임계구역을 가진 공유메모리 병렬프로그램에서 효율적인 경합 탐지를 위한 사건 선택기법*

김영주° 이승렬 전용기
경상대학교 컴퓨터과학과
{yjkim, tony}@race.gsnu.ac.kr, {jun}@nongae.gsnu.ac.kr

Filtering Accesses for Detecting Races in Parallel Programs with Locking

Young-Joo Kim° Seung-Ryul Lee Yong-Kee Jun
Dept. of Computer Science, Gyeongsang National University

요 약

경합은 공유메모리 병렬프로그램의 비결정적인 수행결과를 초래하므로, 디버깅을 위해서 경합탐지는 중요하다. 임계구역을 가진 병렬프로그램을 위한 수행중 경합 탐지 기법은 공유 자료구조를 사용하므로, 매 접근 사건 시에 병목현상을 유발한다. 본 연구에서는 동기화가 있는 병렬프로그램에서 매 반복을 수행할 때마다 공유 자료구조의 접근 횟수를 기껏해야 임계구역의 수에 비례하도록 매 접근사건을 검사한다. 그러므로 이 기법은 수행중 경합탐지의 확장성과 효율성을 제공한다.

1. 서론

공유 메모리를 기반으로 하는 병렬프로그램에서는 스레드들 간의 상호영향으로 인한 예기치 않는 수행결과를 초래할 수 있다. 이는 적절한 동기화 없이 공유변수를 적어도 한번은 수정해야 하는 병렬 스레드가 존재할 때 발생하는 오류를 경합(race)[4]라고 한다. 경합은 공유 메모리 병렬프로그램의 비결정적인 수행결과를 초래하므로, 디버깅을 위해서 경합탐지는 중요하다. 특히 가장 먼저 발생하는 경합을 최초경합(first races)[3]이라고 하며, 이를 탐지하는 것은 매우 중요하다.

이러한 경합을 탐지하는 기존의 기법들은 정적분석기법(static analyis)[5], 사후추적 분석기법(post-mortem analysis)[5], 수행중 분석 기법(on-the-fly analysis)[5] 등의 3종류가 있다. 본 논문에서는 이 세 가지 기법들 중에서 가장 현실적인 기법으로 알려진 수행중 분석 기법을 사용한다.

기존의 수행중 탐지 기법은 매 접근사건을 검사하여 접근역사 내에 유지되는 특정 공유변수에 대한 다른 접근사건들과 비교를 통해 경합을 탐지한다. 이 기법은 공유 자료구조의 접근 시에 모든 공유변수들이 순서적으로 접근해야만 하는 병목현상으로 인해 성능저하를 초래한다. 이러한 문제점을 해결하기 위한 여러 가지 방안이 제시되어 왔지만, 임계구역을 가지는 프로그램 모델에서는 여전히 문제가 된다.

본 논문에서는 탐지 기법 자체를 개선하여 임계구역을 가진 프로그램의 수행중 경합탐지 효율성을 높인다. 개선된 탐지 기법의 효율성을 평가하기 위해서, OpenMP 커널 프로그램을 이용하여 개선된 기법을 구현하고 그 효과를 실험하였다. 2장에서는 병렬프로그램의 수행에 관계되는 사항을 설명하고, 기존의 수행중 경합탐지 기법이 가진 문제점을 소개한다. 3장에서는 개선된 탐지기법과 알고리즘을 설명하고, 4장에서는 본 논문에서 제안하는 기법의 평가를 위한 실험과 그 결과를 분석하고, 마지막으로 5장에서는 결론을 내린다.

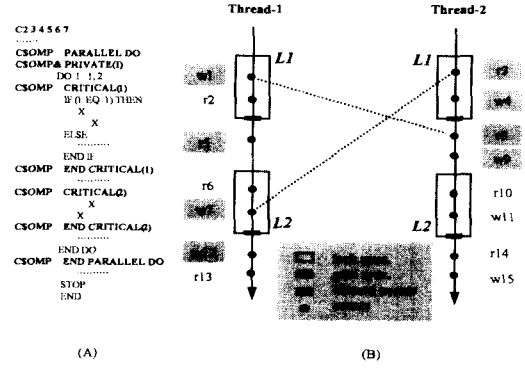
2. 연구배경

본 절에서 대상으로 하는 병렬프로그램의 수행모델과 그 프로그램 모델의 수행에서 나타나는 경합을 보이고, 이를 탐지하기 위한 기존의 기법과 문제점에 대해 기술한다.

2.1 병렬프로그램의 수행

본 논문에서는 공유 메모리를 사용하는 병렬프로그래밍 모델로서 산업 표준인 OpenMP[6] Fortran 프로그램을 대상으로 하며, 내포 병렬성이 없이 임계구역을 가질 수 있다. 임계구역(critical section)[2]은 특정 명령문들의 수행 시에 상호배제를 제공하므로, 한 순간에 한 스레드만이 수행할 수 있는 동기화된 구역이다. 비임계구역(non-critical section)은 임계구역에 속하지 않은 스레드 수행 부분을 뜻한다. 동기화 블록(synchronized block)은 동일 스레드 내에서 임계구역의 해제를 수행하는 연속된 두 신호명령(unlock)들간의 명령어 블록이다.

(그림1)의 (A)는 OpenMP 병렬프로그램의 예이다. 2개의 스레드가 병행적으로 수행하기 위해서 삽입된 OpenMP 디렉티브[6]는 C$OMP PARALLEL DO이다. 그리고 임계구역을 표시하는 디렉티브는 C$OMP CRITICAL과 C$OMP END CRITICAL이다. (그림1)의 (B)는 OpenMP로 작성된 프로그램을 도식화했



[그림 1] 병렬 프로그램의 수행모델

* 정보통신부에서 지원하는 대학기초연구지원사업으로 수행

다. 스레드에서 화살표의 방향은 사건들의 수행을 나타내고, r(w)은 읽기(쓰기) 접근사건을 나타내며, 각 번호들은 사건의 수행 순서를 나타낸다. 임계구역의 접근은 대기/신호(lock/unlock) 명령들에 의해 결정된다. 임의 변수에 대한 접근사건 w1과 r2는 임계구역1(L1)에 속하고, r3와 w4는 임계구역2(L2)에 포함되어 있다. 같은 임계구역끼리는 사건 동기화가 이루어지지만, 다른 임계구역끼리는 동기화가 이루어지지 않는다. 그리고 Thread-1에서의 w1와 r2는 첫 번째 동기화 블록에 속하고, r5, r6, w7은 두 번째 동기화 블록에 포함되어 있다.

2.2 수행중 탐지기법

수행중 탐지 기법은 병렬프로그램 수행 중에 접근사건의 발생 시마다 공유 자료구조인 접근역사를 유지하며 논리적인 병행성을 체크하여 경합을 보고한다. 레이블링(labeling)은 수행중인 각 스레드에 병행성을 식별 가능한 고유한 정보를 할당하는 기법으로서, 접근역사에 기록될 수 있다. 매 접근사건의 수행 시에 이전에 발생한 접근사건들과의 병행성을 검사하여 접근 역사를 유지하는 기법을 탐지 프로토콜(protocol)이라 한다. (그림1)에서 w1과 r8이 경합으로 보고되기 위해서는, 접근역사에 w1에서 w7까지가 저장된 상태에서 r8이 발생될 때 논리적 병행성을 검사하여 경합을 보고한다. 이 때, w1과 r8 간의 경합을 w1-r8으로 표시한다.

프로토콜은 매 접근사건의 발생 시마다 접근역사에 순서적으로 접근해야 하므로 병목현상을 초래한다. 본 논문에서는 매 접근 사건이 최초경합의 구성 대상인 경우에만 접근 역사를 접근하는 프로토콜을 제안한다. 이는 기존의 프로토콜로서 최초 경합을 구성하는 사건을 검사하면, 적어도 하나의 경합탐지는 보장되기 때문이다. 그 결과, 매 스레드를 수행할 때마다 접근 역사의 접근횟수를 최대 임계구역의 수에 비례하고, 매 접근사건의 발생 수와는 무관하게 된다. 예로서, (그림1)의 Thread-2에서 기존의 수행중 탐지기법으로 수행을 했다면 최악의 경우 접근 역사에 8번을 접근해야 하지만, 본 연구 기법으로는 4번의 접근으로 탐지된다.

3. 사건 선택기법

본 절에서는 병목현상을 개선하는 수행중 탐지를 위해서, 매 접근 사건을 검사하여 각 동기화 블록마다 최대 2개의 접근사건만 접근 역사에 접근할 수 있도록 하는 선택사건 알고리즘을 소개한다. 먼저 이를 위해 선택되는 사건인 선택사건(filtered event)를 소개하고, 선택사건 알고리즘을 설명한다.

3.1 선택사건

선택사건이란 각 스레드에 대해서 최초경합에 참여할 수 있는 접근사건들을 말한다. 선택사건으로는 읽기 선택사건(read filtered event)과 쓰기 선택사건(write filtered event) 등의 2가지 종류가 있다

[정의1] 한 동기화 블록 안에서 어떤 읽기 (쓰기) 사건 a_i에 선행하는 임의의 다른 읽기 (쓰기) 사건이 없다면 a_i를 읽기 (쓰기) 선택사건이라 한다.

[정의2] 임의 스레드에서 처음으로 비임계구역에서 발생한 쓰기 선택사건을 선택중단(finally-filtered) 사건이라 한다.

(그림1)의 (B)는 임계구역을 가진 병렬프로그램 수행에서 나타난 총 15개의 접근사건들 중에 8개의 선택사건들과 최초경합을 보이고 있다. Thread-2에서 r3은 첫 번째 동기화 블록에서 [정의1]의 조건을 만족하므로 읽기 선택사건이 된다. Thread-1에서 w12는 세 번째 동기화 블록에서 [정의1]의 조건을 만족함으로 쓰기 선택사건인 동시에 [정의2]의 조건도 만족함으로 선택중단 사건이기도 하다. 따라서 Thread-1에서는 w1, r5, w7, w12 등이 선택사건들이고, 그 중에서 w12는 선택중단 사건이 된다. Thread-2에서는 r3, w4, r8, w9 등이 선택사건들이고, 그 중에서 w9가 선택중단 사건이 된다.

기존의 경합탐지 기법[3]을 사용한다면 w1, r3, w4 등 세 가지의 사건만 선택되고 동일한 임계구역이므로 경합을 보고하지 않는다. 그러나 실제로 최초경합은 w1-r8, r3-w7이다. 이처럼 임계구역을 가진 병렬프로그램 모델에서는 기존의 경합탐지 방법[3]만으로는 경합탐지에 오류를 범할 수 있기 때문에, 사건을 선택하는 새로운 기법을 사용하여 이러한 문제점을 해결한다. 이 기법은 임계구역을 가진 병렬프로그램에서 동기화 블록마다 기껏해야 두 개의 선택사건만 선택하더라도 경합 탐지를 보장할 뿐 아니라, 병목현상의 문제점도 감소시킬 수 있다

3.2 사건 선택 알고리즘

(알고리즘1)과 (알고리즘2)은 동기화 명령이 존재하는 병렬프로그램에서 읽기 혹은 쓰기 접근사건이 발생할 때, 선택사건인지를 검사하는 알고리즘이다. (알고리즘1)과 (알고리즘2)에서 post_count는 각 스레드마다 하나씩 존재하며, 그 값이 0으로 초기화되어, 동기화 블록이 시작될 때마다 1씩 증가되는 private 변수이다. (그림 2)는 임의 공유변수를 위한 사건 선택을 위하여 각 스레드마다 존재하는 4개 변수의 private 자료구조로서, 스레드 접근역사(thread access history)라 한다. 그림에서 r, w는 사건 선택 여부를 검사하는 불리언 변수이고, p_n는 현재 수행 이전까지의 동기화 블록 개수를 저장하는 정수형 변수이고, f_d는 사건 선택의 중단 여부를 저장하는 불리언 변수이다.

(그림1)의 수행에서 나타나는 접근사건들을 (알고리즘1)과 (알고리즘2)에 있는 CheckRead()와 CheckWrite() 함수에 적용시키면, 선택사건인 경우에만 탐지 프로토콜[2]을 호출하여 경합을 탐

```
procedure CheckRead(X, post count, c locks, c tag)
    if t ah(X, post no) ≠ post count and ¬ t ah(X, filter done) then
        reset t ah(X);
        t ah(X, post no) := post count;
    endif
    if c locks = ∅ then
        if ¬ t ah(X, r) and ¬ t ah(X, w) then
            CheckReadFiltered(X, c locks, c tag);
            t ah(X, r) := true;
        endif
    else
        if ¬ t ah(X, r) and ¬ t ah(X, w) then
            CheckCSReadFiltered (X, c locks, c tag);
            t ah(X, r) := true;
        endif
    endif
end procedure
```

[알고리즘1]

```
procedure CheckWrite (X, post count, c locks, c tag)
    if t ah(X, post no) ≠ post count and ¬ t ah(X, filter done) then
        reset t ah(X),
        t ah(X, post no) := post count;
    end if
    if c locks = ∅ then
        if ¬ t ah(X, r) and ¬ t ah(X, w) then
            CheckWriteFiltered (X, c locks, c tag);
        elseif t ah(X, r) and ¬ t ah(X, w) then
            CheckR-writeFiltered (X, c locks, c tag);
        endif
        t ah(X, filter done) := true;
        t ah(X) := true;
    else
        if ¬ t ah(X, r) and ¬ t ah(X, w) then
            CheckCSWriteFiltered (X, c locks, c tag);
        elseif t ah(X, r) and ¬ t ah(X, w) then
            CheckCSR-writeFiltered (X, c locks, c tag);
        endif
        t ah(X) := true;
    endif
end procedure
```

[알고리즘2]

r	w	p_n	f_d

[그림 2] 스레드 접근자료구조

t_ah(thread-1)

r	w	p_n	f_d
	w1	0	F
r5		1	F
	w7	1	F
	w12	2	T

t_ah(thread-2)

r	w	p_n	f_d
r3		0	F
	w4	0	F
r8		1	F
	w9	1	T

[그림3] 스레드 자료구조 변화상태

지하고 난 후, 스레드 접근역사에 그 선택사건을 저장한다. 따라서 선택사건 발생 때만 접근역사에 접근함으로써, 매 접근사건 발생 시에 생기는 병목현상을 선택사건 발생 시에만 나타나도록 줄이게 된다.

알고리즘에서 매 동기화 블록마다 스레드 접근역사는 초기화되기 때문에, 그 후에는 다시 사건 선택을 시작한다. 그러므로 스레드 접근역사는 최대 2개의 선택사건에 대한 정보만 저장된다. w1이 선택되어 저장된 상태라고 가정한 후 r5 접근사건이 발생하면 스레드 접근역사 내에 저장된 p_n과 post_count을 비교하며, 다른 동기화 블록임을 판단하게 되어 w1을 삭제하고 r5가 저장된다. 실제로 스레드 접근역사에 저장되는 r, w의 값은 불리언 값이 저장된다. (그림3)은 (그림1)의 수행 예에서 알고리즘이 수행될 때, 그 결과를 보여주는 스레드 접근역사의 상태 변화를 보인다.

사건 선택 알고리즘은 (그림1)의 경우에 Thread-1의 w12 선택중단 사건과 Thread-2의 w9 선택중단 사건 이후에 발생한 모든 접근사건들에 대해서는 접근역사에 접근을 허용하지 않으므로, 그러한 접근사건들이 많을 수록 병목현상을 줄이는 효과가 크다. 선택된 사건만으로도 경합이 존재한다면 적어도 하나의 경합은 탐지할 수 있기 때문에, 그 이후에 발생되는 접근사건은 무시해도 무관하다. 그러므로 선택중단 사건은 그 스레드에서 마지막으로 선택될 사건을 의미한다.

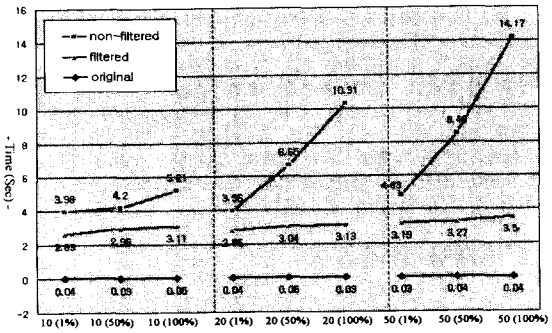
요약하면 사건선택 알고리즘의 특징은 두 가지이다.

[특징1] 기존의 경합탐지 프로토콜을 적용할 수 있다.
[특징2] 각 스레드 내의 선택사건 검사는 다른 스레드에 영향을 받지 않고 독립적으로 수행할 수 있다.

4. 실험 환경 및 분석

본 기법의 효율성을 평가하기 위해서 OpenMP Fortran으로 실험했다. 실험을 위해 레이블링 기법으로는 BD 레이블링[1]을 사용했고, 프로토콜 기법은 Lock Cover 프로토콜[2]을 사용했다. BD 레이블링은 동기화를 가진 병렬프로그램에서 스레드마다 고유번호를 붙이는 레이블링 기법이고, Lock Cover 프로토콜 기법은 임계구역을 가진 병렬프로그램에서 경합을 탐지하는 프로토콜이다. 그리고 선택사건 알고리즘은 C언어로 구현하였다. BD 레이블링과 선택사건 알고리즘은 라이브러리화하였고, 실험 대상인 OpenMP Fortran 프로그램에 라이브러리 호출코드를 추가하였다. 대상 프로그램은 커널 프로그램으로 작성하여 평가하였다.

커널 프로그램은 9개로서, (1) 최대병렬성을 10으로 하고, (2) 한 스레드 당 블록 수는 201 개, (3) 공유변수는 1 개, (4) 블록 당 접근사건 수는 10, 20, 50 등의 3가지 경우로 하고, (5) 선택중단 사건의 위치는 전체 접근사건들 중의 1%, 50%, 100% 등에 위치하는 것으로 하였다. 그러므로 각 프로그램의 총 접근사건 수는 각각 20100, 40200, 100500 개이다. 이러한 OpenMP Fortran 커널 프로그램들을 이용하여 3가지 수행시간을 측정하였다. 측정한 시간은 (1) 원래 커널 프로그램의 수행시간, (2) 선택사건 기법을 적용하지 않고 경합을 탐지한 시간, (3) 사건선택 기법을 적용하여 경합을 탐지한 시간 등이다. 실험 결과인 (그림5)의 X축은 블록 당 접근사건 수를 의미하고 괄호안의 값은 선택중단 사건의 위치를 의미한다. Y축은 시간(초)를 의미한다. (그림4)에서 선택사건 개수는 표시되지 않았지만, 총 접근사건 수와는 무관하며 선택중단 사건의 위치에



[그림 4] 사건선택 수행 결과

의존적이다. 예를 들어, 그림에서 50(1%) 커널 프로그램 경우의 총 접근사건 수는 100500개이지만, 경합으로 선택된 사건 수는 불과 40개이다. 그러나 50(100%) 경우의 총 접근사건 수는 동일하게 100500개이지만, 선택된 사건수는 2020 개나 된다. 그러므로 선택사건 기법의 효율성은 선택중단 사건의 위치 값이 작을수록 극대화된다. 그리고 블록 당 접근사건 수가 많아지고, 선택중단 사건의 위치 값이 클수록 그 효율성은 낮아진다. 마지막으로, 본 탐지 기법은 동기화 명령의 횟수에 의존적이지만, 선택사건 알고리즘을 사용하지 하지 않은 경우와 비교해 볼 때 (그림4)에서 보인 바와 같은 높은 효율성을 제공한다.

5. 결 론

수행중 경합탐지 기법은 모든 접근사건 발생 시에 병목현상이 유발되기 때문에, 모든 접근사건들이 순서적으로 검사되어야 하는 문제점이 있다. 본 연구에서는 OpenMP 병렬프로그램에서 적용되는 경합탐지 방법을 제시하였고, 병목현상을 유발하는 접근사건 중에서 경합에 필요한 사건들만 선택함으로써 효율성을 향상시키는 사건선택 알고리즘과 그 실험적 결과를 보였다. 본 기법은 내포되지 않는 임계구역 모델에 제한된 것이므로, 향후 연구과제로는 내포 가능한 임계구역 모델을 위해서 본 기법을 확장할 필요가 있다.

참 고 문 헌

[1] Audenaert, K., "Maintaining Concurrency Information for On-the-fly Data Race Detection," Parallel Computing 97, pp. 1-8, North-Holland, Sept. 1997.

[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] Kim, J., and Y. Jun, "Scalable On-the-fly Detection of the First Races in Parallel Programs," 12nd Int'l. Conf. on Supercomputing (ICS'98), pp. 345-352, ACM, Melbourne, Australia, July 1998.

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

[5] 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.

[6] OpenMP: An Industry-Standard API for Shared-Memory Programming in IEEE Computational Science & Engineering, Vol. 5, No. 1, January/March 1998.