병렬 프로그램의 메시지경합 탐지기법에 대한 시험도구

배수연, 박미영, 전용기
경상대학교 컴퓨터과학과
{javaio, park, jun}@race.gsnu.ac.kr

A Testbed for Message Race Detection Techniques of Parallel Programs

Su-Yun Bae, Mi-Young Park, Yong-Kee Jun
Dept. of Computer Science, Gyeongsang National University

요 약

메시지전달 병렬프로그램에서는 메시지들의 경합으로 인해서 의도하지 않은 비결정적인 수행결과가 초래되므로, 메시지들의 경합을 탐지하는 것은 중요하다. 이전 연구에서 경합을 탐지하는 다양한 기법들을 제안하였으나, 각 탐지기법의 기능을 검증할 수 있는 방법은 없다. 이는 기존 경합 탐지기법과 새롭게 개발되는 경합 탐지기법의 기능을 검증하기 어렵게 한다. 본 연구는 기존의 경합 탐지기법들을 구현하여 각 기법들의 탐지결과를 비교함으로써 그 기능을 검증할 수 있게 하는 시험도구를 제안한다. 본 시험도구의 사용자는 기존의 경합 탐지기법과 새로운 탐지기법을 선택하여 적용할 수 있고, 탐지된 결과에 대한 시각화 정보의 비교를 통해서 탐지기법들의 기능을 검증할 수 있다. 그러므로 본 도구는 새로운 경합 탐지기법의 개발을 위한 효과적인 기능시험을 가능하게 한다.

Ⅰ. 서론

병렬컴퓨터에서 널리 사용되고 있는 병렬프로그래밍 모델은 메시지전달[9]이다. 메시지전달 병렬프로그램에서는 두 개 이상의 송신자가 동시에 논리적으로 같은 채널을 통하여 수신자에게 메시지를 보낼 때 메시지경합[8]이 발생된다. 메시지경합은 의도하지 않은 비결정적인 수행결과를 초래하므로, 메시지들의 경합을 탐지하는 것은 중요하다.

경합탐지를 위한 기존의 기법은 크게 레이블링 기법과 프로토콜 기법으로 나눌 수 있다. 레이블링 기법은 병행하게 수행하는 프로세스에 대한 병행성 정보를 생성하는 기법으로서, Timestamp[4, 3], BD[1], CT[2], DV[12] 등이 있다. 프로토콜 기법은 두 송수신간의 경합여부를 결정하는 기법으로, 경합존재 탐지기법[8]과 최초경합 탐지기법[7]으로 나눌 수 있다. 경합존재 탐지기법은 수행 중인 프로그램에서 경합의 존재여부만을 탐지하는 반면에, 최초경합 탐지기법은 경합이 발생하는 최초의 수신사건만을 탐지하는 기법이다. 경합탐지를 위한 이러한 기법들이 다수 제안되고 있으나, 각 기법의 기능을 검증하는 방법은 아직 제시되고 있지 않다. 이것은 경합탐지를 위한 기존의 기법과 새롭게 개발되는 기법의 기능검증을 어렵게 만든다.

본 연구는 경합탐지를 위한 기존의 다양한 기법들을 구현하여, 각 기법의 탐지결과를 비교함으로써 기능을 검증하는 시험도구를 제안한다. 본 연구의 시험도구는 레이블링 기법으로 Timestamp, BD, DV 등의 기법을 제공하고, 프로토콜 기법으로 기존의 경합존재 탐지기법과 최초경합 탐지기법을 제공한다. 또한 본 연구에서 개발 중인 최초경합 탐지기법을 제공하여 그 기능을 검증한다. 본 도구의 사용자는 경합탐지를 위한 기존의 기법과 새로운 기법들을 선택하여 사용할 수 있고, 시각화[6, 11]된 경합정보의 비교를 통해서 기법들간의 기능을 시각적으로 검증할 수 있다.

본 시험도구에서 제공하는 여러 기법을 사용하여 MPI 병렬프로그램에서 발생되는 경합을 탐지하고, 그 결과를 비교 분석함으로써 각각의 기능을 검증할 수 있다. 또한 새로운 기법은 검증된 기존의 기법과의 비교를 통해서 효과적인 기능시험을 할 수 있다.

본 논문에서 제시하는 도구는 메시지전달 병렬프로그램의 표준 라이브러리인 MPI[9] 병렬프로그램을 대상 프로그램으로 하며, 시험도구의 개발 언어로는 자바를 사용한다. 본 도구가 수행되는 플랫폼은 네 개의 노드로 구성된 Alpha 클러스터 시스템으로서, MPI의 표준을 그대로 구현한 MPICH[5]가 설치된 환경이다. 이어지는 Ⅱ절에서 메시지경합과 경합탐지를 위한 기존의 여러 가지 기법들을 소개한다. Ⅲ절에서는 시험도구의 설계와 시험도구를 사용해서 기법들에 대한 검증 예를 보인다. 그리고 마지막으로 Ⅴ절에서 결론을 내린다.

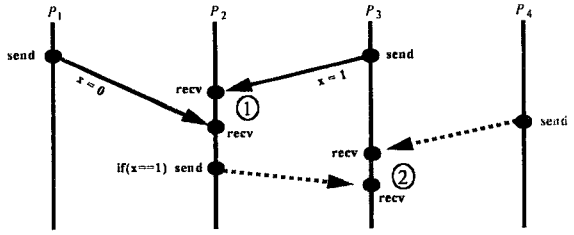
Ⅱ. 연구배경

본 절에서는 메시지전달 프로그램에서 발생하는 메시지경합에 대해서 설명하고, 경합탐지를 위해 제시된 기존의 기법들에 대해서 살펴본다.

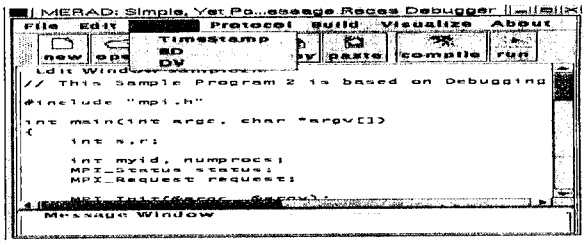
2.1 메시지경합

메시지전달 프로그램[9]의 의도하지 않은 비결정적 수행을 야기하는 주된 오류 중의 하나가 메시지경합이다. 메시지경합은 두 개 이상의 송신자가 논리적으로 같은 채널로 메시지를 동시에 송신할 때 발생되며, 다른 경합과의 관계에 의해서 인위적 경합과 비인위적 경합으로 구분된다[7]. 인위적 경합은 먼저 발생한 경합으로부터 영향을 받은 경합으로서 비실제적인 경합이다. 반면에 비인위적 경합은 다른 경합으로부터 영향을 받지 않은 경합으로서, 의도하지 않은 비결정적 수행결과를 초래한다.

<그림 2-1>는 인위적 경합과 비인위적 경합을 가지는 메시지전달 프로그램의 부분수행이다. 그림에서 P_1, P_2, P_3, P_4는 병행하게 수행하는 프로세스를 나타내고, 각 정점은 송수신사건을 의미하며 프로세스간의 화살표는 메시지들의 전달을 나타낸다. <그림 2-1>에서 P_1과 P_3이 P_2에게 메시지를 보내고, P_2와 P_4가 P_3으로 메시지를 전송한다. 이때 그림의 ①, ②에서 수신되는 각각의 메시지들은 그들의 도착순서가 보장되지 않고 비결정적으로 수신되므로 ①과 ②에서 경합이 발생한다. 여기서



<그림 2-1> 메시지경합



<그림 3-1> 레이블링 기법

②의 경우는 (x=1) 조건에 의해서 발생되는데, x값은 ①에 도착하는 메시지들의 순서에 의해서 결정된다. 즉, ②는 먼저 발생한 경합 ①로부터 영향을 받는 인위적인 경합으로서, ①을 탐지하여 디버깅하면 자연히 사라질 수 있는 경합이다. 반면에 ①은 다른 경합으로부터 영향 받지 않는 비인위적인 경합으로서, 반드시 탐지되어 디버깅되어야 한다.

비인위적 경합들 중에서 가장 먼저 발생하는 경합을 최초경합[7]이라 한다. 그림에서 ①은 최초경합인 비인위적 경합이다. 최초경합의 디버깅은 최초경합 이후에 발생하는 인위적 경합을 사라지게 하므로 디버깅에 효과적인 경합이다.

2.2 경합 탐지기법

병렬프로그램에서 동적으로 메시지경합을 탐지하는 기법으로는 사후추적 기법[10], 수행중 탐지기법[8, 7] 등이 있다. 사후추적 기법은 프로그램 수행이 종료된 후에 추적파일을 분석하여 경합을 보고하는 기법으로서, 수행시간이 긴 병렬 프로그램에서는 대용량의 기억장소를 요구하므로 비현실적이다. 반면에 수행중 탐지기법은 프로그램 수행 중에 경합을 탐지하는 기법으로서, 수행 중에 불필요한 정보들은 제거되므로 사후추적 기법보다는 공간복잡도 측면에서 이점이 있다.

수행중 경합 탐지기법으로는 병행성 정보를 생성하는 레이블링 기법과 레이블정보로 경합을 탐지하는 프로토콜 기법 등이 있다. 레이블링 기법은 병행하게 수행하는 프로세스에 대한 병행성 정보를 생성하여 병행성 여부를 판단하는 기법이다. 병행성 정보를 나타내는 레이블 값은 각 송수신 사건마다 생성되며, 송신사건이 발생할 때 메시지와 함께 전송된다. 기존의 레이블링 기법으로는 Timestamp[4, 3], BD[1], CT[2], DV[12] 등이 있다. Timestamp은 메시지전달 프로그램에서 보편적으로 사용되는 기법으로서, 내포 병렬성을 허용하지 않는다. BD와 CT 레이블링 기법은 내포병렬성을 허용하는 기법으로서, BD는 공유메모리 병렬프로그램을 위한 기법[1]이고, CT는 분산메모리의 메시지전달 프로그램을 위한 기법[2]이다. BD 레이블링 기법에서 병행성 여부를 판단하는데 요구되는 시간 복잡도를 개선한 레이블링 기법이 DV[12]이다.

레이블링 기법을 이용하여 경합을 탐지하는 프로토콜 기법은 수신사건 때마다 수행되며, 이전 수신사건과 송신사건간의 병행성 여부를 판단하여 경합을 탐지한다. 경합탐지를 위한 프로토콜 기법은 경합존재 탐지기법[8]과 최초경합 탐지기법[7]으로 나눌 수 있다. 경합존재 탐지기법은 수행 중인 프로그램에서 경합의 존재 여부만을 판단할 수 있는 기법이다. 반면에 최초경합 탐지기법은 경합이 발생하는 최초의 수신사건을 탐지하는 기법으로, 이때 탐지된 경합은 비인위적인 경합이다. 최초경합 탐지기법은 사후추적 기법과 수행중 기법이 혼합된 형태로서, 경합탐지를 위해 프로그램을 두 번 수행해야 한다. 첫 번째 수행에서는 최초경합 탐지에 필요한 정보수집을 하고, 두 번째 수행에서는 첫 번째 수행에서 수집된 정보를 이용해서 최초경합을 탐지한다.

이와 같이 기존연구에서 경합탐지를 위한 다양한 기법들이 제시되고 있으나, 현재까지 각 기법들의 기능적 정확성을 검증할 수 있는 방안은 제시되고 있지 않다. 이것은 경합탐지를 위한 기존의 기법에 대한 기능검증과 새롭게 개발되는 기법의 기능검증을 어렵게 만든다. 그러므로 경합탐지를 위해 제시되는 각 기법들의 기능을 검증할 수 있는 도구가 요구된다.

Ⅲ. 경합 탐지기법을 위한 시험도구

본 연구는 기존의 경합 탐지기법들을 구현하여 각 기법들의 탐지결과를 비교함으로써, 그 기능을 검증할 수 있는 시험도구를 제안한다. 본 절에서는 시험도구가 수행되는 환경과 시험도구의 각 기능들에 대해 살펴본다. 그리고 경합탐지를 위한 기존의 기법과 새로운 기법의 기능검증을 수행하는 예를 보인다.

3.1 시험도구의 설계

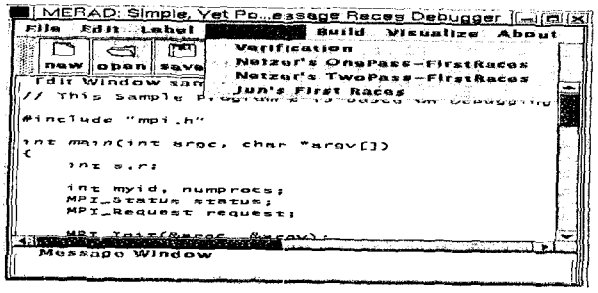
본 논문의 시험도구는 산업 표준화되어 널리 사용되고 있는 MPI[9] 병렬프로그램을 대상 프로그램으로 한다. MPI 병렬프로그램의 수행환경은 네 개의 노드로 구성된 Alpha 클러스터 시스템으로 하였고, 각 노드에는 MPI 표준을 그대로 구현한 MPICH[5]를 설치하였다. 그리고 시험도구의 개발 언어로는 자바를 사용하였다.

기능검증을 수행하는 시험도구의 주 메뉴 구성은 대상 프로그램을 관리하는 메뉴, 적용할 기법을 선택하는 메뉴, 그리고 선택된 기법을 실행하고 그 결과의 시각화를 위한 메뉴 등으로 구성된다. 대상 프로그램의 관리를 위해 제공되는 File 메뉴는 경합탐지를 위한 기존의 기법을 적용할 대상 프로그램을 열기 위한 것으로, 일반적인 파일관련 부 메뉴 외에도 Simple, Benchmark 등의 부 메뉴를 추가적으로 제공한다. Simple 부 메뉴는 본 도구의 사용 예를 보이기 위한 간단한 예제 프로그램을 제공하기 위한 것이며, Benchmark 부 메뉴는 NAS 벤치마크를 포함한 표준 벤치마크를 제공하여, 사용자가 각 기법들을 적용할 수 있도록 하기 위한 것이다.

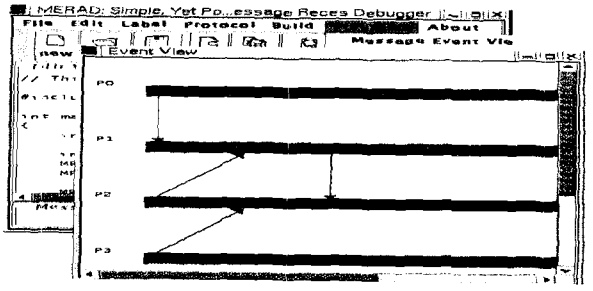
검증대상에 적용할 기법을 선택하는 메뉴로는 Label 메뉴, Protocol 메뉴 등으로 구성된다. Label 메뉴는 기능검증의 대상이 되는 레이블링 기법들로 구성되며, Timestamp, BD, DV 등이 부 메뉴로 제공된다. Protocol 메뉴는 검증대상이 되는 프로토콜 기법 등으로 구성되며, 경합존재 탐지기법과 Netzer의 최초경합 탐지기법 등을 제공한다. 또한 새롭게 개발되는 기법도 등록하여 타 기법과의 비교검증도 가능하다. 본 논문에서는 현재 개발 중인 기법을 "Jun's First Races"라는 메뉴로 제공하여 기능을 검증한다. 본 도구를 사용해서 사용자는 Label 메뉴와 Protocol 메뉴에서 제공되는 기법들의 조합과 그 수행결과의 비교로써, 검증하고자 하는 기법의 기능검증을 수행한다.

마지막으로, 선택된 탐지기법을 실행하고 실행 결과를 시각화하는 메뉴로서 Build와 Visualize 메뉴가 제공된다. Build 메뉴는 Compile과 Run으로 구성되며, 선택된 레이블링 기법과 프로토콜 기법으로써 변형된 대상 프로그램을 컴파일하고 수행한다. 컴파일 및 실행결과는 도구 하단에 있는 Message Window에서 텍스트 형태의 자료로 제공된다. Visualize 메뉴는 프로그램의 수행결과를 시각적으로 표현하기 위한 것으로, Event View와 Race View 등으로 구성된다. Event View는 프로그램 수행 중에 발생한 모든 송수신 사건을 나타내며, Race View는 탐지된 경합을 시각적으로 보이는 기능이다.

시험도구의 사용 예는 <그림 2-1> 프로그램 수행으로 보인



<그림 3-2> 프로토콜 기법



<그림 3-3> 시각화

다. <그림 3-1>은 <그림 2-1>에 해당하는 MPI 병렬프로그램에 레이블링 기법을 선택하여 적용하는 그림이다. <그림 3-2>는 레이블링 기법을 선택한 후 프로토콜 기법을 적용하는 그림으로서, 레이블링 기법과 프로토콜 기법을 선택한 후에 컴파일 과정을 거쳐 수행을 하게 된다. 이때 수행되는 프로그램은 수행중 경합 탐지기법으로 경합을 탐지하게 되며, 프로그램 수행이 종료된 후 Visualize 메뉴를 통해서 수행 중에 발생한 송수신 사건들을 시각화한다. <그림 3-3>는 <그림 2-1> 프로그램 수행에 대한 Event View로서, 텍스트 형태의 정보보다 효과적으로 기능을 검증할 수 있게 한다.

3.2 시험도구를 이용한 검증

본 시험도구를 통한 각 기법들의 기능검증은 비교검증으로써 수행되므로, 본 연구에서는 비교기준으로 이용하기 위하여 이미 검증된 Timestamp 레이블링 기법과 경합존재 탐지기법을 제공한다. 다른 기법들을 위한 비교검증은 먼저 레이블링 기법을 대상으로 수행하며, 검증된 레이블링 기법을 프로토콜 기법에 적용하여 수행결과를 비교한다.

레이블링 기법은 각 기법에 대해서 검증된 프로토콜을 동일하게 적용하여 그 결과를 비교함으로써 검증된다. 기존의 BD, DV 등과 같은 레이블링 기법이나, 새롭게 개발된 레이블링 기법은 검증된 Timestamp 기법의 수행결과와 비교되어 동일여부를 확인함으로써 기능이 검증된다. 이것은 레이블링 기법이 각 프로세스에 부여하는 병행성 정보는 다르지만, 병행하게 수행되는 사건들에 대한 프로토콜의 기능은 동일하기 때문에 가능하다.

프로토콜 기법으로 제공되는 최초경합 탐지기법이나, 새롭게 개발될 프로토콜 기법은 경합존재 탐지기법의 수행결과와 비교함으로써 그 기능을 검증할 수 있다. 최초경합 탐지기법의 검증은 경합존재 탐지에서 보고되는 수신사건들 중에서 가장 먼저 발생한 수신사건과 최초경합 탐지기법이 탐지한 수신사건을 비교하여 기능을 검증한다. 또는 검증된 최초경합 탐지기법이 있을 경우에는 그 기법과 새로이 개발될 기법의 수행결과를 비교함으로써 그 기능을 검증할 수도 있다.

Ⅳ 결론 및 향후과제

본 연구는 경합탐지를 위한 기존의 다양한 기법들을 구현하여, 각 기법의 탐지결과를 비교함으로써 기능을 검증하는 시험도구를 제안한다. 본 연구의 시험도구는 레이블링 기법으로 Timestamp, BD, DV 등의 기법을 제공하고, 프로토콜 기법으로 기존의 경합존재 탐지기법과 최초경합 탐지기법을 제공한다. 또한 본 연구에서 개발 중인 최초경합 탐지기법을 제공하여 사용자는 경합탐지를 위한 기존의 기법과 새로운 기법들을 선택하여 사용할 수 있고, 시각화된 경합정보의 비교를 통해서 기법들간의 기능을 시각적으로 검증할 수 있다. 향후 과제로는 사용자에게 편리한 디버깅 환경을 제공하기 위하여 플랫폼에 독립적인 웹기반 사용자 인터페이스가 요구된다.

참고문헌

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

[2] Audenaert, K., "Clock Trees: Logical Clocks for Programs with Nested Parallelism," Tr. on Software Engineering, 23(10): 646-658, IEEE, Oct. 1997.

[3] Bechini, A., and K. C. Tai, "Timestamps for Programs using Messages and Shared variables," Int'l Conf. on Distributed Computing Systems, pp. 266-273, IEEE, May 1998.

[4] Fidge, C. J., "Partial Orders for Parallel Debugging," SIGPLAN/SIGOPS Workshop on Parallel and Distributed Debugging, pp. 183-194, Madison, WI, May 1988.

[5] Gropp, W. and E. Lusk, User's Guide for Mpich, A Portable Implementation of MPI, TR-ANL-96/6, Argonne National Laboratory, 1996.

[6] Kranzlmuller, D., and J. Volkert, "Why Debugging Parallel Programs Needs Visualization," Workshop on Visual Methods for Parallel and Distributed Programming, Seattle, WA, Sept. 2000.

[7] Netzer, R. H. B., T. W. Brennan, and S. K. Damodaran-Kamal, "Debugging Race Condition in Message-Passing Programs," SIGMETRICS Symp. on Parallel and Distributed Tools, pp. 31-40, Philadelphia, PA, ACM, May 1996.

[8] Netzer, R. H. B., and B. P. Miller, "Optimal Tracing and Replay for Debugging Message-Passing Parallel Programs," Supercomputing 92, pp. 502-511, Minneapolis, MN, Nov. 1992.

[9] Snir, M., S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra, MPI-The Complete Reference: Volume 1, The MPI Core, 2nd Ed., Cambridge, MA, MIT Press, 1998.

[10] Tai, K. C., "Race Analysis of Traces of Asynchronous Message-Passing Programs," 17th Int'l Conf. on Distributed Computing Systems, pp. 261-268, IEEE, May 1997.

[11] 박미영, 김영주, 김성대, 이승렬, 박소희, 전용기, "메시지전달 프로그램의 디버깅을 위한 경합조건의 시각화", 컴퓨터시스템연구회 추계학술발표회논문집, pp. 47-56, 한국정보과학회, 2000. 9.

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