OpenMP 프로그램을 위한 경합디버깅 환경의 자동생성*

강문혜,° 김영주, 전용기
경상대학교 컴퓨터과학과
{turtle, yjkim, jun}@race.gsnu.ac.kr

Automatic Generation of Race Debugging Environment for OpenMP Programs*

Moon-Hye Kang,° Young-Joo Kim, Yong-Kee Jun
Dept. of Computer Science, Gyeongsang National University

요 약

공유메모리 병렬프로그램에서 경합은 프로그램의 비결정적인 수행을 초래하므로 디버깅을 위해서 반드시 탐지되어져야 한다. 경합을 수행 중에 디버깅하기 위한 기존의 도구들은 경합탐지 엔진과 시각화 엔진으로 구성된 경합디버깅 엔진을 특정 프로그램 모델에 의존적으로 적용하여 경합디버깅 환경을 구성한다. 이러한 도구들은 프로그램 모델의 변경 시에 경합디버깅 환경이 최적의 경합디버깅 엔진으로 구성되지 못한다는 문제점이 있다. 본 논문에서는 OpenMP 병렬프로그램에서 각 프로그램 모델에 따라 효과성, 효율성, 확장성 등을 고려한 경합탐지 엔진과 추상성을 고려한 시각화 엔진으로 경합디버깅 환경을 자동으로 생성하는 도구를 제안한다. 이 도구는 디버깅 대상이 되는 프로그램의 모델에 최적인 경합디버깅 엔진을 적용하므로 경합탐지 목적에 부합하는 최적의 성능과 효과적인 시각화를 제공한다. 따라서, 본 도구는 디버깅 환경을 생성하기 위한 부담을 줄여서 효과적인 디버깅을 할 수 있게 한다.

1. 서 론

공유메모리 병렬프로그램에서의 오류인 경합[11]은 병행적으로 수행되는 쓰레드들이 공유변수에 대해 적절한 동기화 없이 적어도 하나 이상의 쓰기 사건으로 접근할 때 나타난다. 이러한 경합은 프로그램상에 의도하지 않은 비결정적 결과를 초래하므로 반드시 탐지되어야 한다.

경합을 탐지하는 기법은 탐지 시점에 따라, 정적분석(static analysis)기법[1], 사후추적(post mortem analysis)기법[5], 수행중 탐지(on-the-fly detection)기법[3, 6]이 있다. 그 중에서 수행중 경합탐지 기법은 상대적으로 시간적·공간적 부담이 적으며, 적어도 하나의 경합탐지를 보장하지만, 프로그램 모델에 따라 다양한 기능 및 성능을 가진다. 경합을 수행 중에 디버깅하기 위해서는 프로그램 모델의 특성에 부합되는 경합탐지 엔진과 시각화 엔진으로 구성된 경합디버깅 엔진이 필요하다. 이러한 기존의 도구들은 특정 프로그램 모델에 의존적으로 경합디버깅 엔진을 적용하여 경합디버깅 환경을 구성하므로, 프로그램 모델의 변경 시에 경합디버깅 환경이 최적의 경합디버깅 엔진으로 구성되지 못한다는 문제점이 있다.

본 논문에서는 OpenMP[2] 병렬프로그램을 대상으로 수행 중에 나타나는 경합을 탐지하기 위해 효과성, 효율성, 확장성 등을 고려한 경합탐지 엔진과 추상성을 고려한 시각화 엔진으로 경합디버깅 환경을 자동으로 생성하는 도구를 제안한다. 제안된 도구는 디버깅의 대상이 되는 프로그램의 모델이 변경되더라도 최적의 경합디버깅 엔진을 선택하여 자동으로 적용시켜주므로 경합디버깅 시에 탐지목적에 따른 높은 성능과 효과적인 시각화를 제공한다. 따라서, 사용자가 디버깅 환경을 생성하기 위한 부담을 줄여서 효과적인 디버깅을 할 수 있게 한다.

본 도구의 경합탐지 엔진은 경합탐지의 목적에 따른 효과성, 시간적·공간적 효율성, 공유변수에 발생하는 병목현상을 줄이기 위한 확장성을 고려한 기법들을 사용한다. 또한, 시각화 엔진은 임계구역과 내포 병렬성만을 대상으로 하는 추상성을 제공한다. 이러한 엔진들로 구성된 경합디버깅 환경의 수행을 위해서 Omni-1.3 OpenMP 컴파일러를 사용하고, 제안된 도구는 플랫폼에 독립적이고, 유지보수 및 개선 등의 부담을 없애기 위해서 웹기반 인터페이스를 사용한다.

2절에서는 본 연구의 배경으로 수행중 경합탐지 기법과 경합디버깅 환경을 살펴보고, 3절에서는 제안된 자동생성 기법과 그 기법을 적용한 도구를 설명한다. 마지막으로 4절에서는 결론 및 향후 과제를 제시한다.

2. 연구 배경

본 절에서는 대상으로 하는 OpenMP 병렬프로그램에서 나타나는 경합을 탐지하는 기법들 중에서 수행중 경합탐지기법을 살펴보고 경합탐지 엔진과 시각화 엔진으로 이루어진 경합디버깅 환경에 대해서 살펴본다.

2.1 수행중 경합탐지

OpenMP 표준 API[2]는 공유 메모리를 사용하는 병렬 프로그래밍 모델로써 표준 C/C++와 Fortran 77/90을 확장하는 디렉티브(directive)와 런타임 라이브러리(run-time library) 루틴들의 집합이다. 디렉티브에는 "parallel do" 등의 병렬화 디렉티브와 "barrier", "critical section" 등의 동기화 디렉티브가 있으며, 라이브러리는 병렬프로그램 수행 시에 필요한 쓰레드의 수를 결정하는 기능 등의 실행환경 루틴과 잠금(lock)을 위한 루틴 등이 있다. 이러한 OpenMP 병렬 프로그램을 작성하고, 프로그램 상에서 발생하는 오류를 수정하는 것은 어렵다. 특히,

* 본 연구는 정보통신부에서 지원하는 대학기초연구지원 사업으로 수행되었음

가장 심각한 병렬오류인 경합은 병행적으로 수행되는 쓰레드들이 공유변수에 대해 적절한 동기화 없이 적어도 하나 이상의 쓰기 사건으로 접근할 때 나타나며, 이는 프로그램 상에 의도하지 않은 비결정적 결과를 초래하므로 반드시 탐지되어야 한다. 이러한 경합들 중에서 가장 먼저 발생하는 것을 최초경합[7, 9, 12]이라 하고, 이를 탐지하여 수정하면 그 후에 발생한 경합들이 제거되거나 새로운 경합들이 발생할 수 있다.

수행중 경합탐지 기법[3, 6]은 프로그램 수행 중에 공유변수들에 대한 스레드들의 접근을 감시하므로 정적분석기법[1]에 비해 실제적인 경합만을 보고하고, 사후추적기법[5]에 비해 불필요한 정보가 제거되어 상대적으로 탐지의 효율성이 우수하다. 또한, 각 공유변수에 대해 적어도 하나의 경합을 탐지하는 것을 보장한다. 수행중 경합탐지 기법은 병렬프로그램의 수행 중에 접근사건의 발생 시마다 공유자료구조인 접근역사를 유지하며, 접근사건들의 논리적 병행성을 검사하여 경합을 보고한다. 수행 중인 각 스레드의 병행성을 식별 가능한 고유정보로 할당하는 기법이 레이블링(labeling)[6, 14]이다. 이러한 레이블링을 이용하여 접근사건 수행 시마다 이전에 발생한 접근사건들과의 병행관계를 검사하여 경합을 탐지하고, 접근역사를 유지하는 기법을 경합탐지 프로토콜(protocol)[4, 9, 10, 12]이라 한다. 또한, 경합탐지 시에 공유변수에 발생하는 병목현상을 줄이기 위해서 경합의 가능성이 있는 접근사건들만 선택하여 확장적 감시를 가능하게 하는 기법을 사건선택기법(filtering)[8, 9 ,13, 15]이라 한다. 그리고, 이 기법들은 프로그램 모델에 따라서 다양한 기능 및 성능을 가지므로 특정 프로그램 모델을 위한 최적의 기법을 선택하는 것은 쉽지 않다.

2.2 경합디버깅 환경

경합을 수행 중에 디버깅하기 위해서 프로그램 모델의 특성에 부합되는 경합탐지 엔진과 시각화 엔진으로 구성된 경합 디버깅 엔진을 적용하여 경합디버깅 환경을 구성한다. 경합탐지 엔진은 수행중 탐지기법에 의해 경합을 탐지하기 위해서 레이블링과 경합탐지 프로토콜을 사용하고, 시각화 엔진은 탐지된 경합의 시각화 결과 및 정보의 복잡성을 줄이기 위해서 추상성[16, 17]을 적용한다.

이러한 경합디버깅 엔진들로 원시 프로그램에서 발생하는 경합을 디버깅하기 위해서는 이들 엔진들이 라이브러리 형태로 원시 프로그램에 삽입되어 경합디버깅 환경을 구성하게 된다. 그러나, 이런 경합디버깅 환경을 프로그래머가 직접 생성한다는 것은 비효율적이며, 프로그래머에게 경합디버깅 환경의 생성에 대한 부담을 준다. 특히, 기존의 도구들은 특정 프로그램 모델에 의존적으로 경합디버깅 엔진을 적용하여 경합디버깅 환경을 구성하므로, 프로그램 모델의 변경 시에 경합디버깅 환경이 최적의 경합디버깅 엔진으로 구성되지 못한다는 문제점이 있다.

3. 경합디버깅 환경의 자동생성

본 절에서는 각 프로그램 모델에 따라서 효과성, 효율성, 확장성, 추상성 등을 고려한 최적의 엔진들을 적용하여 경합디버깅 환경을 자동으로 생성하는 기법과 도구를 살펴본다.

3.1 자동생성 기법

자동생성 기법은 수행 중에 경합을 탐지하기 위해 필요한 경합디버깅 환경을 프로그램 모델에 최적화하여 자동으로 생성해 준다. 프로그램 모델은 내포병렬성과 동기화 모델로 분류된다. 내포병렬성의 기준은 내포된 fork/join의 유무이며, 동기화의 기준은 barrier, critical section, ordered, event 등이지만 OpenMP에서는 ordered와 event 동기화를 현재 제공하지 않으므로 동기화 대상에서 제외한다. 일반적으로 프로그램 모델에 따른 경합디버깅 기법은 내포병렬성만을 고려한 기법[6, 8, 10, 12, 14]이나 동기화만을 고려한 기법[4, 6, 9, 10, 14, 15]들이 내포 병렬성과 동기화를 모두 고려한 기법[4, 6, 8, 10, 12, 14]에 비해서 기능 및 성능이 우수하다.

Technique \ Goal	Effectiveness		Scalablility		Efficiency	
	Verify	Locate	High	Medium	Space	Time
Detection Protocol	[4], [10]	[9], [12]		[8], [13] [9], [15]	[4], [10]	[9], [12]
Thread Labeling			[6], [14]	[3]	[4], [6] [14]	[4], [6] [14]

[그림 3.1] 감시기법 요약표

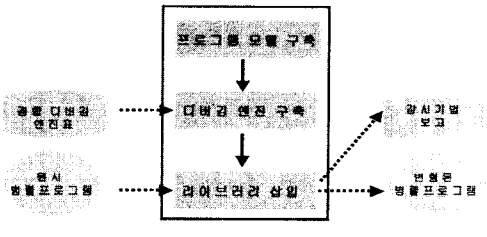
이러한 프로그램 모델에 따른 최적의 경합디버깅 환경을 생성하기 위해서 효과성(effectiveness), 확장성(scalability), 효율성(efficiency), 추상성(abstraction) 등을 고려한다. 첫 번째, 효과성은 적어도 하나의 경합을 탐지하는 것을 보장하는 경합검증 기법(verify)[4, 10]과 최초로 발생하는 경합을 탐지하는 것을 보장하는 최초경합 탐지기법(locate)[9, 12]으로 구분한다. 두 번째, 확장성은 병목현상의 발생 유무에 따라서 병목현상이 없는 기법(high)[6, 14]와 병목현상이 존재하는 기법(medium)[3, 8, 9, 13, 15]으로 구분한다. 세 번째, 효율성은 공간적(space)[4, 6, 10, 14]·시간적(time)[4, 6, 9, 12. 14] 복잡도에 따라서 구분한다. 마지막으로 추상성은 시각화 기법의 유무에 따라서 시각화 기법이 적용되는 기법(abstract)[15, 16]과 적용되지 않는 기법(concrete)으로 구분한다.

그리고, 이러한 고려사항들을 경합디버깅 엔진의 구성요소인 레이블링, 사건선택, 프로토콜, 시각화에 적용하여 최적의 기법들을 선택한다. 따라서, 자동생성 기법은 프로그램 모델에 따른 최적의 기법들을 적용한 경합디버깅 엔진을 자동적으로 선택하여 경합디버깅 환경을 구성해 준다. [그림 3.1]은 디버깅 대상이 되는 프로그램에서 경합디버깅을 위해 변형된 프로그램에 사용된 기법들을 보여주는 감시기법 요약표이다.

3.2 자동생성 도구

제안된 자동생성 기법은 Pentium 서버 시스템에서 경합디버깅 엔진의 라이브러리화를 위해서 Linux RedHat 6.2를 설치하고, 경합디버깅 환경의 수행을 위해서 Omni-1.3 OpenMP 컴파일러를 사용하고 웹 인터페이스를 구현하기 위해서 JSP를 사용하였다. 이런 실험환경에서 자동생성 기법을 적용하여 디버깅 환경을 자동생성해 주는 도구의 구조는 그림 3.2]와 같다. 자동생성 도구는 디버깅 대상이 되는 원시 병렬프로그램의 모델을 선언할 수 있게 하고, 해당하는 경합탐지엔진과 시각화 엔진을 적용하여 디버깅 엔진을 선택할 수 있게 한다. 그리고, 선택된 엔진에 해당하는 경합탐지 라이브러리와 시각화 라이브러리를 삽입하여 경합디버깅을 위한 변형된 병렬프로그램을 생성한다. 이때 프로그램 모델에 의존적으로 적용된 감시기법 요약표가 생성된다.

이러한 자동생성 구조에 의해 구현된 사용자 인터페이스는 플랫폼에 독립적이고, 유지보수 및 개선 등의 부담이 없게 하기 위해서 웹 기반으로 한다. 경합디버깅을 위한 사용자 웹 인터페이스는 Source, Benchmark, Configure, Instrument, Monitor, Visualize, Stop, Information 등의 메뉴로 구성된다. 이 중에서 경합디버깅 환경을 자동으로 구성시켜주는 메뉴는 Source, Configure, Instrument,



[그림 3.2] 자동생성 도구 구조

(A) (B) (C)

[그림 3.3] 자동생성 도구의 사용자 웹 인터페이스

Information 등이다. [그림 3.3-A]의 Source 메뉴는 디버깅 대상이 되는 원시 병렬프로그램의 모델을 선언할 수 있게 하고, 자신의 컴퓨터에 있는 디버깅 대상 프로그램을 디버깅할 수 있는 서버로 업로드할 수 있게 한다. [그림 3.3-B]의 Configure 메뉴는 업로드된 프로그램 모델에서만 적용될 수 있는 최적의 엔진들을 보여주고, 선택할 수 있게 한다. [그림 3.3-C]의 Instrument 메뉴는 선택된 엔진들에 해당하는 경합디버깅 라이브러리를 자동으로 삽입하여 변형된 병렬프로그램으로 변환시킨다.

4. 결론 및 향후과제

본 논문에서는 OpenMP 병렬프로그램에서 각 프로그램 모델에 따라 효과성, 효율성, 확장성 등을 고려한 경합탐지 엔진과 추상성을 고려한 시각화 엔진으로 경합디버깅 환경을 자동으로 생성하는 도구를 제안했다. 제안된 도구는 디버깅 대상이 되는 프로그램의 모델에 최적인 경합디버깅 엔진을 적용하므로 경합탐지의 목적에 부합하는 최적의 성능과 효과적인 시각화를 제공한다. 따라서, 디버깅 환경을 생성하기 위한 부담을 줄여서 효과적인 디버깅을 할 수 있게 한다. 향후과제로는 자동생성 도구를 확장하여, 효과적인 경합디버깅이 가능하도록 사용자에게 보다 편리한 인터페이스를 제공하고자 한다.

참고 문헌

[1] Callahan, D., K. Kennedy, and J. Subhlok, *"Analysis of Event Synchronization in a Parallel Programming Tool,"* 2nd Symp. on Principles and Practice of Parallel Programming, pp. 21-30, ACM, March 1990.

[2] Dagum, L., and R. Menon, *"OpenMP: An Industry-Standard API for Shared - Memory Programming,"* Computational Science & Engineering, 5(1): 46-55, IEEE January/March 1998.

[3] Dinning, A., and E. Schonberg, *"An Empirical Comparison of Monitoring Algorithms for Access Anomaly Detection,"* 2nd Symp. on Principles and Practices of Parallel Programming, pp. 1-10, ACM, March 1990.

[4] Dinning, A., and E. Schonberg, *"Detecting Access Anomalies in Programs with Critical Sections,"* 2nd Wrokshop on Parallel and Distributed Debugging, pp. 85-96, ACM, May 1991.

[5] Emrath, P. A., S. Ghosh, and D. A. Padua, *"Detecting Nondeterminacy in Parallel Programs,"* Software, 9(1): 66-77, IEEE, Jan. 1992.

[6] Jun, Y. and K. Koh, *"On-the-fly Detection of Access Anomalies in Nested Parallel Loops,"* the 3rd ACM/ONR Workshop on Parallel and Distributed Debugging, pp. 107-117, ACM, May 1993.

[7] Jun, Y., and C. E. McDowell, *"On-the-fly Detection of the First Races in Programs with Nested Parallelism,"* 2nd Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), CSREA, Sunnyvale, Calif., Aug. 1996.

[8] Jun, Y., and C. E. McDowell, *"Scalable Monitoring Technique for Detecting Races in Parallel Programs,"* 5-th Int'l Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS2000), pp. 340-347, IEEE, Cancun, Mexico, May 2000.

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

[10] Mellor-Crummey, J., *"On-the-fly Detection of Data Races for Programs with Nested Fork-Join Parallelism,"* Supercomputing '91, pp. 24-33, ACM/IEEE, Nov. 1991.

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

[12] Park, H., and Y. Jun, *"Detecting the First Races in Parallel Programs with Ordered Synchronization,"* Proc. of the 6th Int'l Conf. on Parallel and Distributed Systems (ICPADS'98), pp. 201-208, IEEE Tainan, Taiwan, Dec. 1998.

[13] 김영주, 이승렬, 박미영, 김성대, 박소희, 전용기, "록킹을 가진 병렬프로그램의 경합탐지를 위한 확장적 감시 기법," 병렬처리시스템 학술발표회논문집, 11(2): 44-53, 한국정보과학회, 2000. 9.

[14] 박소희, 박미영, 김병철, 전용기, "병렬 프로그램의 효율적인 수행중 경합탐지를 위한 레이블링 기법," 한국정보과학회 영남지부 학술발표논문집, 9(1): 121-130, 한국정보과학회, 2001. 12.

[15] 이승렬, 김영주, 전용기, "동기화를 가진 공유메모리 병렬프로그램의 최초경합을 위한 효율적인 수행중 탐지 기법," 한국정보과학회 추계학술발표논문집, 26(2): 774-776, 1999. 10.

[16] 이진희, 김영주, 전용기, "OpenMP 프로그램의 경합 디버깅을 위한 추상적 시각화," 병렬처리시스템 학술발표회 논문집, 12(1): 13-21, 한국정보과학회, 2001. 9

[17] 이진희, 김영주, 전용기, "OpenMP 프로그램의 경합 디버깅을 위한 추상적 시각화 정보의 추적선택기법," 한국정보과학회 영남지부 학술발표논문집, 9(1): 112-120, 한국정보과학회, 2001. 12.