OpenMP 프로그램의 디버깅을 위한 효과적 경합 시각화

김금희,* 김영주,** 전용기***
경상대학교 컴퓨터과학과
{picot, yjkim, jun}@race.gsnu.ac.kr

Effective Race Visualization for Debugging OpenMP Programs

Kum-Hee Kim, Young-Joo Kim, and Yong-Kee Jun
Dep't. of Software Technology, Gyeongsang National University

요 약

OpenMP 프로그램에서 수행되는 스레드들간에 적절한 동기화 없이 적어도 하나의 쓰기사건으로 동일한 공유변수에 접근하는 경우에 발생되는 오류인 경합은 비결정적인 수행결과를 초래하므로 디버깅을 위해서 반드시 탐지되어야 한다. 이러한 경합탐지를 위한 기존의 디버깅 도구는 프로그램의 복잡한 수행구조 및 디버깅 정보를 시각화하기 위한 공간이 제한적이므로 효과적인 시각화를 제공하지 못한다. 본 논문에서는 경합 시각화를 위해서 3차원적 시각화와 스레드 및 이벤트 등의 추상화 기능으로 공간적 제약성을 해결하는 도구를 제안한다. 제안된 도구는 추상적 시각화 정보를 제공하므로 프로그램의 이해가 용이하고 효과적인 경합디버깅 환경을 제공한다.

1. 서 론

OpenMP[2]는 공유메모리를 사용하는 병렬프로그래밍을 위한 산업표준 API다. 이러한 프로그램에서 수행되는 스레드들간에 적절한 동기화 없이 적어도 하나의 쓰기사건으로 동일한 공유변수에 접근하는 경우에 발생되는 오류인 경합은 비결정적인 수행결과를 초래하므로 디버깅을 위해서 반드시 탐지되어야 한다. 특히 다른 경합에 영향을 받지 않고 발생하는 경합들 중에서 처음으로 발생하는 경합인 최초경합[5, 6]의 탐지는 중요하다. 왜냐하면 최초경합의 제거는 이로부터 영향을 받아서 발생한 다른 경합들을 제거하거나 새로이 나타나게 할 수 있기 때문이다.

병렬프로그램의 경합탐지를 위한 기존의 디버깅 도구[4, 7]는 관련된 경합정보를 생성하여 보여주고, 사용자는 이 정보를 사용하여 프로그램을 디버깅하게 된다. 그러나 프로그램의 병렬성 증가로 인해 수행이 복잡해짐에 따라 생성된 경합정보와 프로그램 수행구조에 대한 이해가 어려워지게 된다. 일반적으로 복잡성 문제는 그래픽 기반의 시각화[1, 4, 6, 7, 9, 10]로써 해결하고 있으나, 시각화 정보와 결과의 복잡성 및 공간적 제약성의 문제가 따른다. 이는 시각화를 위한 정보의 수집대상을 한정시키거나, 생성된 정보 중 필요한 정보만을 선택적으로 시각화하고, 특정 대상을 임의의 심볼로 대체하는 추상화 기법[6, 9, 10]을 적용함으로써 해결한다. 그러나 기존의 디버깅 도구는 시각화를 위한 공간이 제한적이므로 이벤트 및 스레드의 추상적 시각화를 효과적으로 제공하지 못한다.

본 논문에서는 경합 시각화를 위해서 3차원적 시각화와 스레드 및 이벤트 등의 추상화 기능으로 공간적 제약성을 해결하는 도구를 제안한다. 3차원적 시각화는 xyz축 공간의 활용이 가능한 정육면체를 사용하여 해결하고, 스레드와 이벤트 등의 추상화는 superthread와 이벤트 심볼을 이용함으로써 해결한다. 제안된 도구는 추상적 시각화 정보를 제공하므로 프로그램의 이해가 용이하고 효과적인 경합디버깅 환경을 제공한다. 시각화 도구는 Intel Pentium 4 단일 프로세서 시스템의 windows XP 환경에서 C++ 언어와 OpenGL을 사용하여 구축된다. OpenGL은 실시간 3D 그래픽스를 구현하기 위한 것으로 이식이 쉽고 속도가 빠른 그래픽 라이브러리이다.

2절에서는 OpenMP 프로그램과 기존의 시각화 도구를 소개하며, 3절에서는 본 논문에서 제안하는 효과적인 경합시각화를 위한 3차원적 추상화 기법을 보이며, 4절에서는 3차원적 추상화 기법이 적용된 시각화 도구의 구현을 보인다. 그리고 마지막 절에서는 결론 및 향후과제를 보인다.

2. 연구배경

본 절에서는 OpenMP 프로그램과 병렬프로그램의 수행구조를 시각화하기 위해서 사용되는 그래프와 병렬프로그램에서 발생되는 오류인 경합에 대해 살펴보고, 경합디버깅을 위한 기존의 시각화 도구를 소개한다.

2.1 OpenMP 병렬 프로그램

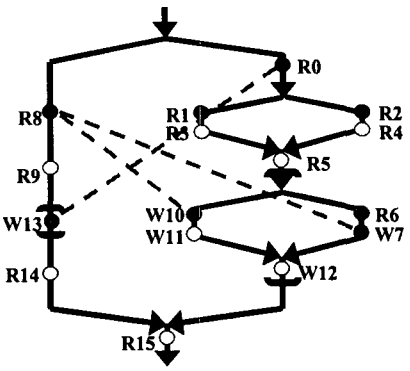
OpenMP는 공유메모리를 사용하는 병렬프로그래밍을 위한 산업표준 API로써 표준 C/C++와 FORTRAN 77/90을 확장하는 디렉티브(directives)와 라이브러리(library)들의 집합이다. 디렉티브로는 스레드 그룹을 생성하는 'Parallel For' 등의 병렬화 디렉티브와 병렬스레드간의 수행순서를 제어하는 'Critical', 'Barrier' 등의 동기화 디렉티브가 있으며, 라이브러리에는 프로그램 수행 시에 필요한 스레드들을 할당받기 위한 omp_set_num_thread(N) 등의 실행환경과 locking을 위한 루틴이 있다. [그림 1]은 C언어를 기반으로 작성된 OpenMP 프로그램이다. 이 예제 프로그램 1번째 줄의 '#pragma omp parallel for'는 병렬스레드 그룹을 생성(fork)하고, 15번째 줄의 괄호가 끝나는 곳에서 합류(join)한다.

OpenMP 프로그램은 방향성이 있는 비순환 그래프인 POEG (Partial Order Execution Graph)을 통해서 경합, 임계구역[3], 병렬영역 내에 다른 병렬영역을 가지는 내포병렬성[9] 등을 설명할 수 있다. [그림 2]는 [그림 1]의 프로그램 수행구조를 POEG으로 나타낸 것이다. 그림에서 정점은 병렬스레드의 생성/합류 지점을 나타내고, 간선은 정점으로부터 수행되는 스레드 수행을 의미한다. POEG은 병렬프로그램의 수행 시 스레드들간의 부분적인 순서를 나타내므로, 스레드들의 발생후(happen before) 관계를 알 수 있다. 두 스레드간에 경로가 존재하면 두

* 학생회원: 경상대학교 석사과정
** 학생회원: 경상대학교 박사과정
*** 종신회원: 경상대학교 컴퓨터과학과 교수
(컴퓨터·정보통신연구소 연구원)

```
 1: #pragma omp parallel for shared(x)
 2:  for(i=0; i<2; i++) {
 3:    if(i==0) {
 4:      y = x + 1 ;
 5:     ...
 6: }
 7: else if(i==1) {
 8: #pragma omp parallel for
 9:    for(j=0; j<2; j++) {
10:     ...
11: #pragma omp critical
12: #pragma omp parallel for
13:    for(k=0; k<2; k++) {
14:     ...
15:    }
16:   z = x + y
```

[그림 1] OpenMP 프로그램



[그림 2] POEG (Partial Order Execution Graph)

스레드는 "순서화된다"라고 하고, 그렇지 않으면 "병행하다"라고 한다. 예를 들어, [그림 2]의 R0과 W10은 서로 경로가 존재함으로 순서적으로 수행되는 반면, R0와 W13은 서로 경로가 존재하지 않으므로 병행하게 수행된다.

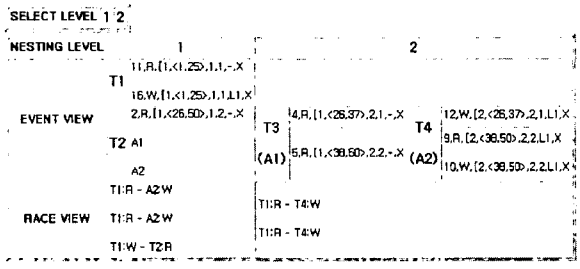
경합은 병행하는 스레드들 사이에서 하나의 공유변수에 적어도 하나의 쓰기사건을 가지는 접근사건의 쌍에 의해 발생한다. [그림 2]에서 점선은 다른 경합에 영향을 받지 않고 발생하는 경합들 중에서 처음으로 발생하는 최초경합[5, 6]을 의미한다.

2.2 경합디버깅을 위한 기존의 시각화 도구

PF-View[9]는 Parallel Fortran Program의 수행정보를 시각화하는 도구로써, 기존의 IBM's PF-Trace를 기반으로 개발되었다. 이 도구는 Postprocessor을 통해 PF-Trace가 제공하는 텍스트 기반의 정보 중에 필요한 정보만을 선택하여 프로그램의 수행중 동작을 그래픽으로 구조화하며, lock/event 동기화와 성능분석에 대한 상세정보도 필요시에 제공된다. 계층적으로 구조화된 소스코드는 시각화를 제공하는 window와 상호작용이 가능토록 함으로써 사용자가 프로그램의 수행구조를 빠르고 쉽게 이해할 수 있도록 한다.

TraceViewer[4]는 Parallel Fortran Program의 수행구조 및 경합보고를 위한 시각화 도구이다. 이 도구는 이벤트 노드와 이벤트간의 순서관계를 표현하는 간선들로 이루어진 event history graph, 소스코드 window, 경합보고를 위한 text window 등으로 구성되어 있다. 디버깅 중에 사용자가 노드를 클릭하면 해당 노드와 그에 대응되는 소스 라인이 highlight된다. 사용자는 AND-OR 논리 조합기능과 Before, Concurrent, After, Go, Step 등의 기능버튼을 통해서 브라우저를 제어한다.

Casuality Graph를 이용한 시각화[10]는 C언어를 확장한 Nicke라는 message passing program을 대상으로 하며, 프로세서가 많은 시스템 환경에서 프로그램의 수행을 이해하기 용이하도록 수행정보를 시각화하는 통합 뷰를 제공한다. 수행 중에



[그림 3] 텍스트 시각화

정보를 생성하고, runtime 부담을 줄이기 위해 추적 대상을 프로세스간 통신과 프로세스의 생성에 관한 정보로 제한한다. casuality graph를 구성하는 관련 이벤트 노드들은 supernode로 추상화되고, 수행상태는 세 가지 집합(past, present, future)으로 분류하여 순차 디버거에서와 동일한 디버깅 기능을 forward/backward single stepping로써 제공한다.

그러나 이러한 기존의 시각화 도구들은 시각화를 위한 공간이 제한적이므로 이벤트 및 스레드의 추상적 시각화를 효과적으로 제공하지 못한다.

3. 효과적 경합 시각화

본 절에서는 OpenMP 프로그램의 수행 시에 생성된 정보의 복잡성을 해결하기 위한 텍스트 추상화 기법, 시각화 결과와 경합정보의 복잡성을 제어하기 위한 그래프 추상화 기법에 대하여 소개한다.

3.1 텍스트 추상화 기법

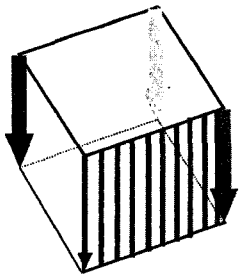
텍스트에서 스레드 추상화는 임의 내포수준에서 각 스레드들간의 병행성 여부를 타스크들의 나열 방향을 통해 보여준다. 스레드가 행(row) 방향으로 나열되면 그 스레드들은 서로 병행관계에 있으며, 열(column) 방향으로 나열되면 순서화 관계에 있음을 의미한다. [그림 3]에서, T1과 T2스레드는 행으로 나열되므로 서로 병행 관계에 있는 반면에, T3과 T4스레드는 열방향으로 표현되므로 서로 순서화 관계에 있음을 알 수 있다.

이벤트 추상화는 내포수준이 1이하인 스레드들을 특정 심볼로 변환하여 시각화함으로써 가능하다. 각 사건들은 소스코드의 라인번호, 사건 타입, 단일방향구역, 내포수준, 루프 인덱스, 락 변수, 각 사건 타입별로 필요한 필드 등으로 구성되어 있으며, 내포 병렬루프는 A1, A2, ..., Ai 등의 심볼로 추상화한다. [그림 3]에서 T2 스레드는 읽기사건 하나와 A1과 A2로 추상화된 내포 병렬루프 두 개가 있다. A1은 2개의 스레드에서 각각 읽기사건 하나씩을 가지는 병렬루프이며, A2는 하나의 쓰기사건을 가지는 스레드와 읽기/쓰기사건을 각각 하나씩 가지는 스레드로 구성된 병렬루프임을 알 수 있다.

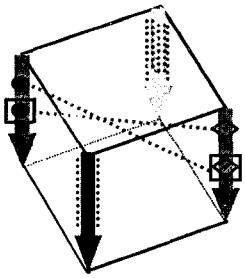
텍스트 추상화 기법은 내포수준을 선택할 수 있는 기능을 두어서 사용자가 각 단계별로 이벤트와 경합을 확인할 수 있으며, 스레드간의 병행성 여부를 쉽게 파악할 수가 있다.

3.2 그래프 추상화 기법

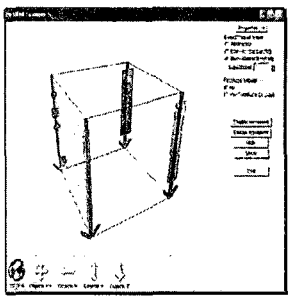
그래프에서 스레드 추상화 기법은 내포수준이 1인 스레드를 일정 개수로 그룹화하여 superthread로 대체함으로써 가능하다. 이때 내포수준 1 이하의 스레드들은 특정 심볼로 변환된다. superthread는 최대 4개로써 정육면체의 각 세로선 모서리에 그려지며, superthread의 오른쪽 세로면을 통해 확장 가능하다.



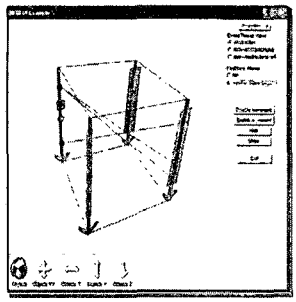
[그림 4] 스레드 추상화



[그림 5] 이벤트 추상화



[그림 6] 이벤트/스레드 뷰



[그림 7] 경합 뷰

[그림 4]는 스레드가 총 40개이며 그중 30개의 스레드가 3개의 superthread로 추상화되고, 나머지 스레드는 추상화 이전의 모습으로 확장된 예제이다.

이벤트 추상화는 스레드들에서 발생되는 각 이벤트들을 특정 심볼로 대체함으로써 가능하다. 이때 superthread에서 발생되는 이벤트가 50개 이상일 경우에는 심볼 대신 점으로 표시하게 된다. 읽기/쓰기사건은 원으로, 내포된 병렬루프는 마름모로, 임계구역은 사각형으로 대체되는데, 이는 각각 선택사건과 내포 병렬루프를 포함하는 임계구역으로 구분된다. 추상화된 내포 병렬루프는 사용자 요구에 따라 구체적인 시각화가 가능하다. [그림 5]에서 좌측 superthread에는 읽기(쓰기)사건과 임계구역이 하나 있으며, 우측 superthread에는 내포 병렬루프와 내포병렬루프를 포함하는 임계구역 하나가 추상화되어 있다.

그래프에서 경합에 참여하는 이벤트 심볼들은 점선으로 연결되고, 이들 경합에 대한 상세정보는 점선을 클릭 시 제공한다. 그래프 추상화 기법은 사용자가 임의의 스레드에서 어떤 이벤트가 경합을 이루는지를 쉽게 파악할 수 있으므로 효과적인 경합 디버깅을 가능하게 한다.

4. 3차원적 시각화 도구의 구현

제안된 추상적 시각화 기법을 이용하여 이벤트/스레드 뷰와 최초경합 뷰로 구성된 시각화 도구가 구현되었다

이벤트/스레드 뷰는 프로그램 수행 시에 생성되는 병렬 스레드들과 최초경합과 관련된 읽기/쓰기사건, 임계구역과 내포병렬루프 등을 정육면체를 이용하여 3차원적으로 시각화한다. 이 뷰는 Abstraction와 Non-Abstraction 상태를 선택적으로 볼 수 있으며, 추상화된 스레드 및 이벤트는 사용자의 요구에 따라 확장될 수 있다. 또한 Objects, Objects X 등의 기능버튼을 통해 객체의 3차원적 회전, 이동, Zoom In/Out 등이 가능하다. [그림 6]은 스레드의 일부만 superthread로 추상화한 결과를 보여주고 있다.

최초경합 뷰는 병렬프로그램의 수행 중에 탐지된 최초경합들을 선택사건과 함께 시각화한다. 이 뷰에서 최초경합은 경합에 참여하는 사건들의 쌍을 점선으로 연결함으로써 나타낸다. [그림 7]은 두 개의 superthread에 있는 3쌍의 사건들이 최초경합을 이룬다는 것을 알 수 있다. 이 뷰에서도 이벤트/스레드 뷰에서와 마찬가지로 기능버튼의 조작을 통해 객체를 임의대로 움직이면서 관찰할 수 있기 때문에 경합의 위치를 파악하기 용이하다. 그리고 점선을 클릭할 경우 그 최초경합에 참여되는 사건의 타입과 원시코드에서의 발생위치 등의 정보를 제공함으로써 대상 프로그램에 대하여 효과적인 경합 수정이 가능하게 한다.

5. 결론

병렬프로그램에서 경합은 비결정적인 수행결과를 초래하기 때문에 디버깅을 위하여 반드시 탐지되어야 한다. 그러나 기존의 시각화 도구는 시각화 및 디버깅 정보의 시각적 복잡성과 공간적 제약성으로 인해 프로그램의 복잡한 수행구조를 효과적으로 보여주지 못한다.

본 논문에서 제안된 시각화 도구는 내포된 병렬루프와 임계구역을 가지는 OpenMP 프로그램을 대상으로 하며, 선택사건과 최초경합 등의 경합 탐지정보를 시각화 한다. 그리고 시각적 복잡성 문제는 스레드와 이벤트의 추상화 기법을 제공함으로써 해결하고, 공간적 제약성의 문제는 xyz축 모두 사용하는 3차원적 시각화를 통해서 해결한다.

향후과제로는 내포병렬성의 깊이에 제한적이지 않으며, 본 연구실에서 개발한 디버깅 도구와 통합하여 웹 기반의 인터페이스를 제공하는 것이다.

참고문헌

[1] Citron D., D. G. Feitelson, and I. Exman, "Parallel Activity Roadmaps," Int'l Conf. on Parallel Computing (ParCo'93), Parallel Computing: Trends and Applications pp. 593-596, Elsevier Science, 1994.

[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, "Detecting Access Anomalies in Programs with Critical Sections," 2nd Workshop on Parallel and Distributed Debugging, pp. 85-96, ACM, May 1991.

[4] Helmbold, D. P., C. E. McDowell, and J. Wang, "TraceViewer: A Graphical Browser for Trace Analysis," TR-90-59, UCSC. 1990

[5] 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.

[6] Kim, J., D. Kim, and Y. Jun, "Scalable Visualization for Debugging Races in OpenMP Programs," Proc. of the 3rd Int'l Conf. on Communications in Computing (CIC), pp.259-265, Las Vegas, Nevada, June 2002.

[7] Kuhn, B., P. Petersen, and E. O'Toole, "OpenMP versus Threading in C/C++," EWOMP '99, Lund, Sweden, Sept. 1999.

[8] McDowell, C. E. and D. P. Helmbold, "Debugging Concurrent Programs," Computing Surveys, 21(4): 593-622, ACM, Dec. 1989.

[9] Sue, U. H. and C. M. Pancake, "Graphical Animation of Parallel Fortran Programs," Supercomputing '91, pp. 491-500, ACM/IEEE, Nov. 1991.

[10] Zernik, D., M. Snir, and D. Malki, "Using Visualization Tools to Understand Concurrency," Software, 9(3): 87-92, IEEE, May 1992.