Journal of The Korea Society of Computer and Information
Vol. 19, No. 4, April 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.4.017

내포병렬성을 가진 공유메모리 프로그램에서 최초경합의 수행후 탐지도구

강 문 혜*, 심 갑 식*

A Post-mortem Detection Tool of First Races to Occur in Shared-Memory Programs with Nested Parallelism

Mun-Hye Kang*, Gab-Sig Sim *

요 약

본 논문에서는 고성능 컴퓨팅 시스템의 성능 향상을 위한 효율적인 동적 작업부하 균등화 정책을 제안한다. 이 정책은 시스템 자원인 CPU와 메모리를 효율적으로 사용하여 고성능 컴퓨팅 시스템의 처리량을 최대화하고, 각 작업의 수행시간을 최소화한다. 또한 이 정책은 수행중인 작업의 메모리 요구량과 각 노드의 부하 상태를 파악하여 작업을 동적으로 할당한다. 이때 작업을 할당 받은 노드가 과부하 상태가 되면 다른 노드로 작업을 이주시켜 각 노드의 작업부하를 균등하게 유지함으로써 작업의 대기시간을 줄이고, 각 작업의 수행시간을 단축한다. 본 논문에서는 시뮬레이션을 통하여 제안하는 동적 작업부하 균등화 정책이 기존의 메모리 기반의 작업부하 균등화 정책에 비해 고성능 컴퓨팅 시스템의 성능 향상 면에서 우수함을 보인다.

▸ Keywords : 공유메모리 병렬프로그램, 최초경합탐지, 수행후탐지기법

Abstract

Detecting data races is important for debugging shared-memory programs with nested parallelism, because races result in unintended non-deterministic executions of the program. It is especially important to detect the first occurred data races for effective debugging, because the removal of such races may make other affected races disappear or appear. Previous dynamic detection tools for first race detecting can not guarantee that detected races are unaffected races. Also, the tools does not consider the nesting levels or need support of other techniques. This paper suggests a post-mortem tool which collects candidate accesses during program execution and then

•제1저자 : 강문혜 • 교신저자 : 심갑식
•투고일 : 2013. 9. 10, 심사일 : 2013. 12. 31, 게재확정일 : 2014. 2. 21.
* 국립경남과학기술대학교 교양학부(Dept. of Liberal Arts, Gyeongnam National University of Science and Technology)
※ 이 논문은 2013년도 경남과학기술대학교 연구비 지원에 의하여 연구되었음.

detects the first races to occur on the program after execution. This technique is efficient, because it guarantees that first races reported by analyzing a nesting level are the races that occur first at the level, and does not require more analyses to the higher nesting levels than the current level.

▸ Keywords : Shared-memory parallel programs, First-race to occur, Post-mortem

I. 서 론

병렬 프로그램의 활용분야가 점차 확대되고 있으나, 높은 성능도 중요하지만 오류 없이 수행되는 프로그램의 신뢰성이 더욱 중요하다. 그러나 공유메모리 병렬 프로그램에서 발생한 오류를 수정하는 것은 오류의 발생을 식별하기 위해 프로그램의 병렬 수행 양상에 대한 이해가 전제되어야 하므로 어려운 일이다. 특히, 가장 심각한 병렬오류인 경합(the races)[1, 2, 3, 4]은 병행적으로 수행되는 스레드들이 공유변수에 대해 적절한 동기화 없이 적어도 하나 이상의 쓰기 사건으로 접근할 때 나타나며, 프로그램에서 의도하지 않은 비결정적 수행 결과를 초래하기 때문에 반드시 탐지되어 제거되어야 한다. 특히 경합 중에서 가장 먼저 발생하고, 다른 경합으로부터 영향을 받지 않은 최초경합(the first races to occur)[1, 5, 6, 7, 8, 9]을 탐지하는 것은 최초경합으로부터 영향받은 다른 경합들이 사라질 수 있거나 새로이 발생할 수 있기 때문에 더욱 중요하다. 또한 최초경합이 탐지된 지점까지는 경합이 발생하지 않았음을 보장할 수 있어 디버깅에 매우 효과적이다.

공유메모리 프로그램에서 경합을 탐지하는 기법은 탐지시점에 따라서 정적 분석기법과 동적 분석기법으로 나눌 수 있다. 정적분석 기법은 프로그램의 원시 코드만을 분석하여 대상이 되는 프로그램에서 발생할 가능성을 가진 모든 잠재적 경합들을 탐지하지만, 많은 false positives를 보고한다. 반면, 동적 분석 기법은 주어진 입력에 대해서 실제 수행 시 발생할 수 있는 경합들만을 탐지하기 때문에 정적분석 기법에 비해 적은 부정확한 경합(false positives)을 보고하지만 실행되지 않는 코드에 대해서는 탐지를 하지 않으므로 탐지되지 않은 경합(false negatives)를 보고하는 단점이 있다. 동적 분석기법은 다시 수행 후(post-mortem) 탐지기법과 수행 중(on-the-fly) 탐지기법으로 나눌 수 있다. 수행 중에 경합을 탐지하는 기법은 수행후 탐지기법에 비해서 추적파일을 위한 기억공간의 부담이 없으며, 공유 자료구조인 접근역사를 구성하여 수집된 후보사건들 간의 병행성 관계를 검사하여 최초경합을 보고한다. 그러나 내포병렬성을 가진 병렬프로그램[10, 11, 12, 13]에서 동적으로 최초경합을 탐지하는 기존의 기법[6, 7, 8, 9]들에 의해서 탐지된 최초경합은 탐지된 경합들 중에서 영향받지 않은 경합인지 발생할 수 있는 경합들 중에서 영향받지 않은 경합인지에 대한 명시가 명확하지 않다. 또한 내포병렬성을 가진 공유메모리 병렬프로그램에서 최초경합을 탐지하는 기존기법들은 경합들간의 영향관계에서 내포수준을 고려하지 않거나 다른 기법의 도움을 받아야만 최초경합을 탐지할 수 있다.

본 연구에서는 내포병렬성을 가진 병렬프로그램에서 최초경합을 탐지하는 기존의 기법들을 탐지하고 있는 최초경합의 종류를 공유변수의 수와 경합들간의 영향관계를 따지는 시점에 따라서 분류하여 재 정의한다. 그리고 프로그램 수행 중에 후보사건을 수집[16]하여 프로그램 수행 후에 수집된 정보를 분석하고 최초경합만을 보고하는 기법을 적용함으로써 최초경합만을 보고하는 수행후 탐지도구를 제시한다. 제시된 도구에서 사용하고 있는 수행 후 분석기법은 각 내포수준에서 수집된 사건들의 영향관계를 고려하여 최초경합을 탐지하므로, 최초경합이 보고된 그 내포수준까지는 그 경합이 영향받지 않은 것임을 보장한다. 이는 현재 내포수준에서 최초경합 탐지 여부를 결정할 수 있게 하므로, 상위 내포수준에 대한 재분석이 필요 없는 효율적인 탐지기법이다.

본 도구의 클라이언트 프로그램은 C# 언어로 구현되었으며 서버는 리눅스 운영체제의 Intel Xeon 듀얼 CPU 컴퓨터에 OpenMP 병렬프로그램[10, 11, 12, 13]의 컴파일을 위해서 Intel C/C++ 10버전을 설치하였다. 엔진 라이브러리는 C언어로 작성하고 병행성정보를 생성하기 위해서 NR-Labeling 기법[14] 을 사용하였다. 이러한 모델은 엔진 라이브러리가 아닌 프로그램 실행 시에 필요한 라이브러리 등도 모두 서버에 설치되어 있어야 한다는 문제점이 남아있다. 클

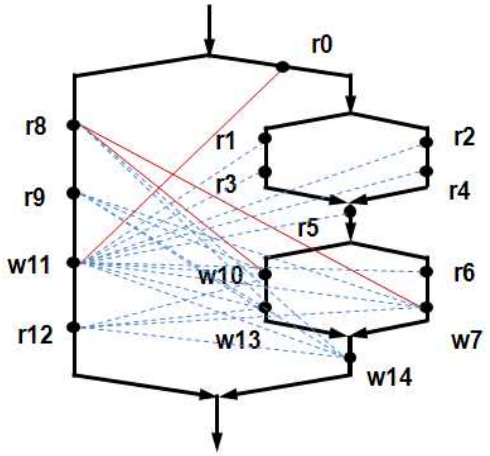

그림 1. 내포병렬성 프로그램과 최초경합
Fig. 1. Shared-Memory Programs and First Race to Occur

라이언트와 서버간의 파일을 전송하기 위해서 소켓프로그래밍이 구현되었으며, 디버깅하고자 하는 입력파일은 OpenMP C/C++ 프로그램만을 지원한다. 정확성 실험을 위해서 OpenMP 프로그램으로 프로그램의 내포깊이와 각 스레드 당 발생하는 접근사건의 타입과 내포수준을 달리하며 작성된 합성프로그램을 사용하였다.

2장에서는 공유메모리 병렬프로그램에서 존재하는 최초경합에 대해서 소개하고, 최초경합탐지 기법들에 대해서 살펴본다. 3장에서는 이들 기법을 공유변수의 수와 경합들간의 영향관계 분석시점에 따라 분류하고 본 연구에서 제시하는 기법을 보인다. 그리고 4장에서는 제안된 기법의 정확성 실험결과를 보이고, 마지막 장에서 결론을 제시한다.

II. 연구 배경

본 장에서는 내포병렬성이 있는 프로그램에서 최초로 발생하는 경합을 소개하고, 이러한 영향받지 않은 최초경합[1, 5, 6, 7, 8, 9]을 탐지하는 기존의 기법들을 살펴본다.

1. 병렬프로그램의 최초경합

본 논문은 내포병렬성을 가진 공유메모리 프로그램을 대상으로 한다. 병렬프로그램의 수행은 방향성이 있는 비순환 그래프인 POEG(Partial Order Execution Graph)[15]으로 나타낼 수 있다. POEG에서의 정점은 병렬 스레드의 생성과 합류를 나타내며, 정점들 사이의 간선은 정점에서 생성된 스레드를 나타낸다. 그리고 각 스레드에 접근하는 읽기와 쓰기 사건을 점으로 표시하고, r과 w로 나타내어 그 유형을 구분한다. 여기서 사용되는 숫자는 특정 수행 시에 접근사건들의 발생 순서를 의미한다. 그림 1은 내포병렬성을 가진 공유메모리 프로그램을 POEG으로 나타낸 것이다.

POEG은 병렬프로그램의 수행 시에 스레드들 간의 부분적 순서관계를 나타낸다. 두 사건사이에 경로가 존재하면 선행 관계가 있고, 두 사건이 순서화되었다고 한다. 그러나 경로가 존재하지 않으면 이 두 사건은 병행하다. 그리고 임의의 사건 ei가 ej보다 선행하고 {ei, ej}의 각 내포수준이 {m, n}일 때, m < n을 만족하면 ej가 ei에 내포된다고 한다. 그림 1에서 r0와 w7은 그들 사이에 경로가 존재하므로 r0는 w7에 선행하고, w7의 내포수준이 2로서 r0의 내포수준 1보다 크기 때문에 w7은 r0에 내포된다. 그러나 r9와 w10 사이에는 경로가 존재하지 않으므로 서로 병행하다.

이렇게 병행관계에 있는 두 개의 접근사건 ei, ej 중에 적어도 하나의 쓰기 사건을 포함하고 하나의 공유변수에 대해 적절한 동기화가 없이 나타날 때 발생하는 오류를 경합이라고 하며, ei-ej로 표시한다. 여기에서 임의의 사건 eh가 ej에 선행하고 eh가 경합 Rh에 포함되면, ej는 eh나 Rh에 의해 영향받은 사건이라고 한다. 임의의 경합을 구성하는 두 사건 {ei, ej}중에서 어느 사건도 다른 사건들에 의해 영향받은 사건이 아니면, 이러한 경합을 영향받지 않은 경합이라고 하고, 두 사건들 중에서 하나만 다른 사건에 의해 영향 받으면 부분적으로 영향받은 경합이라 한다. 이 때, 서로 부분적으로 영향받는 경합들이 두 개 이상인 경합집합을 얽힘이라 하고, 얽힘에 속한 임의의 경합을 얽힌 경합이라 한다. 또한 모든 접근사건들이 기껏해야 하나의 얽힌 경합에 영향받은 경합들의 집합인 얽힘을 최초얽힘[16]이라 한다.

최초얽힘 정리 임의의 공유변수에 접근하는 접근사건들간의 영향받지 않은 얽힘이 존재한다면, 이 얽힘에 포함되지 않는 최초경합은 존재하지 않는다.

이러한 영향받지 않은 경합이거나 최초얽힘을 최초경합이라 한다. 예를 들어, 그림 1에서 {r1-w11}은 {r0-w11}로부터 영향받은 경합이고, {r0-w11, r8-w10, r8-w7}만이 최초경합이며 이들은 얽힘을 이루고 있다. 이러한 최초경합을 탐지하여 제거하는 것은 이 경합에 영향받은 다른 경합들이 사라질 수 있거나 새로이 발생할 수 있기 때문에 매우 중요하다. 또한 최초경합이 탐지된 지점까지는 경합이 발생하지 않았음을 보장할 수 있어 디버깅에 매우 효과적이다.

2. 기존 연구

내포병렬성을 가진 공유메모리 병렬프로그램에서 최초경합을 탐지하는 기존의 기법은 크게 두 가지 유형으로 나눌 수 있다. 첫 번째는 이전의 경합탐지 도구들에 의해서 모든 공유변수를 대상으로 탐지된 경합들 중에서 영향받지 않은 경합을 탐지하는 기법이고, 두 번째는 실제 프로그램 수행에서 각 공유변수에 대해 발생할 수 있는 경합들 중에서 가장 먼저 발생하고 다른 경합에 영향받지 않은 경합을 탐지하는 기법이 있다.

모든 공유변수를 대상으로 탐지된 경합들 중에서 영향받지 않은 최초경합을 탐지하는 기법 중 Race Frontier 기법[6]은 프로그램 수행중에 최소(Minimal) 경합탐지 기법을 사용하여 탐지된 경합에 포함되는 접근사건을 경합경계(race frontier)로 설정하여, 이 사건까지 프로그램을 결정적으로 재수행하면서 최초경합을 탐지한다. 최소경합탐지 기법은 공유변수에 접근하는 최근의 읽기사건과 병행하는 쓰기사건을 유지함으로써 적어도 하나의 경합탐지를 보장한다. 이 기법은 프로그래머로 하여금 중단점 설정이나 반복 수행을 통한 경합탐지 작업을 추가로 요구하며, 특히 경합경계까지의 모든 경합을 재생산하기 위해서 예상할 수 없는 프로그램의 재수행 횟수를 요구한다. 또한 이 기법은 경합경계로 선택된 접근사건과의 경합만을 탐지하므로 얽힘을 포함하지 못하므로 얽힘을 포함한 최초경합의 탐지를 보장하지 못한다.

이 기법을 응용하고 있는 Recplay 기법[7]은 첫 번째 수행에서 동기화 사건만을 추적파일에 기록하고, 두 번째 수행에서 동기화 사건들 사이의 읽기와 쓰기사건 집합을 수집 및 비교하여 경합의 존재를 확인한다. 그리고 세 번째 수행에서 최초경합을 초래한 기계명령어의 위치탐지를 시도한다. 기계명령어의 위치를 프로그램 위치로 보고하기 위해서는 추가적인 보완이 필요하다. 이와 같이 이 기법은 최초경합을 탐지하기 위해서 최소한 세 번의 수행이 필요하다. 반면, 발생가능하고 영향받지 않은 경합이 적어도 하나는 포함된 첫 번째 그룹을 보고하는 기법은 프로그램 수행 중에 기존의 기법에 의해서 수집된 추적 정보들을 분석하여 영향받지 않은 경합을 탐지한다. 즉, 기존 기법에 의해 탐지된 사건들을 그룹화하여 순서 그래프를 생성하고, 공유데이타 의존성과 접근사건 의존성을 이용하여 그들 간의 순서관계와 발생여부를 판단한다. 따라서 영향받지 않고 발생가능한 경합으로 보고함으로써, 얽힘을 포함한 경우 영향받지 않은 경합을 모두 보고하지 못하는 문제점이 있다. 또한 이 기법은 이론적 증명은 하고 있지만, 이를 정확하게 수행할 효율적 알고리즘이 개발되어 있지 않다.

실제 프로그램 수행에서 각 공유변수에 대해 발생할 수 있는 경합들 중에서 가장 먼저 발생하고 다른 경합에 영향받지 않은 경합을 탐지하는 기법은 최초경합의 가능성이 있는 후보사건을 수집하는 기법[8]과 접근역사를 상수적 크기로 유지하는 효율적 기법[9]이 있다. 후보사건을 수집하는 기법은 두 번의 프로그램 수행으로 최초얽힘을 포함한 최초경합의 탐지는 보장하지만, 두 번의 프로그램 수행 중에 각 공유변수마다 프로그램의 최대병렬성에 의존적인 크기의 접근역사를 유지하므로 비효율적인 수행시간과 기억공간을 요구한다. 효율적 기법은 두 번의 프로그램 수행을 통해서 수행중에 각 공유변수에 대한 접근역사를 상수적 크기로 유지하여 각 접근사건의 수행시에 상수적 복잡도의 사건비교 횟수와 기억공간만을 요구하여 효율적이지만, 최초얽힘[16]이 포함된 경우 최악의 경우 최초얽힘에 참여하는 하나의 경합만을 최초경합으로 탐지하므로 최초경합의 존재를 검증하지 못한다. 또한 기존의 기법들이 탐지하는 최초경합의 종류가 현재 명확히 구분되어 있지 않은 문제점이 있다.

III. 최초경합탐지

내포병렬성을 가진 공유메모리 프로그램의 동적 자료경합 탐지를 지원하기 위해서 본 연구에서는 최초경합을 탐지하는 기존의 기법들에서 탐지하고 있는 최초경합의 종류를 공유변수의 수에 따라서 분류하여 재 정의하고, 프로그램 수행 중에 후보사건을 수집하여 프로그램 수행 후에 수집된 정보를 분석하고 최초경합만을 보고하는 기법을 적용함으로써 최초경합을 보고하는 수행후 탐지도구를 제시한다.

1. 최초경합탐지 분류

최초경합을 탐지하는 기존의 기법들이 탐지하는 최초경합의 종류가 현재 명확히 구분되어 있지 않으므로 탐지된 경합이 영향받지 않은 경합임을 보장할 수 없다. 본 연구에서는 탐지된 경합의 위치까지는 경합이 발생하지 않음을 보장하는 경합을 정확히 탐지할 수 있도록 최초경합의 종류를 공유변수의 수와 탐지된 경합의 성격에 따라서 분류하여 정의한다. 먼저, 모든 공유변수에 대해서 이전의 다른 기법에 의해서 탐지된 경합집합 중에서 영향받지 않은 경합을 국부적 최초경합이라고 하며, 프로그램 수행중에 임의의 공유변수에 대해 발생하는 모든 경합들 중에서 영향받지 않은 경합을 전역적 최초경합이라고 한다. 국부적 최초경합은 모든 공유변수를 위해 주어진 영향받지 않은 경합집합에서 부정확한 경합들을 포함

하지 않으며, 전역적 최초경합은 각 공유변수를 위한 영향받지 않은 경합집합 중에 false negatives를 포함하지 않는다. 본 연구에서 탐지하고자 하는 최초경합은 전역적 최초경합이다.

전역적 최초경합을 탐지하는 기법 중에서 내포병렬성과 순서적 동기화를 대상으로 경합에 포함될 가능성이 있는 후보사건을 수집하는 기법은 두 번의 프로그램 수행으로 최초경합을 탐지하므로 프로그램 수행 중에 각 공유변수마다 프로그램의 최대병렬성에 의존적인 크기의 접근사건을 유지하므로 비효율적인 수행시간과 기억공간을 요구한다. 또한 탐지된 최초경합이 최초읽힘을 포함하지만 내포수준까지 고려하여 경합들 간의 영향관계를 검사하지는 않으므로 내포된 읽힘이 최초경합일 경우에 최초경합이 아닌 경합까지 보고하므로 부정확한 경합을 포함한다. 두 번의 프로그램 수행 중에 효율적인 기억공간을 요구하는 기법은 프로그램의 첫 번째 수행에서 상수개의 외부사건을 이용하여 후보 최초경합 사건의 부분집합을 수집하고 두 번째 단계에서는 첫 번째 단계에서 수집된 사건들의 부분집합을 이용하여 후보 최초경합 사건의 집합을 완성한다. 이 기법에 의해서 탐지된 최초경합도 내포수준이 고려되지 않았으므로 최초경합이 아닌 경합까지 포함한다.

2. 최초경합탐지 도구

본 도구는 프로그램의 첫 번째 수행중에 최초경합 사건의 부분집합을 수집하고, 두 번째 수행중에 수집된 최초경합 사건의 부분집합을 이용하여 최초경합에 참여할 가능성이 있는 사건들을 감시한다. 이때 최초읽힘을 포함한 최초경합을 탐지하는데 필요한 모든 접근사건이 저장되면 이후에 발생된 사건은 감시하지 않고 프로그램을 종료한다. 마지막으로 프로그램 수행후에 그 사건들의 영향관계를 분석하여 최초경합만을 탐지한다. 이는 탐지시점까지는 프로그램에 경합이 존재하지 않음을 보장하므로 디버깅 비용을 감소시킬 수 있다.

그림 2는 본 도구의 구조를 나타낸 것으로 클라이언트 서버 모델로 구성된다. 이 모델은 전문가가 아닌 일반 사용자가 탐지를 위한 프로그램이 수행되어야 하는 환경 등을 고려하지 않아도 사용할 수 있는 장점이 있다. 클라이언트에서 디버깅하고자 하는 OpenMP 병렬프로그램[10, 11, 12, 13]이 입력되면 감시하고자 하는 공유변수를 선택하여 선택된 정보를 저장한다. 그리고 소스코드를 분석하여 최초경합 탐지 엔진을 호출하기 위한 감시코드를 소스코드에 삽입하고 삽입된 소스코드는 서버로 전송된다. 서버에서는 프로그램을 실행시켜 최초경합에 참여할 가능성이 있는 사건들을 감시하고, 수행후 그 정보들을 분석하여 최초경합만을 탐지한다. 그리고 그 결과를 클라이언트에 전송하고 사용자가 그 결과를 확인한다.

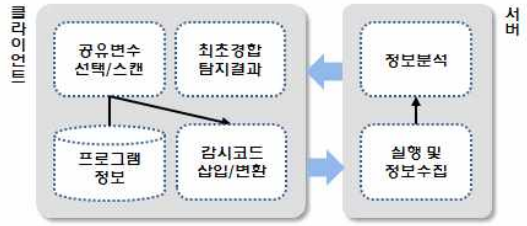

그림 2. 도구설계
Fig. 2. Design of the tool

수행중에 최초경합에 포함될 가능성이 있는 사건들을 수집하는 기법은 두 번의 수행으로 접근사건들 간의 선행관계와 좌우관계를 검사하여 내포수준을 고려하지 않고 최초읽힘을 포함한 영향받지 않은 최초경합에 포함되는 사건들을 사건의 유형별로 수집한다. 이렇게 수집된 사건들을 프로그램 수행후에 내포수준별로 확정된 최초경합과 하위내포수준에서 나타날 최초경합에 의해 영향받을 수 있는 미확정된 경합으로 구분하고, 확정된 경합과 영향받지 않은 미확정된 경합을 최초경합으로 결정한다. 따라서 디버깅 대상이 되는 내포병렬성을 가진 프로그램의 각 내포수준에서 최초경합을 탐지하면서 상위 내포수준에 대한 재분석이 필요 없으며, 탐지되면 더 이상 하위내포수준을 분석하지 않아도 되므로 효율적이다.

레이블링을 위한 감시코드의 삽입 위치는 병행 디렉티브가 시작되는 코드의 전과 끝나는 코드의 뒷부분이며, 접근사건들을 감시하기 위한 코드는 지정된 공유변수에 읽기사건 또는 쓰기사건으로 접근하는 코드들 뒤에 읽기사건 감시를 위한 코드 또는 쓰기사건 감시를 위한 코드로 삽입된다. 모든 Init() 라이브러리는 스레드들 간의 경합을 탐지하기 위한 초기환경

```
typedef struct Access_History_ {
    Label_NR *Left_r;
    Label_NR *Right_r;
    Label_NR *Write;
    char name[100];  char type[100];  char mode;
} Access_History;

typedef struct Candidate_History_ {
    Label_NR *Left_r;
    Label_NR *Right_r;
    Label_NR *Left_w;
    Label_NR *Right_w;
} Candidate_History;
```

그림 3. 수집알고리즘 라이브러리의 자료구조
Fig. 3. Data Structure of Collector Algorithm

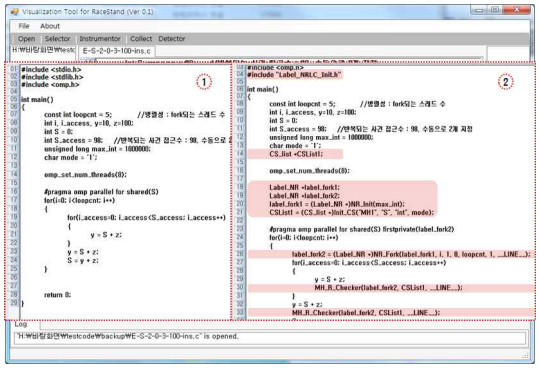

그림 4. 도구 인터페이스
Fig. 4. Interface of the tool

을 설정하는 것으로 각 기법에서 사용되고 있는 공유자료 구조인 접근역사 등의 기억공간을 할당하고 초기화한다. 그림 3에서 두 번의 프로그램 수행으로 경합사건을 수집하는 기법의 접근역사와 후보사건 역사의 자료구조를 보인다. 접근역사 구조체 Access_History는 두 개의 읽기사건과 하나의 쓰기사건으로 구성되며, 후보사건역사 Candidate_History는 두 개의 읽기사건과 두 개의 쓰기사건이 저장되게 구성되어 있다. 각각의 항목에는 저장된 접근사건이 포함된 레이블링 정보들이 저장되므로 NR 구조체로 정의된다.

도구의 인터페이스는 그림 4에서 보이며, 도구는 메뉴바, 도구바, 작업창, 그리고 결과창으로 구성된다. 메뉴는 디버깅하고자 하는 프로그램을 입력받는 file 메뉴가 있으며 도구는 파일을 오픈하는 open, 감시하고자 하는 공유변수를 선택하는 selector, 그리고 감시 라이브러리를 삽입하는 instrumentor, 최초경합 탐지를 위해 프로그램 수행중에 경합사건들을 수집하는 collector, 최초경합탐지를 위한 detector 등으로 구성된다. collector는 감시코드가 삽입된 소스 프로그램을 서버로 보내어 엔진라이브러리를 포함하여 컴파일한 실행파일을 생성하고 최초경합사건들을 수집하고, detector는 최초경합탐지 프로그램을 수행시켜 탐지된 정보를 보여준다. 확대된 창의 왼쪽 ①은 디버깅할 프로그램의 원본이며, ②는 감시코드가 삽입된 프로그램이다.

3. 구현 및 실험

본 도구의 프로그램은 서버클라이언트 모델로 클라이언트 프로그램은 C# 언어로 구현되었으며, 서버는 리눅스 운영체제하의 4개의 Intel Xeon CPU를 사용하는 컴퓨터에서 대상 프로그램 모델인 OpenMP 3.0을 위해 Intel C/C++ 10버전을 설치하였으며, 엔진 라이브러리는 C언어로 작성하고 병

표 1. 정확성 실험결과
Table 1. Accuracy result

	합성 프로그램 유형			탐지된 최초경합
1	r1	r2		r1-w4
	r3	w4		
2	r1	r2		r1-w5, r4-w5
	r3	r4	w5	
3	r1	w2		r1-w2
	r3	r4	w5	
4	w1	r2		w1-r2
	r3	r4	r5	
5	r1	r2		r1-w5, w5-r6
	r3	r4		
	r7	w5	r6	
6	r1	r2		r1-w4
	r3	w4		
	r7	r5	r6	
7	r1	r2	r3	w4-r2, w4-r3
	w4	r5	r6	
8	r1	r2	r3	r1-w6, r3-w4
	w4	r5	w6	
9	r1	r2		r1-w4, r1-w5, w4-w5
	r3	w4	w5	
10	r1	r2		r1-w4, r1-w5, r2-w3
	w3	w4	w5	
11	r1	r2		r1-w5, r4-w5
	r3	r4	w5	
	r6	r7		
12	r1	r2		r1-w7
	r3	r4	r5	
	r6	w7		
13	r1	r2	r3	w6-r2, w5-r3
	r4	w5	w6	
14	r1	r2		w4-r1, r5-w4
	r3	w4	r5	
			w6	
15	r1	r2		w4-r1, w5-r1, w4-w5
	r3	w4	w5	
16	r1	r2		r4-w5, r3-w6
		r3	r4	
		w5	w6	

행성 정보를 생성하기 위해서 NR-Labeling 기법을 사용한다. 이러한 모델은 엔진 라이브러리가 아닌 프로그램 실행 시에 필요한 라이브러리 등도 모두 서버에 설치되어 있어야 한다. 클라이언트와 서버간의 파일을 전송하기 위해서 소켓프로

그래밍이 구현되었으며, 디버깅하고자 하는 입력파일은 OpenMP C/C++ 프로그램만을 지원한다.

구현된 기법들은 소스프로그램에 수동으로 삽입하며, 정확성 실험을 위해서 OpenMP 프로그램으로 내포깊이와 각 스레드 당 발생하는 접근사건의 타입을 달리하며 작성한 합성프로그램을 사용한다. 표 1은 정확성 실험을 위한 16개의 합성프로그램의 유형 중에서 탐지된 최초경합을 나타낸 것이다. 표의 열은 POEG에서의 각 스레드를 나타낸 것이고 그 스레드에서 발생한 사건을 셀에 표기하였다. 합성프로그램은 스레드 2개와 3개인 경우로 내포와 쓰기사건의 수 및 위치를 달리한 경우의 수로 내포얽힘을 포함한 최초얽힘이 발생되게끔 의도하여 만들었다. 첫 번째 프로그램은 스레드가 2개이고 내포사건이 없는 유형으로 쓰기사건이 하나 발생하였다. 두 번째 프로그램은 두 번째 스레드에서 r2사건이 발생하고 내포수준 2에서 쓰기사건과 읽기사건 그리고 읽기-쓰기사건이 수행된다. 기존의 기법[9]으로 합성프로그램을 수행한 결과 8(r1-w6, r2-w4, r2-w6, r3-w4), 13(w4-r1, w4-r3, w5-r1, w5-r2), 14(w4-r1, r5-w4, w6-r1), 16(r1-w5, r4-w5, r3-w6, r1-w6)의 프로그램에서 얽힌경합에 영향받은 경합을 최초경합으로 보고한 반면, 본 도구에서 수행한 결과 내포수준에서 발생한 최초얽힘을 포함한 최초경합도 정확하게 탐지함을 확인할 수 있었다.

IV. 결 론

동적으로 최초경합을 탐지하는 기존의 기법들은 탐지된 경합이 발생할 수 있는 경합들 중에서 영향받지 않은 경합인지에 대한 명시가 명확하지 않으며, 경합들간의 영향관계에서 내포수준을 고려하지 않거나 다른 기법의 도움을 받아야만 최초경합을 탐지할 수 있다. 따라서 본 논문에서는 최초경합을 탐지하는 기존의 기법들을 탐지하고 있는 최초경합의 종류를 공유변수의 수에 따라서 분류하여 재 정의하고, 프로그램 수행 중에 후보사건을 수집하여 프로그램 수행 후에 수집된 정보를 분석하고 최초경합만을 보고하는 기법을 적용함으로써 최초경합만을 보고하는 도구를 제시하였다. 제시된 도구는 다른 도구의 도움없이 디버깅하고자 하는 프로그램에서 영향받지 않은 최초경합을 탐지할 수 있으며, 상위 내포수준에 대한 재분석이 필요 없는 효율적인 탐지도구이다. 경합탐지의 결과를 텍스트가 아닌 시각적으로 표현하기 위한 기법적용과 경합탐지 및 실행까지 모두 사용자 컴퓨터에서 수행할 수 있도록 하는 등이 향후과제로 남아있다.

참고문헌

[1] Netzer, R. H. B., and B. P. Miller, "What Are Race Conditions? Some Issues and Formalizations," Letters on Program Languages and Systems, 1(1): 74-88, ACM, March 1992.

[2] Wester, B., D. Devecsery and P. M. Chen, "Parallelizing Data Race Detection," The International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), ACM, Houston, Texas, USA., March 2013.

[3] Banerjee, U., B. Bliss, Z. Ma, and P. Petersen, "A Theory of Data Race Detection," Workshop on Parallel and Distributed Systems: Testing and Debugging (PADTAD), pp. 69-78, Int'l Symposium On Software Testing and Analysis (ISSTA), ACM, Portland, Maine, July 2006.

[4] Adve, S., M. Hill, B.Miller, and R. Netzer. Detecting data races on weak memory systems. In proceedings Of the Annual Int'l Symposium On computer Architecture (ISCA'91), pp. 234-243, May 1991.

[5] Kim, J., and Y. Jun, "Scalable On-the-fly Detection of the First Races in Parallel Programs," Proc. of the 12nd Int'l Conf. on Supercomputing (ICS), pp. 345-352, ACM, Melbourne, Australia, July 1998.

[6] Choi, J., and S. Min, "RACE FRONTIER: Reproducing Data Races in Parallel Program Debugging," 3rd Symposium on Principles and Practice of Parallel Programming (PPoPP91), pp. 145-154, ACM, Williamsburg, Virginia, April 1991.

[7] Ronsse, M., K. De Bosschere, "RecPlay: A Fully Integrated Practical Record/Replay System," Transacsion On Computer Systems, 17(2): 133-152, ACM, May 1999.

[8] Park, H., and Y. Jun, "Detecting the First Races in Parallel Programs with Ordered Synchronization," 6th Int'l Conf. on Parallel

and Distributed Systems, pp. 201-208, IEEE, Tainan, Taiwan, Dec. 1998.
[9] Ha, K., Y. Jun, and K. Yoo, "Efficient On-the-fly Detection of First Races in Nested Parallel Programs," Workshop on State-of-the-Art in Scientific Computing (PARA), pp. 75-84, Copenhagen, Denmark, June 2004.
[10] Ayguade, E., M. Gonzalez, X. Martorell, and G. Jost, "Employing Nested OpenMP for the Parallelization of Multi-zone Computational Fluid Dynamics Applications," In Proceedings of the 18th Int'l Conference on Parallel and Distributed Processing Symposium (IPDPS), pp. 6-15, IEEE, April 2004.
[11] Blikberg, R., and T. Sørevik, "Nested parallelism: Allocation of processors to tasks and OpenMP implementation," In Proceedings of the 2nd Int'l Conference on European Workshop on OpenMP (EWOMP), Edinburgh, Scotland, UK, Springer-Verlag, August 2000.
[12] Bücker, H. M. , A. Rasch, and A. Wolf, "A Class of OpenMP Applications Involving Nested Parallelism," In Proceedings of the 19th ACM Symposium on Applied Computing (SAC), 1: 220-224, Nicosia, Cyprus, New York, ACM, March pp. 14-17, 2004.
[13] Chandra, R., L. Daum, D. Ko hr, D. Maydan, J. McDonald, and R. menon, Parallel Programming in OpenMP, Academic Press, 2001.
[14] Jun, Y. and K. Koh, " On-the-fly Detection of Access Anomalies in Nested Parallel Loops," 3rd Workshop on Parallel and Distributed Debugging, pp. 107-117, ACM, May, 1993.
[15] Dinning, A., and E. Schonberg, "An Empirical Comparison of Monitoring Algorithms for Access Anomaly Detection," 2nd Symposium on Principles and Practice of Parallel Programming (PPoPP), pp.1-10, ACM, March 1990.
[16] M. Kang, and Y. Jun., "Efficient Verification of First Tangled Races to Occur in Programs with Nested Parallelism," In Proceedings of the ASEA 2011, CCIS 257, pp. 437-450, Springer, Heidelberg, 2011.

저 자 소 개

강 문 혜
2001: 국립경상대학교 컴퓨터과학과 이학사.
2003: 국립경상대학교 컴퓨터과학과 공학석사.
2011: 국립경상대학교 컴퓨터과학과 공학박사
현 재: 국립경남과학기술대학교 교양학부 강사
관심분야: 컴퓨터시스템, 병렬컴퓨팅
Email : kturtle@hanmail.net

심 갑 식
1985: 전남대학교 계산통계학과 이학사.
1987: 전남대학교 계산통계학과 이학석사.
1993: 전남대학교 전산통계학과 이학박사
현 재: 국립경남과학기술대학교 교양학부 교수
관심분야: 유비쿼터스 컴퓨팅, 정보보안
Email : gssim@gntech.ac.kr