스레드 동기화가 없는 OpenMP 디렉티브 프로그램을 위한 최적의 경합검증 도구

하옥균[O1] 강문혜[1] 김영주[2] 전용기[1]
[1]경상대학교 컴퓨터과학부
{jassmin, turtle}@race.gnu.ac.kr, jun@gnu.ac.kr
[2]한국정보통신대학교 IT 공학부
kimyj@icu.ac.kr

An Optimal Tool for Verifying Races in OpenMP Directive Programs With No Interthread Synchronization

Okkyoon Ha[O1] Moonhey Kang[1] Youngjoo Kim[2] Yongkee Jun[1]
[1]Division of Computer Science, Gyeongsang National University
[2]Information and Communications University School of Engineering

OpenMP 디렉티브 프로그램[1, 11]에서의 심각한 오류중의 하나인 경합(race)[10]은 병행적으로 수행되는 스레드들이 하나의 공유변수에 대해 적절한 동기화 없이 적어도 하나 이상의 쓰기 사건으로 접근할 때 나타난다. 이러한 경합은 프로그램 상에 의도하지 않은 비결정적 (non-deterministic)인 수행결과를 초래하므로 디버깅을 위해서 반드시 탐지되어야 한다. 이러한 경합을 프로그램 수행 중에 탐지하는 수행 중 탐지기법(On-the-fly Detection)[2, 9]은 접근사건의 정보를 수집하는 과정에서 불필요한 정보를 삭제하므로 효율적이다. OpenMP 디렉티브 프로그램에서의 경합을 수행 중에 탐지하기 위한 기존 도구[3, 4, 5, 13]는 병행하는 스레드들을 순서적으로 수행한 정보를 이용하여 프로그램 수행 중에 데이터의 의존성 여부를 검사한 후에 경합을 탐지하므로 소요되는 시간과 공간의 비용이 크다. 또한, 이 도구는 내포병렬성이 존재하는 프로그램 모델에서 내포된 스레드를 부모 스레드와 순서적인 스레드로 간주하고, 동기화가 있는 프로그램 모델에서 경합에 참여할 가능성이 있는 접근사건을 무시하는 경우가 존재하므로 경합의 존재를 검증하지 못한다.

이러한 기존 도구의 문제점을 해결하기 위해서 본 연구팀은 프로그램의 수행 중 생성되는 병렬 스레드에 고유한 식별자를 부여하는 레이블링[2, 6, 12] 기법과 접근사건들의 발생 시 마다 병행성 여부를 검사하는 효율적 경합탐지 프로토콜[2] 기법을 OpenMP 디렉티브 프로그램의 모델에 따라서 분류하여 경합을 검증하는 도구[8]를 제안하였다. 이 도구는 스레드 동기화가 있는 프로그램 모델에서는 최적화되어 있지만 스레드 동기화가 없는 프로그램 모델에서는 스레드 레이블 정보에서 부모 스레드의 정보를 탐색하기 위한 시간적 효율성이 낮으므로 최적화되지 못하였다.

본 논문에서는 스레드 동기화가 없는 프로그램 모델에서의 최적화되지 못한 문제를 해결하고 선행연구의 통합도구를 확장하기 위해서 제안하는 도구의 레이블링을 지역자료 구조를 사용하여 병행성 정보를 생성하며 최대 병행 스레드 수에 비례하는 병목현상이 발생하지 않고 시간과 공간적으로 가장 우수한 성능을 보이는 Nest-Region (NR) Labeling[7] 기법을 적용한다. 그리고 탐지를 위한 프로토콜은 접근역사에 가장 최근의 쓰기 접근사건과 읽기 접근사건 중에서 가장 오른쪽, 왼쪽에서 발생한 읽기접근 사건만을 유지함으로써 공간적 효율성이 높은 Mellor-Crummey의 프로토콜[9] 기법을 적용한다. 제안된 도구의 대상 프로그램은 C기반의 OpenMP 디렉티브 프로그램으로 병렬화 디렉티브인 "#pragma omp parallel for"와 동기화 디렉티브인 "#pragma omp critical"만을 고려한다. 합성 프로그램으로 경합의 검증여부와 경합탐지 시에 소요되는 시간을 측정한다. 실험을 위해서 사용되는 시스템은 리눅스 운영체제하의 64bit Intel Xeon 듀얼 CPU 컴퓨터에 대상 프로그램 모델인 OpenMP 디렉티브 프로그램을

위한 Intel C/C++ 컴파일러[7]를 설치하였으며 기존도구의 실험을 위해서 Thread Checker ver. 3.0을 설치하고 제안된 도구의 실험을 위해서는 경합검증을 위한 라이브러리들을 설치한다. 이 라이브러리들은 C 언어로 구현되었다. 그리고 두 도구는 동일한 시스템 환경하에서 실험한다.

스레드 동기화가 없고 비내포병렬성이 존재하는 합성 프로그램 모델을 대상으로 최대병렬성을 2의 지수승으로 증가시키고 스레드 당 접근 사건수를 증가시키면서 경합탐지의 소요시간을 측정한 결과 두 도구 모두 총접근사건수(최대병렬성 * 스레드 당 접근사건수)에 따라 증가하였으나, 제안된 도구는 기존 도구에 비해서 평균 250배 이상 빠르다. 또한 총접근사건 수를 4000으로 고정시키고 최대병렬성을 2의 지수승만큼 증가시키며 작성된 비동기적 비내포병렬성의 합성프로그램으로 경합탐지에 소요된 시간을 측정한 결과에서는 Thread Checker는 최대병렬성이 증가하여도 경합검증을 위한 소요 시간의 변화에 미치는 영향이 거의 없는 반면 제안된 도구는 최대병렬성이 증가함에 따라 오히려 탐지시간이 감소한다. 이상의 효율성 비교 실험에 의해 제안된 도구는 스레드 동기화가 없는 프로그램 모델에 최적화되었음을 알 수 있다.

참고 문헌

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

[2] 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.

[3] Intel Co., Intel Thread Checker Release Notes, 2002.

[4] Intel Co., Getting Started with the Intel Thread Checker and Thread Profiler, 2003.

[5] Intel Co., Threading Methodology: Principle and Practices, Version 2.0, 2003.

[6] I. Nudler, L. Rudolph, "Tools for the Efficient Development of Efficient Parallel Programs," *First Israeli Conference on Computer Systems Engineering*, May 1986.

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

[8] Kim, Y., M. Park, S. Park, and Y. Jun, "A Practical Tool for Detecting Races in OpenMP Programs," *Proc. of 8th Int'l Conf.* on Parallel Computing Technologies (PaCT), Krasnoyarsk, Russia, Lecture Notes in Computer Science, 3606: 321-330, Springer-Verlag, Sept. 2005.

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

[10] 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.

[11] OpenMP Architecture Review Board, *OpenMP Fortran Application Program* Interface, Ver. 2.0, Nov. 2000.

[12] Park, S., M. Park, and Y. Jun, "A Comparison of Scalable Labeling Schemes for Detecting Races in OpenMP Programs," *Proc. of the Int'l Workshop* on OpenMP Applications and Tools (Wompat), Purdue Univ., West Lafayette, Indiana, Lecture Notes in Computer Science, 2104: 68-80, Springer-Verlag, July 2001.

[13] Petersen, P., and S. Shah, "OpenMP Support in the Intel Thread Checker," *WOMPAT* 2003, pp. 1-12, June 2003.