리눅스 클러스터 시스템을 위한 병렬프로그램의 메시지경합 탐지 도구*

박소희, 김영주, 박미영, 김성대, 이승렬, 전용기†
경상대학교 컴퓨터과학과
shpark@race.gsnu.ac.kr

A Detection Tool of Message Races in Parallel Programs for Linux Cluster Systems*

S. Park, Y. Kim, M. Park, S. Kim, S. Lee, Y. Jun†
Dept. of Computer Science, Gyeongsang National University

요 약

병렬 오류인 메시지경합을 가진 메시지전달 프로그램은 비결정적인 수행결과를 보이므로, 이를 탐지하고 수정하는 것이 어렵다. 기존의 메시지경합 탐지 도구들은 메시지경합에 관련된 간접적인 정보를 제공하는 수준이며, 메시지경합의 원인을 자동으로 탐지하지 못한다. 그리고 탐지 과정 중에 부가적인 메시지전달 작업이 발생하며, 대상 프로그램을 수정해야 하는 부담이 있다. 본 논문에서 제안된 탐지 도구는 리눅스 클러스터 시스템을 위한 병렬 프로그램의 메시지경합을 자동으로 탐지하여 직접적인 경합 정보를 제공한다. 그리고 탐지 엔진 부분을 리눅스 커널에 설치함으로써 경합 탐지를 위한 부가적인 메시지전달의 필요성을 제거하고, 대상 프로그램의 수정 없이 경합을 탐지할 수 있는 투명성을 제공한다.

1. 서 론

클러스터 시스템과 같은 병렬처리 시스템에서 수행되는 메시지전달(message-passing) 프로그램은 논리적 오류뿐만 아니라 메시지들간의 경합상태(race condition)나 교착상태(deadlock) 등의 병렬 오류로 인한 비결정적인(nondeterministic) 수행을 한다. 이러한 비결정적인 현상은 프로그래머가 의도하지 않은 결과를 초래하므로 반드시 탐지해야 한다. 이들 오류 중에서도 가장 탐지하기 어려운 메시지 경합상태(message race)는 두 개 이상의 메시지가 하나의 수신(receive) 프로세스에게 적절한 동기화 없이 동일한 채널로 전송(send)될 때 발생된다.

이러한 메시지경합들은 인위적 경합(artifact races), 비인위적 경합(nonartifact races), 얽힌 경합(tangled races) 등으로 구분된다. 인위적 경합은 이전에 발생한 경합의 영향을 받아 발생되는 경합으로서, 이전에 발생한 경합이 수정되면 발생되지 않을 수 있는 경합들이다. 비인위적 경합은 다른 경합에 영향받지 않고 발생되는 실제 경합으로서 비결정적인 수행의 직접적인 원인이 되므로 반드시 수정되어야 한다. 얽힌 경합은 두 개의 경합이 서로 영향을 주는 경합으로서, 적어도 하나는 실제 경합이므로 탐지해야 한다. 이들 경합들 중에서 다른 경합의 영향을 받지 않으면서 가장 먼저 발생하는 최초경합(first race)을 탐지하는 것은 중요하다. 그 이유는 최초경합이 수정되면 여기에 영향을 받는 다른 경합들이 사라질 수 있으므로 효과적인 수정이 가능하기 때문이다.

이러한 메시지경합들을 탐지하는 기존의 도구들은 메시지경합 상태와 관련된 간접적인 정보를 제공하는 수준이므로 효과적으로 경합을 수정하기가 어렵다[Tai95, DaFr94, CyLe95]. 본 논문에서는 메시지경합과 관련된 직접적인 정보를 제공하여 병렬 오류의 수정에 효과적인 최초경합을 자동으로 탐지하는 도구를 제안한다. 제안된 경합 탐지 도구는 리눅스 기반 PC 클러스터 시스템의 MPI(Message

* 본 연구는 정보통신부에서 지원하는 한국전자통신연구원 선도기반기초개발사업의 위탁연구로 수행되었음.
† 전산개발연구소, 정보통신연구센터 연구원

Passing Interface) 프로그램[SOHW96]을 대상으로 하고, 메시지경합 탐지를 위한 엔진 부분을 리눅스 커널에 설치한다. 그러므로 이 도구는 경합 탐지를 위한 부가적인 메시지전달의 필요성을 제거하고, 대상 프로그램의 수정없이 경합을 탐지할 수 있는 투명성을 제공한다.

2. 연구 배경

본 절에서는 MPI의 수행 모델과 탐지될 메시지 경합을 소개한다. 그리고 기존의 메시지경합 탐지에 대한 연구들을 소개하고, 본 탐지 도구에 응용될 기법들을 설명한다.

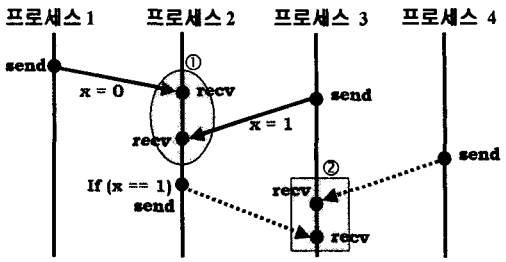
2.1 메시지전달 프로그램

메시지전달 프로그램 모델에서 널리 보편화된 라이브러리는 MPI와 PVM(Parallel Virtual Machine)이다. 특히 MPI는 메시지전달 라이브러리에 대한 표준 명세로서, 사용자에게 풍부하고 다양한 라이브러리를 제공하고 있으며, 높은 이식성과 성능으로 인해 널리 사용되고 있다. MPI의 수행 모델은 메시지전달 방법에 따라 송신측에서의 블록킹(blocking), 비블록킹(non-blocking), 수신측에서의 블록킹, 비블록킹 등으로 나눌 수 있으며, 그들간의 동기식 및 비동기식 수행 모델이 있다.

<그림 2-1>에서는 메시지전달 프로그램을 수행했을 때 발생된 경합들을 보여 주고 있다. 프로세스 1과 프로세스 3이 동일한 채널을 통해서 동시에 프로세스 2에게 메시지를 보내고, 마찬가지로 프로세스 3은 동시에 다른 프로세스로부터 메시지를 받게 되어 ①과 ②영역에서 메시지경합이 발생된다. 여기서 ①은 비인위적인 경합이면서 최초경합이므로 반드시 탐지해야 하는 대상이 된다. ②는 이전의 메시지경합에 의해 영향을 받는 인위적인 경합이므로 ①을 탐지하여 수정하면 발생되지 않을 수도 있다. 그러므로 본 논문에서는 메시지경합의 수정에 효과적인 최초경합을 탐지하여 보고하는 도구를 제안한다.

2.2 메시지경합 탐지

기존의 메시지경합 탐지 기법들을 탐지 시점에 의해 분류할 수 있다. 수행후 탐지 기법(post-mortem detection)[Tai95]의 경우는 다양한 메시지전달 모델에 적용될 수 있으나, 탐지되는 메시지경합에 대한 인위적 혹은 비인위적 여부를 구분하지 못하기 때문에 병렬 오류 수정에 효과적이지 못하다. 수행중 탐지 기법(on-the-fly detection)[DaFr94, CyLe95]의 경우에는 수행 시에 하나의 프로세스에만 관련된 메시지경합을 보고하므로 여전히 다른 프로세스들에서 영향받은 인위적 경합이 존재할 수 있다. 하지만 혼합 탐지 기법(hybrid detection)[NeBD96, NeMi92]의 경우는 수행 중에 생성된 정보를 기반으로 한다. 특히, [NeBD96]은 최초경합을 탐지하기 위해 2단계(two-pass) 알고리즘을 사용하는데, 최초경합의 탐지에 필요없는 많은 정보들을 고려하지 않으므로 공간적인 측면에서도 효율적이다.

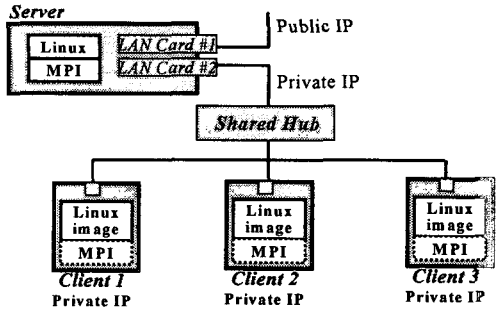


<그림 2-1> 탐지 대상의 메시지경합

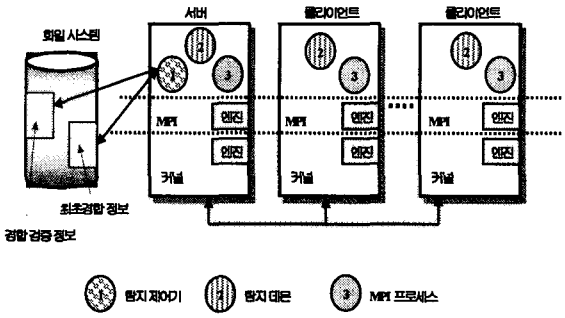
본 논문에서는 병렬 오류를 효과적으로 수정하기 위해 최초경합 탐지 기법[NeBD96]을 적용한다. 1단계에서는 프로그램 수행 중에 경합의 발생 여부만을 확인하게 되고, 2단계에서 동일한 입력으로 재수행하여 최초경합을 탐지하게 된다. 여기서, 수행 중에 발생된 사건을 검사하여 메시지경합 여부를 탐지하기 위한 병행성 정보인 레이블 생성 기법은 시간과 공간 복잡도 측면에서 가장 효율적인 NR 레이블링 기법[JuKo93]을 확장하여 적용한다. MPI 프로그램은 내포 병렬성이 없이 각 프로세스간에 메시지전달이 이루어지므로, 공유 메모리 기반에서 내포 병렬성을 위한 NR 레이블링을 확장하여, 각 프로세스의 정보와 함께 메시지전달에 의한 동기화 정보를 포함하는 새로운 레이블을 생성한다.

3. 클러스터 시스템의 메시지경합 탐지 도구

본 절에서는 대상 MPI 프로그램의 수행을 위해 구축한 리눅스 기반의 PC 클러스터 시스템과, 제안된 탐지 도구의 수행 구조를 보인다. 그리고 메시지경합의 수정에 가장 효과적인 최초경합을 탐지하여 보고하는 탐지 엔진의 수행 환경과 구조를 보여준다.



<그림 3-1> 리눅스 클러스터



<그림 3-2> 탐지 도구의 수행 구조

3.1 클러스터 환경

본 논문에서 설계한 탐지 도구를 실험하기 위해서 리눅스 기반의 PC 클러스터 시스템을 구축하였다. <그림 3-1>은 서버 1대와 클라이언트 3대로 구축된 클러스터 시스템의 구성도를 보여주고 있다. 일반적으로 클러스터 시스템의 구축 형태는 각 컴퓨터 노드마다 하드디스크가 있는 기법과, 서버에만 하드디스크가 있고 클라이언트는 하드디스크가 없는 기법 등의 두 가지로 나눌 수 있다. 클라이언트에 하드디스크가 없는 클러스터 시스템은 상대적으로 확장이 쉽고, 확장에 따른 부가적인 필요 자원이 적어서 비용을 절감할 수 있는 장점이 있다.

본 논문에서는 클라이언트에 하드디스크가 없는 기법을 사용하였다. 서버는 100Mbps를 지원하는 이더넷(ethernet)카드 2개가 설치된 이중(dual) CPU시스템으로, 리눅스와 MPI를 설치하였다. 각 클라이언트에는 각각 1개씩의 네트워크 카드를 설치하였으며, 부팅시 서버로부터 리눅스의 이미지 파일을 가져오게 된다. 서버의 네트워크 카드들은 각각 공인 IP와 비공인 IP로 지정되고, 클라이언트는 비공인 IP를 사용하게 된다.

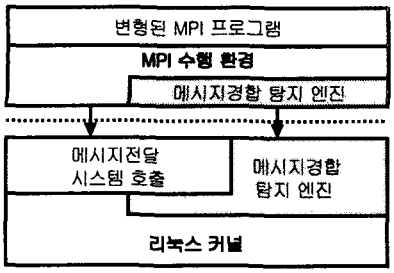
실험에서 사용된 메시지전달 라이브러리는 MPI의 구현들 중에서 이식성과 고성능을 목표로 개발된 MPICH를 사용한다. MPICH는 프리웨어로 제공되고 있으며, 다른 프리웨어들에 비해 소스 코드가 공개되어 있으며, 리눅스와 인텔 x86 시스템의 지원하면서도 MPI 명세를 완전히 구현하고 있다는 장점이 있다. MPICH의 특징으로는 높은 이식성을 제공하는 기존의 메시지전달 라이브러리들인 P4(Portable Programs for Parallel Processors), Chameleon, Zipcode 등을 이용하고 있으며, 분산 및 공유 메모리 클러스터 시스템에도 지원 가능하다. 또한 기본적인 구성은 세 가지 요소들의 계층구조로서, MPI 라이브러리, ADI(Abstract Device Interface), 채널 인터페이스 등으로 구현되어 있다.

3.2 탐지 도구

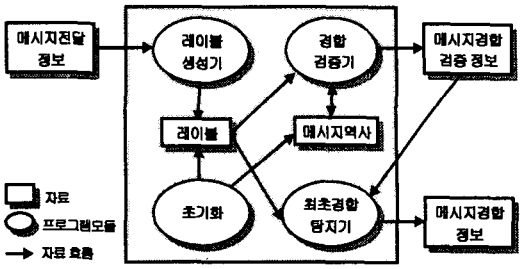
<그림 3-2>는 클러스터 시스템에서 메시지경합 탐지 도구의 수행 구조를 보여준다. 탐지 도구는 탐지 제어기, 탐지 엔진, 탐지 데몬 등의 세 가지 요소로서 구성된다. 이 중에서 탐지 제어기는 서버 노드에만 설치된다. 먼저, 서버가 부팅되고 난 후에, 클라이언트들이 부팅되어 탐지 데몬을 통해서 연결을 요청해 올 때까지 서버의 탐지 제어기는 경청(listen)하며 기다리게 된다. 클라이언트의 탐지 데몬이 서버에 통신을 요청하면, 서버는 파일시스템으로부터 MPI 프로그램을 포함하여 필요한 자료를 가져와 각 클라이언트에게 TCP 프로토콜에 의해 전송하게 된다. 각 클라이언트의 MPI 프로그램을 수행하는 프로세스가 프로그램 수행 중에 다른 클라이언트들에게 메시지전달 정보를 보내게 되면, 그 클라이언트들의 커널에 설치된 탐지 엔진이 구동하게 된다. 탐지 엔진은 경합의 존재 여부를 판단하고 최초 경합을 탐지하여 서버에게 전달하게 된다. 서버의 탐지 제어기는 전달받은 메시지경합을 파일시스템의 메시지경합 정보에 저장하게 된다.

3.3 탐지 엔진

<그림 3-3>에서와 같이 메시지경합 탐지 엔진의 수행은 MPI 수행 환경과 리눅스 커널에서 각각 이루어진다. MPI 수행 환경을 위한 탐지 엔진은 MPI 프로세스와 탐지 데몬을 위한 두 가지 기능을 수행한다. MPI 프로세스를 위해서는 레이블 및 메시지



<그림 3-3> 탐지 엔진의 수행 환경



<그림 3-4> 탐지 엔진의 수행 구조

역사를 초기화하고, 탐지 데몬을 위해서는 클러스터 시스템에서의 메시지전달을 위한 소켓(socket)을 개설한다. 리눅스 커널을 위한 탐지 엔진은 메시지경합을 탐지하기 위한 기본 입출력 호출과 경합 탐지 기능의 호출 등이 수행된다. 본 논문에서 메시지경합 탐지 엔진을 리눅스 커널에 설치하는 이유는 경합 탐지를 위한 부가적인 메시지전달이 필요 없게 되어 수행 속도가 향상되고, 대상 프로그램의 수정 없이 경합을 탐지할 수 있는 투명성을 제공하기 때문이다. 이를 위해 리눅스 커널을 부분적으로 수정하게 된다.

메시지경합 탐지 엔진은 MPI 수행 환경과 리눅스 커널로부터 메시지전달 정보를 입력받아 레이블을 생성하고, 메시지경합의 존재여부를 파악하여 최초경합에 대한 정보를 보고하게 된다. 탐지 엔진은 <그림 3-4>와 같이 레이블 생성기, 경합 검증기, 최초경합 탐지기, 초기화 프로그램 모듈로 구성된다. 레이블 생성기는 수행 중에 메시지경합의 여부를 판단하기 위해 필요한 병행성 정보인 레이블을 생성한다. 경합 검증기는 메시지경합 탐지 프로토콜에 의해 수행 중에 메시지경합의 존재 여부를 판단한다. 최초경합 탐지기는 경합 검증기의 수행 후에 같은 입력으로 프로그램을 재수행할 때, 파일 시스템에 저장된 경합 검증 정보를 통해 최초경합의 위치를 탐지하여 보고하게 된다.

4. 결 론

기존의 메시지경합 탐지 도구들은 경합상태와 관련된 간접적인 정보를 제공하는 수준이므로 효과적인 경합 탐지가 어렵다. 그리고 탐지 과정 중에 부가적인 메시지전달 작업이 발생하며, 대상 프로그램을 수정해야 하는 부담이 있다. 본 논문에서는 리눅스 클러스터 시스템을 위한 MPI 프로그램의 메시지경합을 자동으로 탐지하여 직접적인 경합 정보를 보고하는 탐지 도구를 제안한다. 이 도구는 메시지경합 수정에 효과적인 최초경합을 탐지하여 보고한다. 또한 탐지 엔진을 리눅스 커널에 설치함으로써 경합 탐지를 위한 부가적인 메시지전달의 필요성을 제거하고, 대상 프로그램의 수정없이 경합을 탐지할 수 있는 투명성을 제공한다. 이 도구는 리눅스 기반의 PC 클러스터 시스템에서 MPICH를 위해서 개발된다. 향후 과제는 메시지경합 탐지 엔진에서 생성된 경합 정보를 이용하여 사용자에게 편리한 디버깅 환경을 제공하도록 시각화하는 것이 필요하다.

참고 문헌

[JuKo93] Jun, Y., and K. Koh, "On-the-fly Detection of Access Anomalies in Nested Parallel Loops," 3rd Workshop on Parallel and Distributed Debugging, pp. 107-117, ACM, May 1993.

[DaFr94] Damodaran-Kamal, S.K., J.M. Francioni, "mdb: A Semantic Race Detection Tool for PVM," 8th Scalable High-Performance Computing Conference. pp. 702-709, IEEE, May 1994.

[NeBD96] Netzer, R. H. B., T. W. Brennan, S. K. Damodaran-Kamal, "Debugging Race Conditions in Message-Passing Program," SIGMETRICS Symp. on Parallel and Distrubuted Tools, ACM, May 1996.

[CyLe95] Cypher, R,. and E. Leu, "Efficient Race Detection for Message Passing Programs with Nonblocking Sends and Receives," 7th IEEE Symp. on Parallel and Distributed Processing, pp. 534-541, IEEE, 1995.

[NeMi92] Netzer, R. H. B., and B. P. Miller, "Optimal Tracing and Replay for Debugging Message-Passing Parallel Programs," Supercomputing '96, ACM/IEEE, 1992.

[Tai95] K. C. Tai, "Race Detection for Message -passing Programs," North Carolina State Univ., Computer Science, TR-96-04, 1995.

[SOHW96] Snir, M., S., Otto, S. Huss-Lederman, D. Walker, J. Dongarra, MPI: The Complete Reference, MIT Press, 1996.