도된 프로그램 공유변수 접근 순서를 포함하는 것인, 다중변수 원자성 위배 자율 수리 방법.

청구항 10

제9항에 있어서,

제1 접근 순서 정보는 상기 응용 어플리케이션 모듈의 다중변수 원자성 위배를 진단하기 위해 요구되는 상기 공유변수 상관관계 정보를 포함하는 것인, 다중변수 원자성 위배 자율 수리 방법.

청구항 11

제8항에 있어서,

상기 제2 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 실제 공유변수 접근 순서를 포함하는 것인, 다중변수 원자성 위배 자율 수리 방법.

청구항 12

제8항에 있어서,

상기 (b) 단계는,

미리 설정된 동기화 과정없이 복수의 스레드가 서로 상관관계를 가지는 복수의 공유변수에 접근하고, 상기 스레드 중 적어도 어느 하나에 쓰기 작업이 포함된 경우 발생하는 다중변수 원자성 위배 오류를 검출하되,

상기 복수의 공유변수에 대하여 상기 제1 접근 정보와 상기 제2 접근 정보의 일치 여부에 기초하여 상기 다중변수 원자성 위배 오류를 검출하는 것인, 다중변수 원자성 위배 자율 수리 방법.

청구항 13

제12항에 있어서,

상기 (b) 단계는,

상기 제1 접근 순서 정보의 사전에 의도된 프로그램 공유변수 접근 순서와 상기 제2 접근 순서 정보의 실제 공유변수의 접근 순서를 비교하여 상기 다중변수 원자성 위배 오류를 검출하는 것인, 다중변수 원자성 위배 자율 수리 방법.

청구항 14

제12항에 있어서,

상기 (d) 단계는,

검출된 오류와 관계된 스레드를 지연시키기 위한 조치 명령을 각 오류마다 결정하고,

상기 (e) 단계는,

상기 오류별로 결정된 조치 명령에 기초하여 상기 오류와 관계된 스레드를 지연시키는 것인, 다중변수 원자성 위배 자율 수리 방법.

발명의 설명

기술 분야

[0001] 본 발명은 항공기 건전성 관리시스템의 다중변수 원자성 위배 자율 수리 장치 및 방법에 관한 것이다.

배경 기술

[0002] 항공 소프트웨어가 발전함에 따라 비행에 항공기에 탑재되는 기능이 증가하고 있다. 항공기에 탑재되는 소프트웨어 비중은 1960년대 생산된 F-4 기종 기준 약 8%였지만, 2007년대 생산된 F-35 기종 기준 약 90%까지 증가되었다고 알려졌다. 항공기의 소프트웨어 비중이 증가함에 따라 탑재되는 기능의 규모 및 복잡도 등이 증가되므로 시스템에서 발생할 수 있는 잠재적인 오류의 수도 증가할 수 있다. 항공기 소프트웨어에 내재된 잠재적인 오류는 항공기 추락, 불시착을 일으키며, 이는 인명 사고로 이어질 수 있다.

[0003] 이와 같은 사고를 사전에 방지하기 위해 항공기 소프트웨어의 오류가 발생해도 사고로 이어지지 않도록 오류를 자율 수리하는 시스템이 필요하다. 항공기는 사고가 발생하면 인명 피해로 이어질 가능성이 크기 때문에 항공기 시스템에서 내재된 잠재적인 오류를 관리하는 시스템이 필요하다. 최근 항공기에 탑재되는 시스템에서 발생할 수 있는 잠재적인 오류를 진단 및 조치하여 파티션의 올바른 상태를 유지될 수 있도록 하는 건전성 관리 시스템이 개발되고 있다.

[0004] 항공기 건전성 관리시스템(Health management System, HMS)은 시스템 운용 중에 발생할 수 있는 오류를 진단(diagnosis) 및 조치(treatment)하여 시스템이 올바르게 동작할 수 있도록 한다. HMS는 시스템에서 발생하는 여

러 이벤트 및 상태 정보를 수집하여 사전에 정의된 오류 발생 조건을 비교하여 오류를 진단(diagnosis)한다. 탐지된 오류는 시스템이 올바르게 동작할 수 있도록 사전에 정의된 복구(recovery) 알고리즘을 수행하여 조치(treatment)하여 시스템의 건전성을 유지하는 것을 목적으로 한다.

[0005] 원자성 위배는 동시성 오류 중 하나로, 병행 프로그램에서 다수의 스레드가 공유변수에 원자적으로 접근해야 하지만, 원자적으로 접근하지 않을 때 발생하는 오류이다. 원자성 위배는 스레드 간의 모든 인터리빙(Interleaving)을 고려할 수 없기 때문에 개발 단계에서 제거하기 어렵다. 원자성 위배가 발생하면 프로그래머가 의도한 결과가 도출되지 않기 때문에 시스템이 실패로 이어지게 된다. 원자성 위배는 위배에 참여하는 공유변수 수에 따라 단일변수 원자성 위배와 다중변수 원자성 위배로 분류할 수 있다.

[0006] 항공기 소프트웨어에서 발생하는 원자성 위배는 스레드 간 모든 인터리빙을 고려할 수 없기 때문에 개발 단계에서 모두 제거할 수 없다. 시스템에 내재된 오류는 시스템을 실패로 이어지게 만들기 때문에 운용 중에 자율 수리하는 기술이 필요하다. 최근 공유변수 접근 사건 정보, 스레드 정보 등을 수집하여 항공기 소프트웨어의 건전성 관리시스템에서 발생하는 원자성 위배를 자율 수리하는 기법이 개발되고 있다. 하지만, 해당 연구는 단일변수 원자성 위배에 초점을 맞추고 있기 때문에 다중변수 원자성 위배를 자율 수리할 수 없다.

[0007] 따라서, 항공기 소프트웨어에서 발생하는 단일 및 다중변수 원자성 위배를 모두 자율 수리할 수 있는 기법이 필요하다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) 대한민국 공개특허공보 제10-2016-0113475호 (2016년 09월 29일 공개)

발명의 내용

해결하려는 과제

[0009] 본 발명은 사전시험을 통해 생성된 올바른 인터리빙 정보를 기반으로 실행 중에 발생하는 다중변수 원자성 위배를 자율적으로 조치할 수 있는 항공기 건전성 관리시스템의 다중변수 원자성 위배 자율 수리 장치 및 방법을 제공하는데 그 목적이 있다.

과제의 해결 수단

[0010] 상기 기술적 과제를 해결하기 위한 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치는 항공기 제어를 위한 응용 어플리케이션 모듈로부터 공유변수 상관관계 정보를 수집하고, 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단하는 진단부, 상기 진단에 따라 검출된 오류를 수집하는 건전성 관리부, 상기 건전성 관리부로부터 수신된 오류를 복구하기 위한 조치 명령을 결정하는 오류 조치부 및 상기 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어하는 스레드 관리부를 포함할 수 있다.

[0011] 또한, 상기 제1 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 사전에 의도된 프로그램 공유변수 접근 순서를 포함할 수 있다.

[0012] 또한, 제1 접근 순서 정보는 상기 응용 어플리케이션 모듈의 다중변수 원자성 위배를 진단하기 위해 요구되는 상기 공유변수 상관관계 정보를 포함할 수 있다.

[0013] 또한, 상기 제2 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 실제 공유변수 접근 순서를 포함할 수 있다.

[0014] 또한, 상기 진단부는, 미리 설정된 동기화 과정없이 복수의 스레드가 서로 상관관계를 가지는 복수의 공유변수에 접근하고, 상기 스레드 중 적어도 어느 하나에 쓰기 작업이 포함된 경우 발생하는 다중변수 원자성 위배 오류를 검출하되, 상기 복수의 공유변수에 대하여 상기 제1 접근 정보와 상기 제2 접근 정보의 일치 여부에 기초하여 상기 다중변수 원자성 위배 오류를 검출할 수 있다.

[0015] 또한, 상기 진단부는, 상기 제1 접근 순서 정보의 사전에 의도된 프로그램 공유변수 접근 순서와 상기 제2 접근

순서 정보의 실제 공유변수의 접근 순서를 비교하여 상기 다중변수 원자성 위배 오류를 검출할 수 있다.

[0016] 또한, 상기 오류 조치부는, 검출된 오류와 관계된 스레드를 지연시키기 위한 조치 명령을 각 오류마다 결정하고, 상기 스레드 관리부는, 상기 오류별로 결정된 조치 명령에 기초하여 상기 오류와 관계된 스레드를 지연시킬 수 있다.

[0017] 상기 기술적 과제를 해결하기 위한 본 발명의 또 다른 일 실시예에 따른 다중변수 원자성 위배 자율 수리 방법은 a) 항공기 제어를 위한 응용 어플리케이션 모듈로부터 공유변수 상관관계 정보를 수집하는 단계, (b) 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단하는 단계, (c) 상기 진단에 따라 검출된 오류를 수집하는 단계, (d) 상기 건전성 관리부로부터 수신된 오류를 복구하기 위한 조치 명령을 결정하는 단계, (e) 상기 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어하는 단계를 포함할 수 있다.

발명의 효과

[0018] 상기한 본 발명의 일 실시예에 따르면, 사전시험을 통해 생성된 올바른 인터리빙 정보를 기반으로 실행 중에 발생하는 다중변수 원자성 위배를 자율적으로 조치할 수 있는 항공기 건전성 관리시스템의 다중변수 원자성 위배 자율 수리 장치 및 방법을 제공할 수 있다.

도면의 간단한 설명

[0019] 도 1은 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 구성을 도시한 도면이다.

도 2는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 공유변수 접근 정보 수집의 예를 도시한 도면이다.

도 3은 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 다중변수 원자성 위배 발생 사례를 도시한 도면이다.

도 4는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 동작을 검증하기 위한 코드 삽입의 예를 도시한 도면이다.

도 5는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 동작 예를 도시한 도면이다.

도 6a 및 6b는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 수리 범위 검증을 위한 실험 유형의 예 및 결과를 도시한 도면이다.

도 7a 및 7b는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치를 활용한 시뮬레이션 결과 및 환경설정을 도시한 도면이다.

도 8a 내지 도 8f는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치와 종래 기법의 효율성 비교를 위한 오버헤드 비교 도면이다.

도 9는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 방법의 흐름을 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0020] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시 예를 가질 수 있는 바, 특정 실시 예들을 도면에 예시하고 상세하게 설명하고자 한다.

[0021] 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0022] 본 출원에서 사용한 용어는 단지 특정한 실시 예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0023] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일

반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 가진 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

- [0024] 이하, 첨부한 도면들을 참조하여, 본 발명의 바람직한 실시 예를 보다 상세하게 설명하고자 한다. 본 발명을 설명함에 있어 전체적인 이해를 용이하게 하기 위하여 도면상의 동일한 구성요소에 대해서는 동일한 참조부호를 사용하고 동일한 구성요소에 대해서 중복된 설명은 생략한다.
- [0025] 도 1은 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 구성을 도시한 도면이다.
- [0026] 도 1을 참조하면, 다중변수 원자성 위배 자율 수리 장치(100)는 진단부(101), 건전성 관리부(102), 오류 조치부(103) 및 스레드 관리부(104)를 포함할 수 있다.
- [0027] 본 다중변수 원자성 위배 자율 수리 장치(100)는 항공기에 적용되어 항공기 시스템 기능의 실패를 방지하기 위한 것이다. 항공기의 운용중에 발생하는 오류는 감시, 진단 및 조치의 대상이며, 항공기 건전성 관리 장치(100)는 항공기 시스템의 오류를 관리할 수 있다. 현재까지 항공기 시스템의 오류를 검출하는 다양한 기법들은 존재하고 활용되어 왔으나, 시스템 오류 중 원자성 위배는 해결하기 어려운 오류 중 하나이다.
- [0028] 원자성 위배는 병렬 프로그램에서 개발자가 원자적으로 실행되도록 의도한 원자성 영역(atomic region)이 의도와 다르게 실행되는 오류이다. 원자성 위배는 공유 변수에 접근하는 두 개 이상의 스레드가 적절한 동기화 없이 적어도 한번 이상의 쓰기 명령이 실행되면 발생할 수 있다. 병렬 스레드 또는 동시성 프로그램의 개발에 대한 충분한 지식과 이해가 있는 개발자라 하더라도 프로그램의 모든 실행 상황을 개발자가 고려할 수 없기 때문에 코드에 잠재적 원자성 위배가 존재할 수 있다.
- [0029] 소프트웨어의 결함을 제거하기 위한 소프트웨어 검증 및 확인 과정에서 먼저 정적·동적 분석 도구를 사용하여 소스 코드의 결함 위치를 탐지한다. 하지만, 원자성 위배와 같은 동시성 오류를 탐지 및 디버깅을 하는 것은 전문적인 지식과 관련 도구의 부족으로 인해 매우 어려운 실정이다. 일반적인 순차 프로그램에서 발생하는 오류와 병렬 프로그램에서 발생하는 동시성 오류를 비교하면, 동시성 오류를 수정하기 위해 보다 많은 개발 자원이 요구된다. 또한, 동시성 오류를 수정하기 위해서는 일반적인 순차 프로그램에서 발생하는 오류에 비해 보다 많은 파일에 영향이 발생한다. 그럼에도 불구하고, 동시성 오류를 수정한 코드 중 일부는 부정확한 수정으로 판명되는 경우가 존재한다.
- [0030] 이처럼 일반적인 원자성 위배는 해결하기 어려운 오류 중 하나이나, 특히 다중변수 원자성 위배는 일반적인 원자성 위배보다 더 해결하기 어려운 오류로 알려져 있다. 다중변수 원자성 위배는 병행 프로그램에서 올바른 동기화 기법 없이 두 개 이상의 스레드가 상관관계를 가지고 있는 두 개 이상의 공유변수에 접근할 때 발생하며, 하나 이상의 쓰기 작업이 포함될 때 발생하는 오류이다. 다중변수 원자성 위배는 모든 공유변수 접근 순서를 고려하기 어렵기 때문에 개발 단계에서 제거하기 어렵다.
- [0031] 건전성 관리 시스템은 제거되지 않은 오류를 자율 수리할 수 있기 때문에 중요한 요소 중 하나이다. 건전성 관리 시스템에서 종래의 원자성 위배 진단 기법은 단일변수 원자성 위배만 자율 수리할 수 있고 다중변수 원자성 위배는 자율 수리할 수 없기 때문에 프로그램의 올바른 결과를 보장하기 어렵다.
- [0032] 도 2는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 공유변수 접근 정보 수집의 예를 도시한 도면이다.
- [0033] 본 발명은 다중변수 원자성 위배를 자율 수리할 수 있는 기법인 Multi-Variable Repairing (다중변수 원자성 위배 자율 수리 장치(100))를 제시한다. 다중변수 원자성 위배 자율 수리 장치(100)은 Anticipating Invariant (AI)라고 불리는 사전 공유변수 접근 순서와 실제 공유변수 접근 순서를 비교하는 기법에 ColorSafe라고 불리는 공유변수 상관관계를 분석(VColor, Variable Color)하는 기법을 결합하여 오류를 진단하고 오류에 참여하는 스레드를 지연시키는 기법을 통해 오류를 조치할 수 있다. 도 2를 참조하면, 다중변수 원자성 위배 자율 수리 장치(100)는 다중변수 원자성 위배를 진단하는데 필요한 공유변수의 상관관계 정보(VColor)를 수집하고, 공유변수의 상관관계 정보(VColor)가 포함된 사전 의도한 공유변수 접근 정보(Colored_BSet)를 수집할 수 있다. 상관관계 정보가 포함된 사전 의도한 공유변수 접근 정보를 기반으로 원자성 위배를 진단하기 때문에 다중변수 원자성 위배를 진단할 수 있다. 다중변수 원자성 위배 자율 수리 장치(100)의 입력과 출력은 아래와 같을 수 있다.
- [0034] 입력: Colored_BSet(), Colored_RPre()

- [0035] 출력: Wait, Signal, No action
- [0036] 구체적으로, 다중 원자성 위배의 오류의 검출을 위하여 진단부(101)는 항공기 제어를 위한 응용 어플리케이션 모듈(20)로부터 공유변수 상관관계 정보를 수집할 수 있다. 구체적으로, 진단부(101)는 응용 어플리케이션 모듈(20)로부터 항공기 시스템의 현재 상태, 센서 데이터, 모듈의 입출력 및 이벤트 발생 정보를 수집하고, 상기 수집된 정보로부터 공유변수 상관관계 정보를 획득할 수 있다. 진단부(101)는 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단할 수 있다. 예시적으로, 진단부(101)는 사전 프로그램 수행으로 수집한 Colored_BSet()과 실제 수행 중에 수집한 Colored_RPre()를 비교하여 일치하지 않으면 원자성 위배 오류를 검출할 수 있다. 또한, 공유변수의 상관관계 정보를 정의한 VColor와 Colored_RPre()의 집합인 Colored_BSet()은 사전 프로그램 수행을 통해 수집된 정보를 통해 정의될 수 있다. 사전 프로그램 수행의 입력과 출력은 아래와 같다
- [0037] 입력: Source Code
- [0038] 출력: VColor, Colored_BSet()
- [0039] 사전 프로그램 수행의 동작 프로세스에 대해 설명하면, 먼저, 프로그램 코드를 입력으로 받아 공유변수 접근사건 정보를 수집하기 위한 추가 코드를 삽입시킨다. 이후 프로그램을 수행시켜 필요한 정보들을 수집한다. 수집하는 정보는 Thread ID, 공유변수 접근 ID, 접근하는 공유변수 주소, 구조체 여부, 구조체 주소를 포함할 수 있다. VColor: Pre-processing 모듈은 수집한 정보를 기반으로 공유변수 상관관계 정보인 VColor를 정의할 수 있다. Colored_BSet: Pre-processing 모듈은 사전 프로그램 수행으로 정의한 공유변수 상관관계 정보인 VColor와 의도한 공유변수 접근 정보인 Colored_RPre()의 집합을 결합한 Colored_BSet()을 정의할 수 있다. 사전 프로그램은 사용자 또는 관리자가 정의한 횟수만큼 수행될 수 있고, 이에 따라 공유변수 접근 정보가 수집될 수 있다.
- [0040] 여기서, 제1 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 사전에 의도된 프로그램 공유변수 접근 순서를 포함할 수 있다. 또한, 제1 접근 순서 정보는 응용 어플리케이션 모듈(20)의 다중변수 원자성 위배를 진단하기 위해 요구되는 공유변수 상관관계 정보를 포함할 수 있다. 제2 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 실제 공유변수 접근 순서를 포함할 수 있다.
- [0041] 정리하면, 제1 접근 순서 정보는 공유변수 상관관계 정보와 공유변수 상관관계 정보가 포함된 사전에 의도된 프로그램 공유변수 접근 순서를 포함할 수 있다. 제2 접근 순서 정보는 공유변수 상관관계 정보가 포함된 실제 공유변수의 접근 순서를 포함할 수 있다.
- [0042] 진단부(101)는 미리 설정된 동기화 과정없이 복수의 스레드가 서로 상관관계를 가지는 복수의 공유변수에 접근하고, 스레드 중 적어도 어느 하나에 쓰기 작업이 포함된 경우 발생하는 다중변수 원자성 위배 오류를 검출할 수 있다. 즉, 진단부(101)는 복수의 공유변수에 대하여 제1 접근 정보와 상기 제2 접근 정보의 일치 여부에 기초하여 상기 다중변수 원자성 위배 오류를 검출할 수 있다.
- [0043] 구체적으로, 진단부(101)는 제1 접근 순서 정보의 사전에 의도된 프로그램 공유변수 접근 순서와 제2 접근 순서 정보의 실제 공유변수의 접근 순서를 비교하여 다중변수 원자성 위배 오류를 검출할 수 있다. 원자성 위배는 공유변수에 2개 이상의 Thread가 최소 한 번 이상 쓰기 작업을 수행할 때 발생할 수 있다. 특히, 다중변수 원자성 위배는 상관관계를 가지는 2개 이상의 공유변수에 2개 이상의 Thread가 올바른 동기화 기법 없이 접근하여 공유변수 간의 일관성이 파괴되는 오류이다.
- [0044] 여기서 상관관계란 하나의 변수가 수정되면 같이 수정되어야 하는 관계에 있는 변수들이다. 예를 들어, Hello라는 문장을 저장하고 있는 Str 변수와 Str변수의 길이인 5를 저장하고 있는 Str_Size라는 변수 2개가 존재한다고 가정한다. 여기서 Str이 Hi로 바뀌게 되면 Str_Size도 Hi의 길이인 2로 바뀌어야 한다. 이처럼 하나의 변수를 수정하면 다른 변수도 같이 수정해야 하는 관계에 있는 변수들을 상관관계에 있는 변수라고 하며, 다중변수 원자성 위배가 발생하면 상관관계에 있는 변수 간의 일관성이 파괴된다.
- [0045] 도 3은 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 다중변수 원자성 위배 발생 사례를 도시한 도면이다.
- [0046] 도 3은 Ardupilot에서 비행 상태를 업데이트하고 현재 비행 상태를 읽는 코드 일부로 다중변수 원자성 위배가 발생한 사례이다. 도 3을 참조하면, UpdateFilter()는 관성 측정 장치(Inertial Measurement Unit, IMU)가 변경될 때마다 수행되어 속도, 위치, 고도, 속도, 자기계, 오프셋 값을 업데이트한다. get_variances()는 현재 속도, 위치, 고도 등을 반환한다. 여기서 상관관계에 있는 공유변수는 IMU가 변경될 때마다 같이 업데이트되는 속

도, 위치, 고도, 속도, 자기계, 오프셋이며, 해당 변수들은 일관성을 유지해야 한다.

- [0047] 하지만, 도 3과 같은 인터리빙으로 수행되면 공유변수 간의 일관성이 파괴되는 다중변수 원자성 위배가 발생할 수 있다. 예를 들어, 현재 속도, 위치, 고도, 속도, 자기계, 오프셋 값은 A 시점의 상태 값이고 Thread 1은 이를 다시 B 시점의 상태 값으로 업데이트한다고 가정한다. Thread 1은 자기계, 속도, 위치, 고도 값만 업데이트 되고 문맥 교환이 발생하여 Thread 2가 모든 공유변수 값을 읽는다. 이후 다시 Thread 2가 속도, 오프셋 값을 업데이트하면 결과적으로 Thread 2는 B 시점의 자기계, 속도, 위치, 고도 값과 A 시점의 속도, 오프셋 값을 읽게 되어 변수 간의 일관성이 파괴되는 오류가 발생한다.
- [0048] 기존의 항공기 소프트웨어의 건전성 관리시스템에서 수행 중 발생하는 원자성 위배를 자율 수리하는 연구의 진단 기법은 공유변수의 상관관계를 고려하지 않으며, 공유변수의 주소에 따라 서로 다른 진단 및 조치를 수행하기 때문에 다중변수 원자성 위배를 자율 수리할 수 없다는 문제점이 존재한다. 과거의 프로그래머들은 다중변수 원자성 위배는 드물게 발생할 것으로 예상했지만, 최근 일반적으로 발생하는 원자성 위배라고 설명하였다. 결정적으로 다중변수 원자성 위배에서는 단일변수 원자성 위배가 발생하지 않기 때문에 기존의 연구기법으로 다중변수 원자성 위배를 자율 수리할 수 없다. 따라서 다중변수 원자성 위배가 발생해도 이를 자율 수리하여 프로그램의 올바른 결과를 보장할 수 있는 기법이 요구된다.
- [0049] 앞서 설명한 Colored_RPre()와 Colored_BSet()은 AI 기반의 RPre()와 BSet() 기법에 ColorSafe 기반의 VColor 기법을 추가한 것이다. ColorSafe 기반의 VColor 기법은 공유변수의 색상을 부여하여 상관관계에 있는 공유변수를 하나의 변수로 인식하는 기법이다. 이 기법을 사용하면 다중변수 원자성 위배 진단하는 복잡성을 단일변수 원자성 위배 복잡성으로 감소할 수 있다.
- [0050] 진단부(101)는 공유변수 상관관계 정보가 추가된 사전 프로그램 수행 정보(Colored_BSet)와 공유변수 상관관계 정보가 추가된 실제 프로그램 수행 정보(Colored_RPre)를 비교하여 오류를 진단할 수 있다. 진단 모듈은 오류를 진단하여 스레드를 멈추는 시점 또는 올바른 수행 시점을 진단하여 건전성 관리부(102)에 전송한다.
- [0051] 건전성 관리부(102)는 진단부(101)로부터 받은 진단 결과를 보고 받고 상태를 모니터링할 수 있다. 이후 보고 받은 결과는 오류 조치부(103)에 전달하여 프로그램이 건전한 상태를 유지하기 위해 조치를 수행할 수 있도록 한다.
- [0052] 오류 조치부(103)는 건전성 관리부(102)로부터 수신된 오류를 복구하기 위한 조치 명령을 결정할 수 있다. 스레드 관리부(104)는 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어할 수 있다. 또한, 스레드 관리부(104)는 오류별로 결정된 조치 명령에 기초하여 상기 오류와 관계된 스레드를 지연시킬 수 있다.
- [0053] 도 1을 참조하면, 진단부(101) 및 오류 조치부(103)은 다중변수 원자성 위배를 자율적으로 수리할 수 있는 자율 수리 모듈(Multi-Variable Repairing (MV-Repairing))(110)에 포함될 수 있다. 건전성 관리부(102) 및 스레드 관리부(104)는 시뮬레이션 통합 모듈형 항공 전자기기(simulated integrated modular avionics, SIMA)(111)에 포함될 수 있다.
- [0054] 자율 수리 모듈(110)은 사전 시험한 접근 정보를 통해 프로그램 실행 중에 자율적으로 원자성 위배를 진단하는 AI(Anticipate Invariant)에 기초할 수 있다. 자율 수리 모듈(110)의 입력은 실행 중 접근 정보, 사전 시험한 접근 정보를 포함하고, 출력은 스레드 멈춤 및 스레드 재개 명령을 포함할 수 있다.
- [0055] 자율 수리 모듈(110)의 입력에서 실행 중 접근 정보는 공유변수의 접근 사건 및 주소와 스레드를 포함할 수 있다. 사전 시험한 접근 정보는 정적 명령어(S_x)의 집합을 포함할 수 있다. 또한, 수행 중 수집된 동적 명령어(D_x)를 기반으로 (a) D_x 의 접근과 동일한 주소를 가진 동적 명령어, (b) D_x 가 속하지 않은 다른 스레드에 접근하는 동적 명령어, (c) D_x 의 수행 직전에 접근하는 동적 명령어의 조건을 만족하는 정적 명령어를 수집할 수 있다.
- [0056] 자율 수리 모듈(110)은 각 스레드에 공유 자원의 접근을 감시하여 다중변수 원자성 위배의 발생을 진단하는 진단부(Diagnosis Module(101)과, 지연(stall)을 삽입하여 인터리빙이 올바르게 되도록 조치하는 오류 조치부(Treatment Module)(103)를 포함할 수 있다. 오류 조치부(103)는, 검출된 오류와 관계된 스레드를 지연시키기 위한 조치 명령을 각 오류마다 결정할 수 있다. 하나의 오류와 관계된 스레드를 예시로 설명한다. 오류가 검출되면 오류 조치부(103)는 오류가 발생한 스레드를 멈출 수 있다. 이후 오류가 해결되면, 오류 때문에 멈춘 스레드가 다른 스레드가 수행되어도 정상적으로 수행할 수 있다고 진단되면 멈춘 스레드를 재개시킬 수 있다.
- [0057] 진단부(101)에서 다중변수 원자성 위배가 검출되면, 오류 조치부(103)는 조건 변수를 이용하여 해당 스레드를

멈추었다가 이후에 진단 모듈이 정상 실행으로 진단하면 멈춘 스레드를 재개시킬 수 있다. 예시적으로, 오류 조치부(103)는 현재의 수리 상태를 확인하기 위해 현재 모니터링하는 오류 상태를 활용할 수 있다. 상태는 최초 프로그램이 실행되었을 때의 상태는 normal일 수 있다. 수행 중에 진단부(101)가 원자성 위배를 진단하여 오류 조치부(103)가 wait()를 수행하면 wait 상태로 전이한다. Wait 상태에서 진단부(101)가 오류를 진단하더라도 오류 조치부(103)는 조치를 시도하지 않는다. 그리고, 진단부(101)가 다중변수 원자성 위배가 없다고(수리되었음) 진단하면, 오류 조치부(103)는 signal()을 통해 멈춘 스레드를 재개시키고 normal 상태로 전이할 수 있다.

[0058] 응용 어플리케이션 모듈(20)에서 다중변수 원자성 위배가 발생하지 않으면, 자율 수리모듈(110)은 어떠한 조치도 하지 않는다. 다중변수 원자성 위배가 발생하면, 응용 어플리케이션 모듈(20)의 운영체제가 오류에 영향을 받을 수 있는 파티션 또는 프로그램들을 격리할 수 있다. 이후 오류 조치부(103)가 스레드를 제어하여 안전하게 다중변수 원자성 위배를 수리하기 위한 명령을 생성하고, 스레드 관리부(104)가 조치 명령에 기초하여 다중변수 원자성 위배 스레드를 수리할 수 있다.

[0059] 도 4는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 동작을 검증하기 위한 코드 삽입의 예를 도시한 도면이다.

[0060] 다중변수 원자성 위배 자율 수리 장치(100)의 동작은 원본 코드에 추가적인 코드가 삽입되어 수행된다. 도 4는 다중변수 원자성 위배 자율 수리 장치(100)에 의해 다중변수 원자성 위배가 올바르게 수리되었는지 확인하기 위한 바이너리 코드 결과를 도시한다. 도 4를 참조하면, 공유변수가 실행되기 전후로 AV-DE 코드가 정상적으로 삽입되었다. 공유변수가 실행되기 전의 AV-DE 코드는 다중변수 원자성 위배가 진단되었을 때 실행을 멈추기 위한 wait() 코드이고, 후의 AV-DE 코드는 원자성 위배가 수리되면 스레드를 재개하기 위한 signal() 코드이다.

[0061] 도 5는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 동작 예를 도시한 도면이다.

[0062] 시험 단계에서 사전 시험한 접근 정보가 생성되고, 다중변수 원자성 위배 자율 수리 장치(100)를 정상적으로 운용 단계를 진행하였다. 도 5는 다중변수 원자성 위배 자율 수리 장치(100)가 다중 변수 원자성 위배 검출 및 수리과정을 설명하기 위한 도면이다.

[0063] 도 5는 다중변수 원자성 위배를 수리하는 사례를 나타낸다. 도 5의 오류 사례는 Thread 1이 상관관계에 있는 공유변수 접근이 끝난 후 Thread 2가 실행되는 경우와 Thread 2가 상관관계에 있는 공유변수 접근이 끝난 후 Thread 1이 실행되는 경우에만 다중변수 원자성 위배가 일어나지 않는다.

[0064] 오류가 발생하는 인터리빙은 Thread 1이 상관관계에 있는 공유변수의 접근하는 도중 Thread 2가 실행되어 다중변수 원자성 위배가 발생하는 상황이다. Execution 열은 각 Thread의 공유변수 접근순서를 나타내며, 상관관계에 있는 공유변수 접근 사건은 VColor 기법을 사용하여 W^{RED}/R^{RED} 1-4로 표시하였다.

[0065] Input 열은 A-E 시점에서 원자성 위배를 진단하기 위한 비교 과정을 나타낸다. Input 열의 Colored_RPre()는 실제 프로그램 수행 중 수집된 사전 공유변수 접근사건이며, Input 열의 Colored_BSet()은 사전에 의도한 공유변수 접근사건을 의미한다. Comp. 열은 Colored_RPre()와 Colored_BSet()을 비교한 결과를 의미한다. Output of module 열은 각 시점에서 원자성 위배를 진단 및 Thread 조치 과정을 나타낸다. Diagnosis 열은 원자성 위배가 진단되면 Atomicity Violation으로 표시하며, treatment 열은 원자성 위배가 진단되면 Thread를 멈추거나 재개시키는 상태를 표시한다. 각 상태는 Wait, Signal, Resume으로 표시한다.

[0066] A 시점에서 W^{RED} 1 접근사건의 Colored_RPre()는 null이고 Colored_BSet()에 null이 포함되어 있으므로 정상적인 수행 순서로 판단한다. B 시점에서 R^{RED} 3 접근사건의 Colored_RPre()는 W^{RED} 1이지만 Colored_BSet()에 W^{RED} 1이 포함되어 있지 않기 때문에 diagnosis 모듈에서 원자성 위배로 판단한다. 원자성 위배가 판단되면 treatment 모듈은 R^{RED} 3 접근사건이 발생하기 전에 Wait()를 호출하여 Thread를 멈춘다. C 시점에서 W^{RED} 2 접근사건의 Colored_RPre()는 null이고 Colored_BSet()에 null이 포함되어 있으므로 정상적인 수행 순서로 판단한다.

[0067] 이후 B 시점에서 멈춘 Thread 2의 R^{RED} 3 사건의 Colored_BSet()에 W^{RED} 2가 포함되어 있기 때문에 C 시점의 W^{RED} 2 접근사건 이후 멈춘 Thread를 재개하기 위해 Signal()을 호출한다. D 시점에서 Thread가 재개하여 R^{RED} 3 접근사건이 다시 수행되며 D 시점에서 Colored_RPre()는 W^{RED} 2이고 Colored_BSet()에 W^{RED} 2이 포함되어 있으므로 정상적인 수행 순서로 판단한다.

- [0068] 마지막 E 시점에서 W^{RED}_4 접근사건의 Colored_RPre()는 W^{RED}_2 이고 Colored_BSet()에 W^{RED}_2 가 포함되어 있으므로 정상적인 수행 순서로 판단한다. 결과적으로 다중변수 원자성 위배 자율 수리 장치(100)는 Thread 1의 상관관계에 있는 공유변수 접근 중 Thread 2가 공유변수에 접근하는 다중변수 원자성 위배가 발생하였지만, Thread 1의 공유변수 접근이 모두 끝난 후 Thread 2가 공유변수에 접근하는 올바른 순서($W^{\text{RED}}_1 \rightarrow W^{\text{RED}}_2 \rightarrow R^{\text{RED}}_3 \rightarrow R^{\text{RED}}_4$)로 실행될 수 있도록 Thread를 제어하여 원자성 위배를 수리할 수 있다.
- [0069] 도 6a는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치의 수리 범위 검증을 위한 실험 유형의 예를 도시한 도면이다.
- [0070] 도 6은 다중변수 원자성 위배 사례와 단일변수 원자성 위배 사례를 포함한다. 예시적으로, PID 10 및 11은 단일변수 원자성 위배 사례이다. PID 11에서, 프로그래머는 초기화가 한 번만 수행되는 것을 의도하였지만, 공유변수에 원자적으로 접근하지 않아 초기화가 2번 발생한 오류를 나타낸다.
- [0071] 종래의 원자성 위배 진단 기법은 AI 기반의 BSet()과 RPre()를 사용하여 원자성 위배를 진단하고 Wait(), Signal()을 통해 Thread를 제어하여 원자성 위배를 조치할 수 있다. 다른 예로, Labeling 기법과 공유변수의 접근 정보를 기반으로 원자성 위배를 진단하는 방식도 있으며, 원자성 위배가 진단되면 WAIT_EVENT(), SET_EVENT()를 통해 Thread를 제어하여 원자성 위배를 자율 수리한다.
- [0072] 종래의 기법은 비대칭 원자성 위배만 다루기 때문에 본 실험을 위해 수집된 원자성 위배 사례는 비대칭 원자성 위배가 아니기 때문에 임의로 하나의 Thread에 올바른 동기화 기법을 적용해 비대칭 원자성 위배가 발생하도록 설정한다. 또한, 종래의 원자성 위배 진단 기법과 다중변수 원자성 위배 자율 수리 장치(100)는 Thread 1이 공유변수에 모두 접근한 후 Thread 2가 접근하거나 Thread 2가 공유변수에 모두 접근한 후 Thread 1이 접근하는 의도한 BSet()과 Colored_BSet()만을 수집한다. 다중변수 원자성 위배 자율 수리 장치(100)는 프로그래머가 정의한 구조체를 통해 공유변수의 상관관계를 수집하기 때문에 다중변수 원자성 위배 패턴에서 상관관계에 있는 공유변수들은 하나의 구조체로 정의한다.
- [0073] 실험을 위한 코딩 환경은 Visual Studio Code 1.63.1을 사용하였다. 개발환경은 GCC 5.4.0 및 Clang 3.3 컴파일러를 사용하였으며, Low Level Virtual Machine (LLVM) 3.3을 사용하여 공유변수 접근사건 전/후로 진단 및 조치하는 함수를 코드를 삽입하였다. 코드 빌드 환경을 구성하기 위해 CMake 2.8.12를 사용하였다. 하드웨어 운영 환경은 Intel Xeon Silver 4216 CPU 2.10GHz CPU 및 64GB RAM을 사용하였다. 또한, Ubuntu 16.04.7 LTS-64bit 운영체제에 Linux 4.14.139-rt66 kernel과 ARINC 653 기능을 제공하는 시뮬레이터인 Simulated Integrated Modular Avionics (SIMA) 1.3.1.0을 구성하였다.
- [0074] 패턴에서 R은 읽기 사건, W는 쓰기 사건을 의미한다. “[”와 “]”은 lock과 같은 동기화를 의미하며, 이 괄호 사이의 사건들은 원자적 실행이 보장되어야 한다. “||”은 동시성 실행을 나타낸다.
- [0075] 도 6b는 도 6a의 실험 유형을 종래의 기법과 본 발명을 비교한 결과를 도시한 도면이다. 진단 및 조치할 수 있는 경우 “O”, 할 수 없는 경우는 “X”. 의도치 않게 진단 및 조치되는 경우는 “△”로 표기하였다. 그 결과, 종래의 기법들은 다중변수 원자성 위배를 진단할 수 없기 때문에 올바르게 조치할 수 없었다. 그에 반해, 본 발명은 다중변수 원자성 위배를 올바르게 진단 및 조치할 수 있었으며, 추가적으로 단일변수 원자성 위배도 자율 수리할 수 있었다.
- [0076] 도 7a 및 7b는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치를 활용한 시뮬레이션 결과 및 환경설정을 도시한 도면이다.
- [0077] SIMA에 적용하는 데 필요한 환경설정은 2개로 구분된다. 다중변수 원자성 위배 자율 수리 장치(100)를 적용하기 위한 첫 번째 단계는 CB(Colored_BSet)_Processing이다. CB_Processing은 먼저 파티션 내의 어플리케이션 코드를 정적 분석한 후 의도한 공유변수 접근사건 정보를 얻기 위한 코드를 삽입하여 사전 프로그램 수행을 진행한다. 이후 CB_Processing은 정적 분석을 통해 수집된 VColor와 사전 프로그램 수행을 통해 얻은 공유변수 접근사건 정보를 기반으로 Colored_BSet()을 정의한다. 두 번째 환경설정 단계는 MVR(Multi-Variable Repairing)_Processing이다. MVR_Processing은 먼저 파티션 내의 어플리케이션 코드에서 공유변수 접근사건 전/후로 다중변수 원자성 위배를 진단 및 조치하는 코드를 삽입한다. 그 후 SIMA와 Simout 도구를 사용하여 파티션 수행을 확인한다.
- [0078] 도 7b는 다중변수 원자성 위배 자율 수리 장치(100)를 제어하기 위한 명령어를 나타낸다. 도 7b의 (a)는 CB_Processing을 수행하기 위한 명령어들이다. 명령어 ①은 MV-Repairing_Prepare.cc 코드를 공유변수 접근사

전 전에 코드를 삽입하여 공유변수 접근 정보를 수집한다. 명령어 ②는 MV-Repairing_Trace.cc 코드를 삽입하고 SIMA 사전 수행에 필요한 SIMA의 라이브러리, Partition Operating System (POS)와 Pthread를 사용하기 위한 옵션을 링크한다.

- [0079] 도 7b의 (b)는 MVR_Processing을 수행하기 위한 명령어들이다. 명령어 ①은 공유변수 접근사건 전/후로 MV-Repairing_Tolerate.cc 코드를 삽입하여 다중변수 원자성 위배 진단 및 조치할 수 있도록 한다. 이후 SIMA 수행에 필요한 각종 라이브러리를 옵션으로 링크한다. 마지막으로 ②는 MV-Repairing을 적용한 파 티션을 SIMA 환경에서 수행하는 명령어로, simout 도구를 사용하여 SIMA의 파 티션 수행을 화면에 출력한다. /simout 뒤의 경로는 파티션의 스케줄링 정보 및 실행을 위한 바이너리 코드가 위치한 경로이다.
- [0080] 도 8a 내지 도 8f는 본 발명의 일 실시예에 따른 다중변수 원자성 위배 자율 수리 장치와 종래 기법의 효율성 비교를 위한 오버헤드 비교 도면이다.
- [0081] 도 8a 내지 8f를 통해서는 다중변수 원자성 위배 자율 수리 장치(100)와 종래 항공기 관리 기법(Repairing-AV)의 진단 과정에서 소요되는 시간 오버헤드를 비교한다. 도 8a는 오버헤드 비교 결과를 나타낸다.
- [0082] 도 8a에서 첫 번째 열은 공유변수 접근 횟수를 의미하고, 두 번째 열은 Repairing-AV의 수행 시간 측정 결과, 세 번째 열은 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)의 수행 시간 측정 결과, 마지막 열은 Repairing-AV에 비해 증가한 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)의 수행 시간 폭을 의미한다.
- [0083] 공유 변수 접근 횟수가 증가할수록 연산량이 증가하기 때문에 수행 시간이 증가하는 것은 자연스러운 일이다. 공유변수 접근 횟수가 50,000번인 경우 기존 연구인 Repairing-AV와 비교하면 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)의 수행 시간 증가 폭은 약 4.8%였다.
- [0084] 공유변수 접근 횟수가 5,000,000번으로 100배 증가하여도 기존 연구와 비해 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)의 수행 시간 증가 폭은 약 4.4%였으며, 공유변수 횟수가 15,000,000번으로 300배까지 증가하여도 수행 시간 증가 폭은 약 4.5%로 일정 하였다.
- [0085] 도 8a의 결과를 그래프로 표현한 결과 도 8b와 같다. 도 8b와 같이 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)과 Repairing-AV의 진단과정에 소요되는 시간 오버헤드 차이가 크지 않은 것을 볼 수 있다.
- [0086] 도 8c는 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)와 Repairing-AV의 조치 과정에 소요되는 시간 오버헤드를 비교한 결과이다. 그 결과 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)와 Repairing-AV는 동일한 조치 절차인 Wait(), Signal()을 수행하기 때문에 시간 오버헤드의 차이가 없다. 시간 증가 폭이 다른 이유는 시간을 측정하는 과정에서 오차가 발생할 수 있기 때문이다.
- [0087] 도 8c의 결과를 그래프로 표현한 결과 도 8d와 같다. 도 8d와 같이 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)와 Repairing-AV의 조치 과정에 소요되는 시간 오버헤드 차이는 거의 없었다.
- [0088] 도 8e는 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)와 Repairing-AV의 전체 시간 오버헤드를 비교한 결과이다. 총 시간 오버헤드는 프로그램 수행 시간, 진단 시간, 조치 시간이 모두 포함된 것이다. 공유변수 접근 횟수가 50,000번인 경우 기존 연구인 Repairing-AV와 비교하면 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)의 수행 시간 증가 폭은 약 2.3%였다. 공유변수 접근 횟수가 5,000,000번으로 100배 증가하여도 기존 연구와 비해 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)의 수행 시간 증가 폭은 약 2.2%였으며, 공유변수 횟수가 15,000,000번으로 300배까지 증가하여도 수행 시간 증가 폭은 약 2.5%로 일정 하였다.
- [0089] 도 8e의 결과를 그래프로 표현한 결과 도 8f와 같다. 도 8f와 같이 다중변수 원자성 위배 자율 수리 장치(100)(MV-Repairing)와 Repairing-AV의 시간 오버헤드 증가 폭은 크지 않은 것을 볼 수 있다.
- [0090] 진단에 소요되는 시간 오버헤드 증가 폭과 총 시간 오버헤드 증가 폭은 다르 다. 그 이유는 총 수행 시간은 원자성 위배 진단 및 조치 시간 외에 추가적인 프로그램 수행 시간이 포함되어 있기 때문이다. 실험에 사용된 합성 프로그램은 공유변수 접근 이외 추가적인 작업을 많이 수행하지 않기 때문에 프로그램 수행 시간 오버헤드가 높게 측정되었다. 공유변수 접근 이외의 추가적인 작업이 많아진다면 전체 프로그램 수행에 증가하는 시간 오버헤드 폭은 감소할 것이다.
- [0091] 요약하면 다중변수 원자성 위배 자율 수리 장치(100)는 기존 연구가 수리할 수 없는 다중변수 원자성 위배를 수

리하기 위한 추가적인 연산을 수행하기 때문에 시간 오버헤드가 증가하였다. 하지만, 공유변수 접근 횟수 50,000번에서 15,000,000번으로 300배 증가해도 평균 2.5%의 전체 시간 오버헤드 증가만으로 기존 연구가 수리할 수 없는 다중변수 원자성 위배를 추가적으로 수리할 수 있었다.

- [0092] 시간 오버헤드가 증가하는 이유는 다중변수 원자성 위배를 진단하기 위한 추가적인 작업을 수행하기 때문이다. 조치 과정은 관련 연구와 비해 증가하는 시간 오버헤드는 없음을 확인할 수 있다.
- [0093] 도 9는 본 발명의 일 실시예에 따른 항공기 건전성 관리 방법의 흐름을 도시한 도면이다.
- [0094] 도 9에 도시된 다중변수 원자성 위배 자율 수리 방법은 앞선 도1 내지 도 8f를 통해 설명된 다중변수 원자성 위배 자율 수리 장치(100)에 의하여 수행될 수 있다. 따라서 이하 생략된 내용이라고 하더라도 도 1 내지 도 8f를 통해 다중변수 원자성 위배 자율 수리 장치(100)에 대하여 설명된 내용은 도 10에도 동일하게 적용될 수 있다.
- [0095] 도 9를 참조하면, 단계 S110에서 진단부(101)는 항공기 제어를 위한 응용 어플리케이션 모듈(20)로부터 공유변수 상관관계 정보를 수집할 수 있다.
- [0096] 단계 S120에서 진단부(101)는 상기 공유변수 상관관계 정보를 포함하는 제1 접근 순서 정보와 실제 공유변수의 접근 순서에 기초한 제2 접근 순서 정보를 통해 오류를 진단할 수 있다.
- [0097] 여기서, 제1 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 사전에 의도된 프로그램 공유변수 접근 순서를 포함할 수 있다. 또한, 제1 접근 순서 정보는 응용 어플리케이션 모듈(20)의 다중변수 원자성 위배를 진단하기 위해 요구되는 공유변수 상관관계 정보를 포함할 수 있다. 제2 접근 순서 정보는 상기 공유변수 상관관계 정보와 연계된 실제 공유변수 접근 순서를 포함할 수 있다.
- [0098] 정리하면, 제1 접근 순서 정보는 공유변수 상관관계 정보와 공유변수 상관관계 정보가 포함된 사전에 의도된 프로그램 공유변수 접근 순서를 포함할 수 있다. 제2 접근 순서 정보는 공유변수 상관관계 정보가 포함된 실제 공유변수의 접근 순서를 포함할 수 있다.
- [0099] 진단부(101)는 미리 설정된 동기화 과정없이 복수의 스레드가 서로 상관관계를 가지는 복수의 공유변수에 접근하고, 스레드 중 적어도 어느 하나에 쓰기 작업이 포함된 경우 발생하는 다중변수 원자성 위배 오류를 검출할 수 있다. 즉, 진단부(101)는 복수의 공유변수에 대하여 제1 접근 정보와 상기 제2 접근 정보의 일치 여부에 기초하여 상기 다중변수 원자성 위배 오류를 검출할 수 있다.
- [0100] 구체적으로, 진단부(101)는 제1 접근 순서 정보의 사전에 의도된 프로그램 공유변수 접근 순서와 제2 접근 순서 정보의 실제 공유변수의 접근 순서를 비교하여 다중변수 원자성 위배 오류를 검출할 수 있다.
- [0101] 단계 S130에서 건전성 관리부(102)는 진단에 따라 검출된 오류를 수집할 수 있다.
- [0102] 단계 S140에서 오류 조치부(103)는 건전성 관리부(102)로부터 수신된 오류를 복구하기 위한 조치 명령을 결정할 수 있다. 오류 조치부(103)는, 검출된 오류와 관계된 스레드를 지연시키기 위한 조치 명령을 각 오류마다 결정할 수 있다. 하나의 오류와 관계된 스레드를 예시로 설명한다. 오류가 검출되면 오류 조치부(103)는 오류가 발생한 스레드를 멈출 수 있다. 이후 오류가 해결되면, 오류 때문에 멈춘 스레드가 다른 스레드가 수행되어도 정상적으로 수행할 수 있다고 진단되면 멈춘 스레드를 재개시킬 수 있다.
- [0103] 단계 S150에서 스레드 관리부(104)는 조치 명령에 기초하여 상기 응용 어플리케이션 모듈의 스레드를 제어할 수 있다. 또한, 스레드 관리부(104)는 오류별로 결정된 조치 명령에 기초하여 상기 오류와 관계된 스레드를 지연시킬 수 있다.
- [0104] 본 발명의 일 실시 예에 따른, 다중변수 원자성 위배 자율 수리 방법은, 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 본 발명의 동작을 수행하기 위해 하나 이

상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0105] 상술한 실시 예에 설명된 특징, 구조, 효과 등은 본 발명의 적어도 하나의 실시 예에 포함되며, 반드시 하나의 실시 예에 만 한정되는 것은 아니다. 나아가, 각 실시 예에 서 예시된 특징, 구조, 효과 등은 실시예들이 속하는 분야의 통상의 지식을 가지는 자에 의하여 다른 실시예들에 대해서도 조합 또는 변형되어 실시 가능하다.

[0106] 따라서 이러한 조합과 변형에 관계된 내용들은 본 발명의 범위에 포함되는 것으로 해석되어야 할 것이다. 또한, 이상에서 실시 예들을 중심으로 설명하였으나 이는 단지 예시일 뿐 본 발명을 한정하는 것이 아니며, 본 발명이 속하는 분야의 통상의 지식을 가진 자라면 본 실시예의 본질적인 특성을 벗어나지 않는 범위에서 이상에 예시되지 않은 여러 가지의 변형과 응용이 가능함을 알 수 있을 것이다. 예를 들어, 실시 예들에 구체적으로 나타난 각 구성 요소는 변형하여 실시할 수 있는 것이다. 그리고 이러한 변형과 응용에 관계된 차이점들은 첨부한 청구 범위에서 규정하는 본 발명의 범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

[0107] 20: 응용 어플리케이션 모듈

100: 다중변수 원자성 위배 자율 수리 장치

101: 진단부

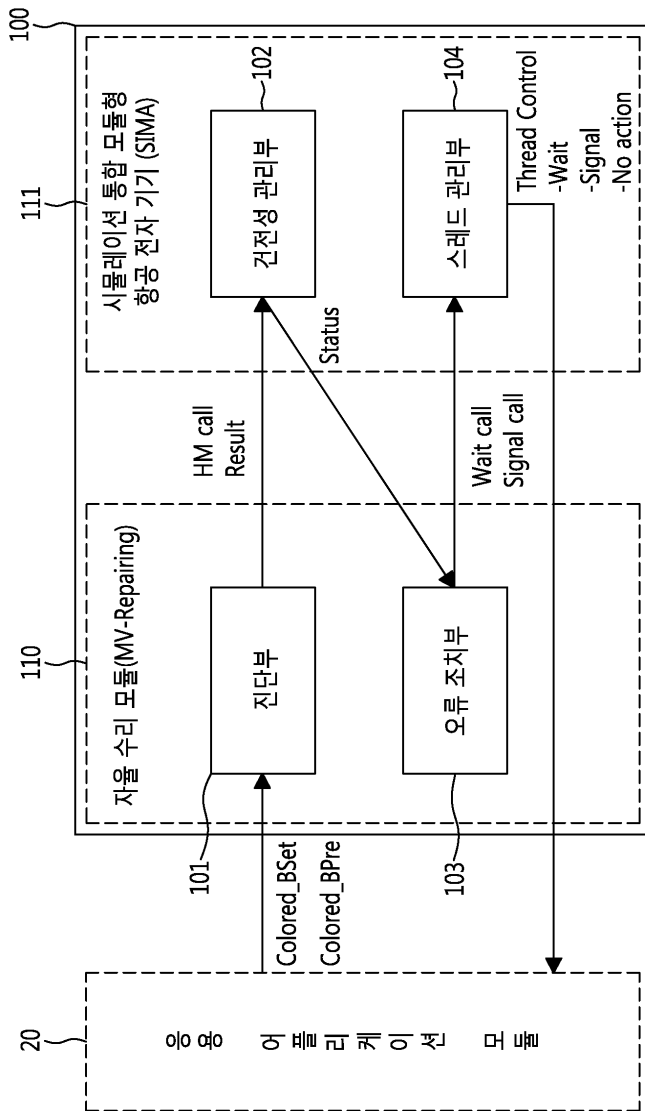
102: 건전성 관리부

103: 오류 조치부

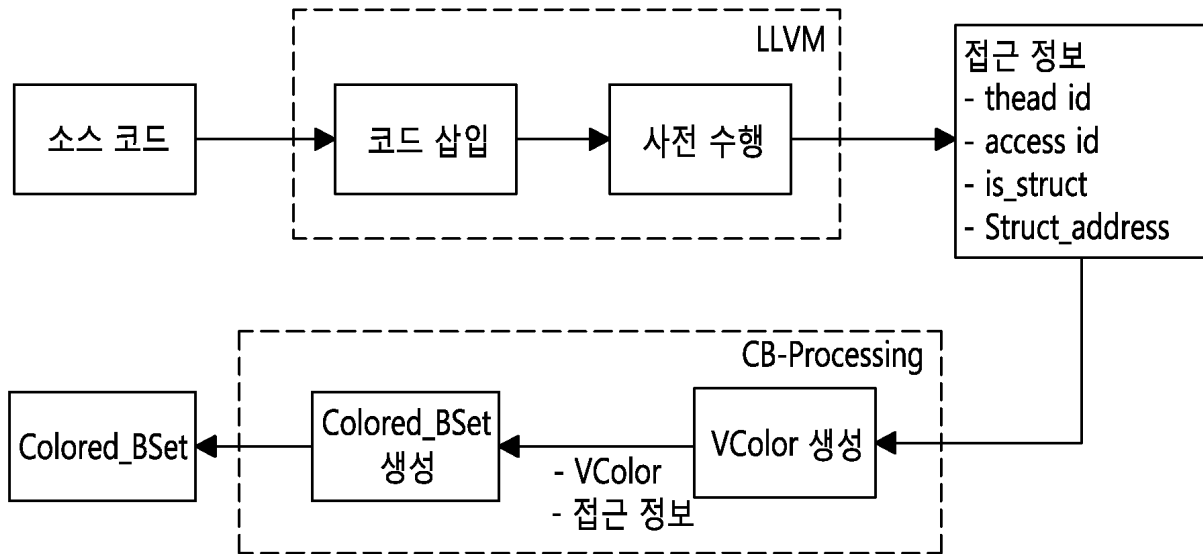
104: 스레드 관리부

도면

도면1



도면2



도면3

AP_NavEKF2_core	AP_NavEKF2_Outputs
Thread 1	Thread 2
<pre> void UpdateFilter(...) { ... selectMagFusion(...); // mag update selectVelposFusion(...); // vel, pos, hgt update ... selectTasFusion(...); // tas update selectFilterSatus(...); // off update } </pre>	<pre> int get_variances(...) { ... velVar = vel; posVar = pos; hgtVar = hgt; tasVar = tas; magVaR = mag; offset = off; ... } </pre>

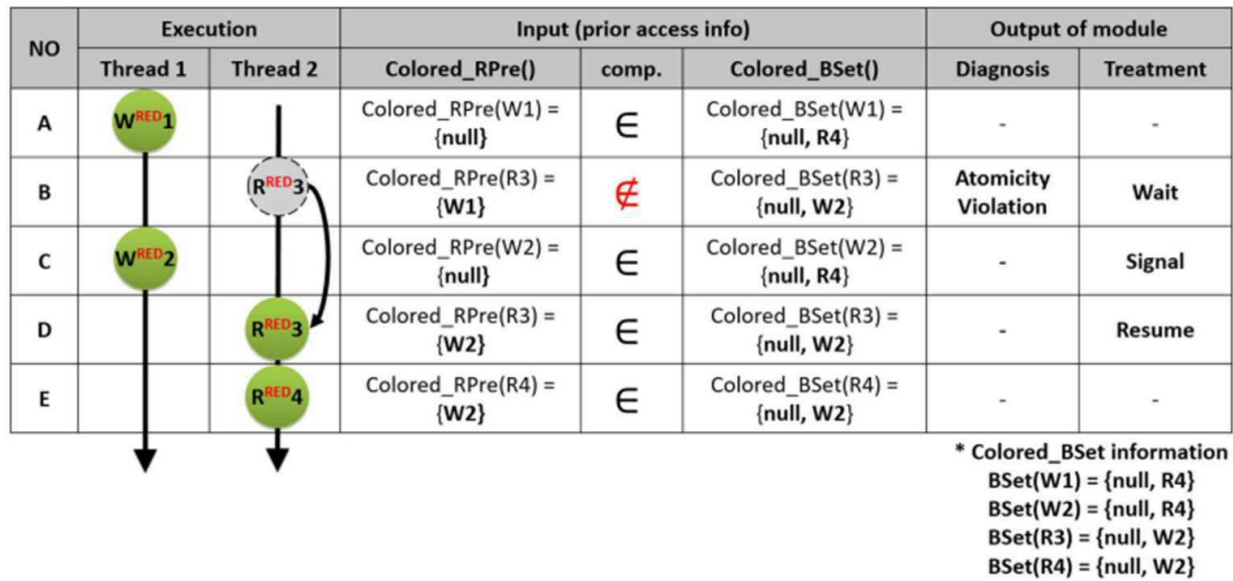
도면4

```

for.end:
call void @__ai_pre_diagnosis_atomicity_violations( ... ; preds = %for.cond
store i32 1, i32* @shared_variable, align 4, !AIMemoryAccessID !6
call void @__ai_post_diagnosis_atomicity_violations()
call void @__ai_calculate_time()
call void @__ai_pre_diagnosis_atomicity_violations( ...
store i32 1, i32* @shared_variable, align 4, !AIMemoryAccessID !7
call void @__ai_post_diagnosis_atomicity_violations()
  
```

→ Pre AV-DE Code
 → Shared Variable
 → Post AV-DE Code

도면5



도면6a

Type of Bug	PID	Type	Pattern	Describe
Multi-Variable Atomicity Violation	1	Common Software	$W_x - W_y \mid R_x - R_y$	MySQL#2011에서 발생한 오류로 log 파일과 log 파일의 이름이 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유변수 간의 일관성이 파괴된 오류
	2-1	Common Software	$W_x - W_y \mid R_y - R_x$	Apache#54962에서 발생한 오류로 Content 와 Content 의 길이가 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유변수 간의 일관성이 파괴된 오류
	2-2	Common Software		Apache#1507에서 발생한 오류로 string 과 string 의 길이가 원자적으로 접근되어야하지만, 원자적으로 수행되지 않아 공유변수 간의 일관성이 파괴된 오류
	3	Common Software	$R_x - W_x - W_y \mid R_x - R_y$	Mozilla#833361에서 발생한 오류로 active 와 check 변수가 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유변수 간의 일관성이 파괴된 오류
	4	Common Software	$W(3) \mid R(3)$	Mozilla#66856에서 발생한 오류로 mContent , mOffset , mLenght 변수가 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유 변수 간의 일관성이 파괴된 오류
	5	Common Software	$R_x - W_x - [R_y - W_y] \mid R_x - R_y$	S. Lu 등의 연구[16]에서 소개된 Mozilla에서 발생한 오류로 TotalStrings 과 lengthSum 변수가 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유변수 간의 일관성이 파괴된 오류
	6	Airborne Software	$W_x - R_x - W_y - R_x - W_z \mid W_x - R_x - W_y - R_x - W_z$	Ardupilot#13553에서 발생한 오류로 current , logger , notify mode 가 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유 변수 간의 일관성이 파괴된 오류
	7	Airborne Software	$W(6) \mid R(6)$	Ardupilot#13541에서 발생한 오류로 vel , pos , hgt , tas , msg , off 변수가 원자적으로 접근돼야 하지만, 원자적으로 수행되지 않아 공유변수 간의 일관성이 파괴된 오류
	8	Synthetic Program	$W_x - W_y \mid W_x - W_y$	UNICORN[20]이 탐지할 수 있는 다중변수 원자성 위배 접근 패턴으로 실제 발생한 패턴이 아니기 때문에 임의의 합성 프로그램으로 작성함
	9	Synthetic Program	$W_x - W_y \mid W_y - W_x$	
Single-Variable Atomicity Violation	10	Airborne Software	$R - W \mid R - W$	Ardupilot#7129에서 발생한 오류로 데이터 통신을 위한 초기화가 완료되면 init_done 변수가 true로 변경돼야 하지만, 원자적 수행되지 않아 초기화가 2번 발생하는 오류
	11	Airborne Software	$(R - W) \mid (R - W)$	Ardupilot#18213에서 발생한 오류로 장치 및 시리얼 초기화가 완료되면 _initialized 변수가 true로 변경돼야 하지만, 원자적 수행되지 않아 초기화가 2번 발생하는 오류

도면6b

Type of Bug	PID	Source	O.K. Ha et al.		G.M. Tchamgoue et al.		Repairing-AV		MV-Repairing	
			Diagnosis	Treatment	Diagnosis	Treatment	Diagnosis	Treatment	Diagnosis	Treatment
Multi-Variable Atomicity Violation	1	MySQL#2011	X	△	X	△	X	X	○	○
	2-1	Apache#54962	X	X	X	X	X	X	○	○
	2-2	Apache#1507	X	X	X	X	X	X	○	○
	3	Mozilla#833361	X	△	X	△	X	X	○	○
	4	Mozilla#66856	X	X	X	X	X	X	○	○
	5	S. Lu et al. [16]	X	△	X	△	X	X	○	○
	6	Ardupilot#13553	X	△	X	△	X	X	○	○
	7	Ardupilot#13541	X	X	X	X	X	X	○	○
	8	UNICORN [20] Test Case	X	△	X	△	X	X	○	○
	9		X	X	X	X	X	X	○	○
Single-Variable Atomicity Violation	10	Ardupilot#7129	○	○	○	○	○	○	○	○
	11	Ardupilot#18213	○	○	○	○	○	○	○	○

도면7a

```

<10 - one
<-----
<
<
<
<
<
<
<
< TraceInitialize!!
< With connected ports: 0
< mos: 2815
< pos: 2818
< cmd: 0
< err: 0
< tarage_process is created: 0000000707800000
< tarage_process is started: 0000000707900000
<
<-----
<0 - Test
<-----
< Error 10 handled by 4.
< My cmd string for partition one:
< ed: 149
< Executing: /bin/sh ./samples/experiment/ai/multi_3/partitions/one/one.sh -g=100000 -m=90010 -t.
< Partition one started
< [0000000013100000] Switching to 0101 for 5.000000000 seconds; assigned to partition 'one' in o
<-----

```

도면7b

```

$(MV-Repairing_DIR)/llvm/install/bin/clang -DTRIGGER ... (APP_NAME)
$(MV-Repairing_HOME)/build/tools/id-server ex.AccessInfo &
$(MV-Repairing_DIR)/llvm/install/bin/opt -load=$MV-Repairing_HOME/build/llvm/MV-Repairing/LLVMMV-
RepairingPrepare.so -MV-RepairingPrepare <(APP_NAME).bc>example-Marked.bc
$(MV-Repairing_DIR)/llvm/install/bin/opt -load=$MV-Repairing_HOME/build/llvm/MV-Repairing/LLVMMV-
RepairingTrace.so -MV-RepairingTrace <example-Marked.bc >example-Trace.bc
$(MV-Repairing_DIR)/llvm/install/bin/llc example-Trace.bc -o example-Trace.s
$(MV-Repairing_DIR)/llvm/install/bin/llc example-Marked.bc -o example-Marked.s
$(MV-Repairing_DIR)/llvm/install/bin/clang++ example-Trace.s -o example.trace -lai -lpthread -lrt -lm
/opt/sima/lib/pos.a /opt/sima/usr/noports.o /opt/sima/usr/nobooks.o entry.o err.o init.o
/opt/sima/usr/noscheds.o /opt/sima/usr/main.o

```

(a)

```

$(MV-Repairing_DIR)/llvm/install/bin/opt -load=$MV-Repairing_HOME/build/llvm/MV-Repairing/LLVMMV-
RepairingTolerate.so ...
$(MV-Repairing_DIR)/llvm/install/bin/llc example-Tolerate.bc -o example-Tolerate.s
$(MV-Repairing_DIR)/llvm/install/bin/clang++ example-Tolerate.s -o $1 -lai -lpthread -lrt -lm
/opt/sima/lib/pos.a /opt/sima/usr/noports.o /opt/sima/usr/nobooks.o entry.o err.o init.o
/opt/sima/usr/noscheds.o /opt/sima/usr/main.o

```

SIMA

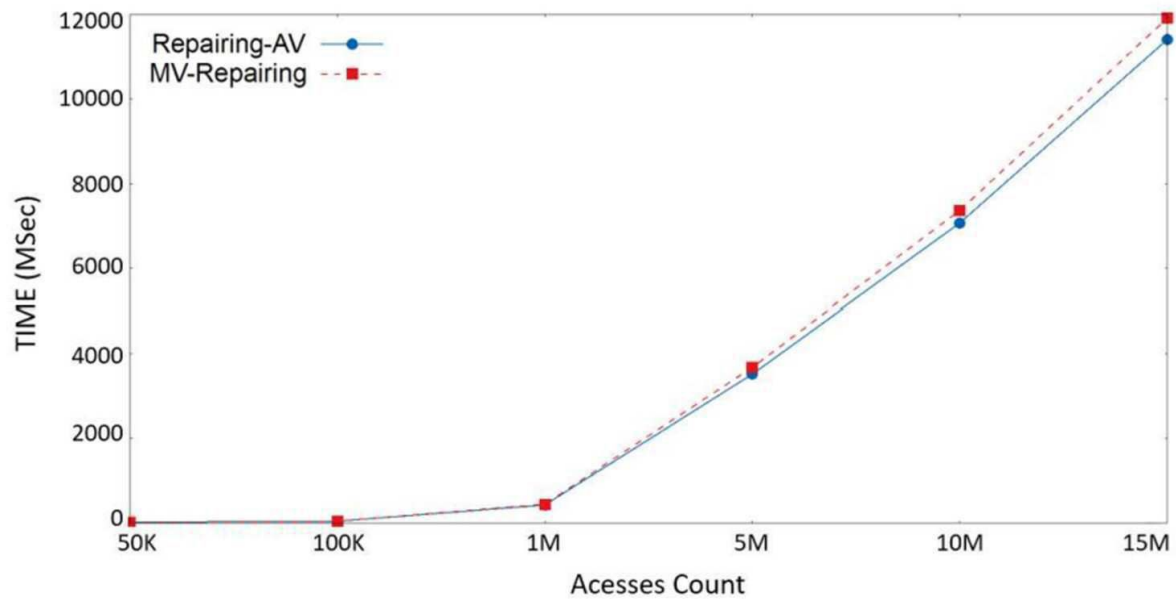
```
sudo bin/simout sample/experiment/config/sima.xml
```

(b)

도면8a

Accesses Count	Repairing-AV	MV-Repairing	MV-Repairing Increased CPU Time Overhead Compared to Repairing-AV
50,000	24.7 msec	25.95 msec	4.8%
100,000	47.4 msec	49.3 msec	4%
1,000,000	432.1 msec	450 msec	4.1%
5,000,000	3,523.3 msec	3,682.5 msec	4.5%
10,000,000	7,066.8 msec	7,366.1 msec	4.2%
15,000,000	11,402.6 msec	11,907.46 msec	4.4%

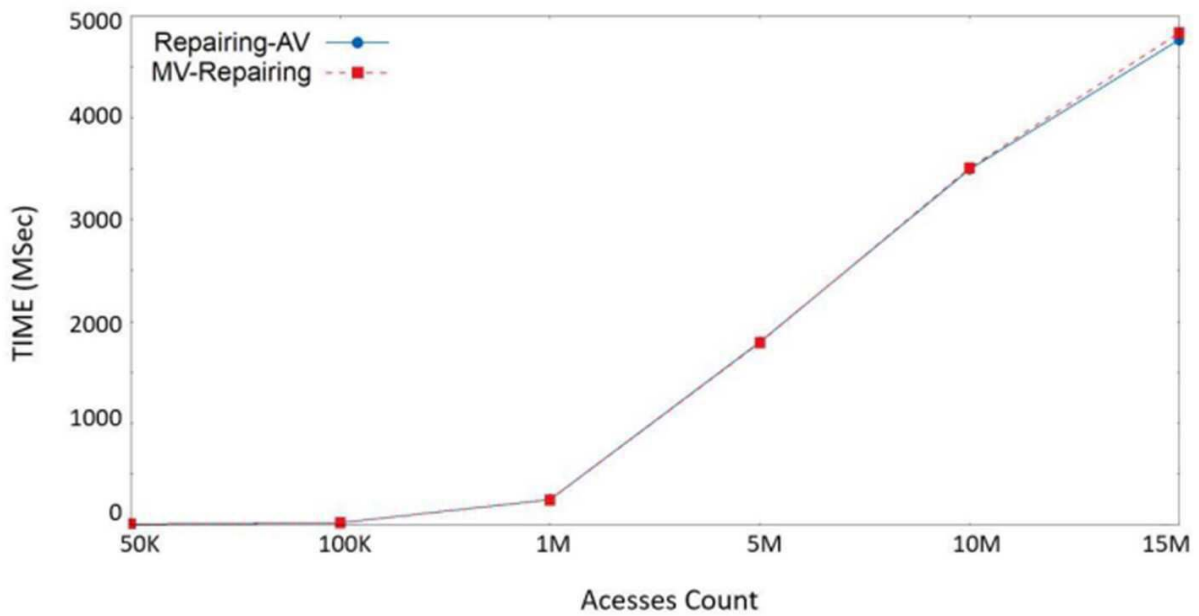
도면8b



도면8c

Accesses Count	Repairing-AV	MV-Repairing	MV-Repairing Increased CPU Time Overhead Compared to Repairing-AV
50,000	11.9 msec	11.6 msec	-1.7%
100,000	25.2 msec	25.8 msec	2.3%
1,000,000	252.13 msec	251.47 msec	-0.3%
5,000,000	1,797 msec	1,792.1 msec	-0.3%
10,000,000	3,492.1 msec	3,504.6 msec	0.3%
15,000,000	4,764.4 msec	4,830.8 msec	1.3%

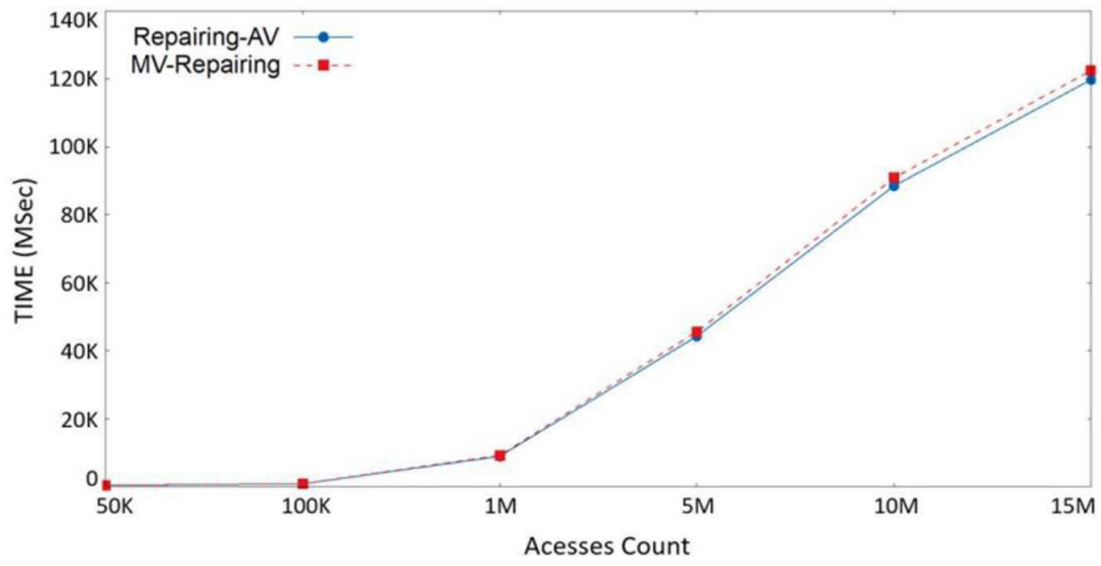
도면8d



도면8e

Accesses Count	Repairing-AV	MV-Repairing	MV-Repairing Increased CPU Time Overhead Compared to Repairing-AV
50,000	459.59 msec	470.42 msec	2.3%
100,000	907.65 msec	925.35 msec	1.9%
1,000,000	8,992.47 msec	9,252.11 msec	2.8%
5,000,000	44,299.37 msec	45,610.24 msec	2.9%
10,000,000	88,486.43 msec	90,833.06 msec	2.6%
15,000,000	119,731.28 msec	122,414.55 msec	2.2%

도면8f



도면9

