동기화를 가진 공유메모리 병렬프로그램의 최초경합을 위한 효율적인 수행중 탐지 기법*

[0]이 승 렬, 김 영 주, 전 용 기
경상대학교 컴퓨터과학과

(Scalable On-the-fly Detection of the First Races in Parallel Programs with Synchronization)*

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

요 약

공유메모리 병렬프로그램에서의 경합은 프로그램 수행에서 원하지 않는 비결정성을 야기할 수 있기 때문에 반드시 탐지되어져야 한다. 기존의 탐지 기법들은 경합을 탐지하기 위해서 공유 자료구조를 사용하므로 심각한 병목현상을 일으킨다. 본 논문에서는 동기화가 있는 프로그램에서 병목현상을 줄임으로써 탐지의 효율성을 높임과 동시에, 최초로 발생한 경합을 탐지하기 위해서 감시대상이 되는 접근사건들의 수를 감소시키는 기법을 제시한다. 이러한 목적을 위해서 사용되는 사건선택 알고리즘과 실제적인 실험결과를 통해 본 기법의 효율성을 보인다.

1. 서 론

일반적으로 병렬 프로그램 수행에서 스레드들이 공유메모리를 이용하여 상호작용하는 경우에 프로그램 개발자가 의도하지 않았던 비결정적 수행 결과가 초래될 수 있다. 이는 적절한 동기화 없이 공유변수를 적어도 한번은 수정하는 병렬 스레드들이 존재할 때 발생하며, 이와 같은 형태의 오류를 경합(race)이라고 한다.

최초경합(first race)은 병렬프로그램에서 발생할 수 있는 여러 경합 중 가장 먼저 발생하는 경합을 말하는데, 이 최초경합을 탐지하는 것은 매우 중요하다. 그 이유는 최초경합을 해결하게 되면 여기에 영향을 받는 다른 경합들은 사라질 수 있기 때문이다.

수행중 최초경합 탐지 기법은 매 접근사건을 검사하여 접근역사(access history)내에 유지되는 특정 공유변수에 대한 다른 접근사건들과의 비교를 통해 최초경합을 탐지한다. 이 기법은 최초경합을 탐지를 위해 공유 자료구조에 병렬 접근하게 됨으로써 병목현상이 발생하여 성능 저하를 유발시킬 수 있다.

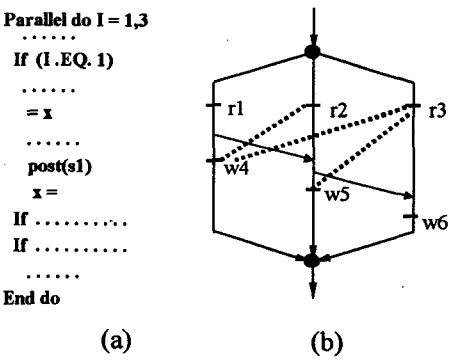
본 논문에서는 이러한 병목현상을 줄이기 위해서 매 접근사건이 수행될 때마다 공유 자료구조인 접근역사를 참조하는 것이 아니라, 최초경합에 참여될 수 있는 후보사건이 수행하는 경우만 접근시킴으로써 접근역사에 대한 병목현상을 줄인다.

2. 연구배경

본 절에서 대상으로 하는 병렬프로그램과 그 프로그램의 수행에서 나타나는 경합을 보이고, 기존의 최초경합 탐지 기법에 대해 논한다.

2.1 병렬프로그램의 수행

본 논문에서의 수행모델은 내포 병렬성 없이 POST/ WAIT 동기화 명령이 존재하는 것으로 한다. POST / WAIT는 사건 동기화를 나타내며, WAIT는 사건 변수 값이 POST를 통해 posted 상태가 될 때까지 해당 스레드의 실행을 지연하게 된다. 반면에 POST는 수행후 즉시 다음 실행을 계속한다. (그림 1)은 대상 프로그램의 예이다.



(그림 1) 동기화가 있는 단일내포 병렬프로그램

* 정보 통신부에서 지원하는 대학기초 연구 지원 사업으로 수행

(그림 1) 의 (a)에서 Parallel do 는 병렬 스레드들의 생성을 나타내고, End do는 병렬 스레드들의 합류를 의미한다. 병렬 프로그램의 수행은 방향성이 있는 비순환 그래프인 POEG (Partial Order Exceution Graph)[2]으로 설명할 수 있다. (그림 1)의 (b)는 (a)의 병렬 프로그램의 수행을 POEG으로 나타낸 것이다. (그림 1)의 (b)에서, 정점은 병렬 스레드의 생성과 합류를 나타내며, 정점들 사이의 간선은 임의의 명령열로 블록(block) 또는 동기화 명령을 나타낸다. POEG은 병렬 프로그램의 수행시 순서 관계를 알 수 있다.

(그림 1)에서 r2-w4, r3-w4, r3-w5 세가지의 경합이 존재한다. 그렇지만 이들 경합 중에서 최초경합은 r2-w4, r3-w4 이고, r3-w5는 r2-w4 경합에 영향을 받기 때문에 유효하지 않는 경합이다.

2.2 관련연구

Race Fronitor 기법[1]은 최초경합 탐지보다는 경합 재생산(reproducing)을 위한 기법으로, 프로그래머로 하여금 중단점(breakpoint)설정이나 여러번 반복 수행을 통한 경합 탐지 기능을 추가로 요구하기 때문에 그 비용이 매우 높아진다는 단점이 있다. 병렬 프로그램에서 효율적인 최초경합 탐지 기법[3]은 최초경합을 탐지하는데 있어서 효율적인 면을 가지고 있지만 동기화가 없는 프로그램만 대상이 된다는 제약을 가지고 있다. 순서적 동기화를 가진 공유메모리 병렬 프로그램에서의 최초경합 탐지 기법[5]은 다양한 모델에서 수행을 한다는 장점은 있지만 순서적 동기화, 즉 일반적인 동기화에서 있어서 수행되지 못한다는 제약과 접근역사에 대한 병목현상으로 성능저하가 발생한다는 단점이 있다.

본 논문에서는 [3]에서 제시하고 있는 효율적인 최초경합 탐지 기법을 확장하여 일반적인 동기화가 있는 프로그램에서도 병목현상을 감소시키는 사건선택 기법을 보이고자 한다.

3. 사건선택 기법

최초경합을 탐지하는 기존의 수행중 탐지 기법은 최악의 경우에 임의 스레드에서 발생하는 접근사건들의 최대 횟수 만큼의 병목현상을 보인다. 따라서 이러한 병목현상을 유발하는 접근사건들의 수를 최악의 경우 동기화 명령간의 블록마다 두개로 줄일 수 있는 사건선택 알고리즘을 제시하고자 한다.

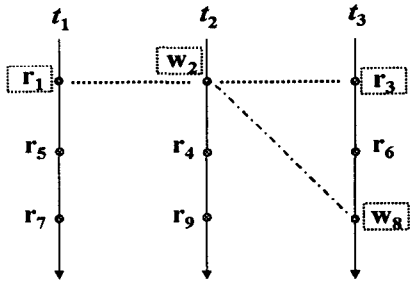
3.1 후보사건

후보사건이란 최초경합에 참여할 수 있는 접근사건들을 말한다. 후보사건으로는 읽기 후보사건, 쓰기 후보사건 그리고 읽은후쓰기 후보사건 등의 3가지 종류가 있다.

1. 어떤 읽기(쓰기) 사건 a_i에 선행하는 임의의 다른 읽기(쓰기) 사건이 없다면, a_i를 '읽기(쓰기) 후보사건(read(write) candidate)'이라 한다.

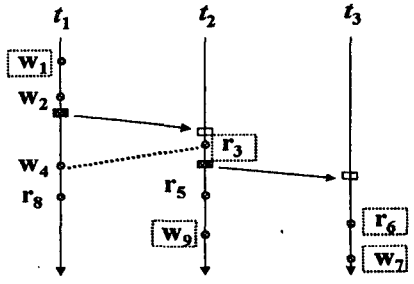
2. 어떤 쓰기 사건 w_i에 선행하는 임의의 다른 쓰기 사건이 없고, w_i에 선행하는 읽기 후보사건이 존재한다면, w_i를 '읽은후쓰기 후보사건(r-write candidate)'이라 한다.

3. 어떤 읽기(쓰기) 후보사건은 스스로 유효(effective)하다. 어떤 읽은후쓰기 후보사건 w_i는 w_i에 병행하는 다른 쓰기 후보사건이 없을 경우 유효하다.



(그림 2) 비동기화에서의 최초경합들

(그림 2)는 후보사건들과 최초경합들을 보이고 있다. 여기에서 r1, w2, r3, w8은 최초경합 후보사건이지만 w2-w8은 유효하지 않는 경합이다. 왜냐하면 최초경합 후보사건들중 w8은 유효하지 않는 읽은후쓰기 후보사건이기 때문이다.



(그림 3) 동기화에서 최초경합

(그림 3)의 병렬 프로그램내에 여러 형태의 동기화 명령들이 존재하면 위에서 정의된 후보사건만으로는 (그림 3)과 같이 최초경합을 탐지할 수 없다. 왜냐하면 동기화 명령에 의해 후보사건 선택이 정확하게 탐지 못함으로써 r3-w4의 최초경합을 탐지할 수가 없기 때문이다. 따라서 위의 후보사건 정의에 확장된 개념이 요구된다.

3.2 사건선택 알고리즘

본 절에서는 동기화를 가지고 있는 프로그램에서 효율적인 최초경합 탐지를 위한 사건선택 알고리즘을 제시한다.

[알고리즘-1] 최초경합 탐지를 위한 사건선택은 스레드간들에 동기화가 존재하더라도 사건선택의 결정은 다른 스레드와 무관하다.

[알고리즘-2] POST 명령을 수행한 후에 이전에 선택된 사건을 제거하고 다시 접근사건들을 판별하여 사건을 선택한다.

위와 같은 방법으로 선택된 사건들은 최초경합 사건들을 포함하고 있다. 왜냐하면 잠재적으로 동시에 수행될 수 있는 스레드의 동기화 블록별로 사건을 추출하기 때문에 [3]에서 제시한 정리를 이용할 수 있다.

다음의 세가지 알고리즘은 동기화 명령이 존재하는 병렬 프로그램에서 사건을 선택하는 것으로써 각 스레드에 접근사건이 발생할 때마다 수행하여 사건들을 결정한다.

t_ah	r	w	rw	postflag

(그림 4) 스레드 접근 자료구조

```
checkread(x, current)
 if t_ah(x, postflag) ≠ posted_count then
  reset t_ah(x, r, w, rw, postflag)
  t_ah(x, postflag) = posted_count
 endif
 if t_ah(x, r) = Ø ∧ t_ah(x, w) = Ø then
  add current to t_ah(x, r)
  checkread_1st(x, current) or checkread_2nd(x, current)
 endif
end checkread
```

(알고리즘 1) Checking a read access

```
checkwrite(x, current)
 if t_ah(x,postflag) ≠ posted_count then
  reset t_ah(x, r, w, rw, postflag)
  t_ah(x, postflag) = posted_count
 endif
 if t_ah(x, r) = Ø ∧ t_ah(x, w) = Ø then
  add currentto t_ah(x, w)
  checkwrite_1st(x,current) or checkwrite_2nd(x,current)
return
 endif
 if t_ah(x, r) ≠ Ø ∧ t_ah(x, rw) = Ø then
  add currentto t_ah(x, rw)
  checkwrite_1st(x,current) or checkwrite_2nd(x,current)
 endif
end checkwrite
```

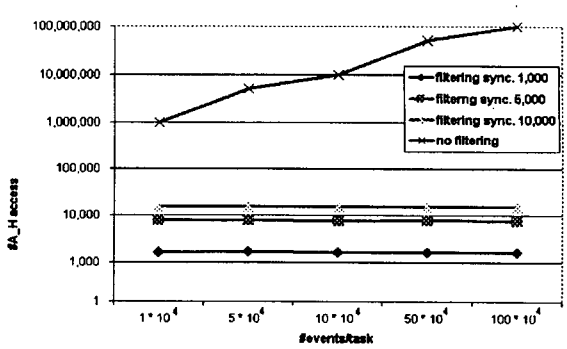
(알고리즘 2) Checking a write access

(알고리즘 1)과 (알고리즘 2)에서 posted_count는 생성된 스레드 수만큼 존재하며, POST 수행시 1씩 증가한다.

checkread_1st, checkread_2nd는 [5]의 최초경합 탐지 프로토콜로서 1차 수행시 checkread_1st에서 선택된 부분경합 사건들을 이용하여 2차 수행시 최초경합 탐지 보고를 위한 경합 사건들을 완성한다.

4. 실험 및 결과 분석

(그림 5)의 실험은 최대 병렬성을 100으로 했을 경우에 각 스레드마다 접근사건의 횟수 및 동기화 명령의 횟수를 변화시켰다. 접근사건 유형들은 무작위로 발생시킴으로써 보다 객관적인 수행 모델로 공유 메모리의 접근횟수를 기존의 탐지 기법과 비교하여 그 효율성을 보이고 있다. (그림 5)의 X축은 각 스레드당 접근사건의 수를 의미하고, Y축은 공유 메모리의 접근 횟수를 나타낸다. filtering sync. 1,000의 'filtering'은 본 논문에서 제시하는 사건선택 알고리즘을 적용한 것을 의미하고 'sync. 1,000'은 스레드들에 1,000개의 동기화 명령이 수행된다는 것을 나타낸다.



(그림 5) 공유메모리 접근횟수 실험결과

위의 실험결과로 본 논문에서 제시하고 있는 최초경합 탐지 기법은 스레드에 접근 횟수가 많은수록 또 병렬성이 클수록 기존의 기법에 비해 효율성을 보이고 있다. 이 탐지 기법은 동기화 명령의 횟수에 의존적이지만 기존의 탐지 기법과 비교해 볼 때 공유메모리 접근횟수가 상대적으로 적다.

5. 결론 및 향후 과제

기존의 최초경합 탐지 기법들은 모든 접근사건들이 순서적으로 처리되어야하는 심각한 병목현상의 문제를 가지고 있다. 본 연구에서는 이러한 병목현상을 유발하는 접근사건중에서 최초경합 탐지에 필요한 사건들만 선택함으로써 효율성을 향상시키는 사건선택 알고리즘과 그 실험적 결과를 보였다.

향후 연구는 적용 대상 프로그램을 확장하여 다중 내포 병렬 프로그램에서 수행할 수 있는 기법의 개발과 최초경합 탐지 기법 수행 환경을 지원할 수 있게 해야 한다.

참 고 문 헌

[1] Choi, J., and S. L. Min, "Race Frontier: Reproducing Data Races in Parallel-Program Debugging," 3rd Symp. on Principles and Practice of Parallel Programming (SPPP), pp 145-154, ACM, April 1991.

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

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

[5] 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, Dec. 1998.