이기종환경을 지원하는 분산디버거의 설계 및 구현

서영애
한국전자통신연구원, 컴퓨터.소프트웨어연구소
네트워크컴퓨팅연구부
Tel: 042-860-4982
Fax: 042-860-3900
E-mail: yaseo@etri.re.kr

Design and Implementation of Distributed Debugger Supporting Heterogeneous Environment

Young-Ae Seo
Network Computing Department
ETRI-Computer & Software Technology Laboratory
Tel: 042-860-4982
Fax: 042-860-3900
E-mail: yaseo@etri.re.kr

Abstract

In an ongoing project at ETRI-CSTL, A debugger for distributed programs that run on a varied collection of machines is being built. To build such debugger, a client-server model is incorporated. This strategy enables us to provide a unified user interface and isolate debugger core from the user interface. Several debugging servers running on a diverse set of platforms permit the implementation of a distributed debugger for heterogeneous environment, and the single debugging client provides unified debugging concept and graphical user interface over various servers. Also, the precise specification of the interaction protocol between the client and server facilitates client to be paired with a variety of server implementations. This paper describes the design and implementation of our debugger, concentrating on the system architecture.

1. 서론

분산시스템은 여러 대의 컴퓨터에 분산된 프로그램들이 상호 통신을 통해 협력하면서 작업을 수행하는 시스템이다. 이러한 분산시스템의 디버깅은 통신에 소요되는 시간을 예측할 수 없다는 점과 여러 프로그램들이 동시에 동작한다는 점 때문에 순차적 프로그램의 디버깅보다 훨씬 복잡하다. 최근, 중·소형급 컴퓨터가 다양해짐으로 인해 개발환경이 다양화되고 혼재되는 현상은 이를 더욱 어렵게 하고 있다.

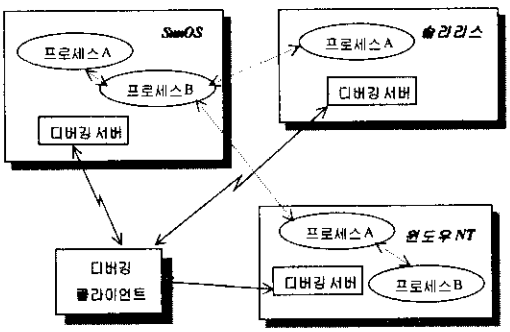
이러한 어려움을 해소하기 위하여 본 논문에서 기술하는 분산디버거인 유니뷰(UniView) 시스템은 이기종의 분산시스템을 효율적으로 디버깅할 수 있도록 하는 데 중점을 두어 개발되었다. 유니뷰 시스템의 설계 목표는 크게 다음의 두 가지로 요약될 수 있다[2].

1. 이식성 - 새로운 플랫폼이나 후단부 디버거에 추

가 확장될 수 있어야 하므로 특정 디버거에 의존하지 않는 이식 가능한 구조를 가져야 한다. 이를 위해 클라이언트는 특정 플랫폼에 의존하지 않게 설계되어야 하며 클라이언트와 서버간의 통신은 표준화된 기능이나 유틸리티를 사용하여 구현한다.

2. 사용의 편리성 - 이기종의 다른 디버거들에 단일화된 사용자 인터페이스를 제공한다. 사용자 인터페이스는 사용하기 쉬우면서도 기존 디버거들의 다양한 기능과 인터페이스를 충분히 고려한 높은 수준의 인터페이스를 제공한다.

본 논문에서는 클라이언트-서버 모델을 채택하여 유니뷰 시스템을 구현함으로써 상기 목표를 달성할 수 있었다. 그림 1 은 유니뷰 시스템을 사용하여 분산시스템을 디버깅하는 모델을 보여준다.



[그림 1] 유니뷰 시스템을 사용한 디버깅 모델

그림 1 에서 보듯이, 여러 개의 디버깅서버가 각기 다른 플랫폼상에서 동작함으로써 이기종의 분산환경을 지원할 수 있으며, 하나의 디버깅클라이언트가 여러 디버깅서버들과의 상호작용을 담당함으로써 사용자에게 높은 수준의 단일화된 인터페이스를 제공할 수 있게 하였다. 또한, 클라이언트와 서버간의 프로토콜을 명확히 정의함으로써, 클라이언트의 단일화된 사용자 인터페이스와 이기종 서버와의 연결이 가능하게 하여 다양한 플랫폼으로 확장될 수 있는 기반을 제공하였다. 이하에서는 분산디버거의 설계 및 구현에 관하여 클라이언트, 서버, 통신모듈별로 나누어 기술하고자 한다.

2. 디버깅서버

디버깅서버는 디버깅클라이언트와 통신하면서 각 호스트 상에서 동작하는 프로그램을 디버깅한다. 서버는 자신이 디버깅해야 할 프로세스들을 각 호스트별로 기존의 디버거를 이용하여 연동함으로써 간접적으로 디버깅을 수행한다[3]. 각 서버는 하나 이상의 후단부 디버거를 관리할 수 있는 구조로 되어 있는데, 이로 인하여 후단부 디버거가 하나의 프로세스만을 제어할 수 있는 경우에도 서버가 여러 개의 후단부 디버거를 제어하므로써 한 호스트에서의 멀티프로세스 디버깅이 가능하다.

이와 같이 기존 디버거를 후단부 디버거로 활용하는 구조는 분산디버거의 구축 시 디버깅 부분을 새로 구현하는 부담을 줄일 수 있고 플랫폼의 네이티브 디버거의 기능을 충분히 활용할 수 있는 장점이 있다. 또한, 후단부 디버거를 이용하여 플랫폼에 의존적인 부분을 최대한 분리시킴으로써, 서버의 이식성을 증대시킬 수 있는 이점을 가지게 된다.

서버의 내부 구성은 통신 모듈과 디버거 연결부, 그리고 서버의 내부 모듈로 나뉘어진다. 통신모듈은 클라이언트의 요청에 대해 이를 처리할 서버 내부 모듈의 해당 루틴을 호출하며, 호출된 루틴은 이를 일련의 후단부 디버거의 명령으로 변환하여 디버거 연결부에 요청한다. 디버거 연결부는 후단부 디버거에게 디버깅 명령을 입력으로 주고 그 결과를 받아 처리를 담당할 서버 내부 모듈의 루틴을 호출하는 구조로 되어 있다. 이상의 내부 구조는 특정 후단부 디버거에 의존적인 코드들을 디버거 연결부로 분리하여 후단부 디버거와의 연결방식 및 후단부 디버거의 특정 명령어나 결과 출력 방식 등을 처리할 수 있게 함으로써 서버의 내부 모듈들에 대한 재사용성을 높여 준다.

3. 디버깅클라이언트

디버깅클라이언트는 전체 분산시스템의 디버깅을 위한 사용자 인터페이스를 제공하고 디버깅서버들과 통신하면서 디버깅관련 정보들을 종괄 관리하는 역할을 수행한다. 유니뷰시스템의 클라이언트-서버 구조는 디버깅 클라이언트로 하여금 다양한 후단부 디버거들에 대해 동일한 사용자 인터페이스를 제공하고 분산된 여러 프로그램에 대한 디버깅의 제어를 가능하게 하였다.

디버깅클라이언트의 전체 구조 설계에서 가장 중요한 점은 사용자 인터페이스와 서버와의 통신을 담당하는 처리부가 독립되어야 한다는 점이다. 이를 위해 클라이언트는 통신을 관리하는 부분(EM:이벤트 매니저)과 사용자 인터페이스를 담당하는 부분 사이에 몇 개의 계층 구조를 갖도록 설계되었다. 즉, 서버와의 통신을 담당하는 통신관리 부분을 따로 두고, 디버깅 정보 및 논리를 관리하는 부분(DF:디버깅프레임워크계층)과 사용자인터페이스를 담당하는 부분(UI:사용자인터페이스 계층)으로 나누었으며, 이들 두 계층간의 상호 독립성을 높이기 위하여 인터페이스 클래스들을 가지는 계층(GF:그래픽프레임워크계층)을 두었다.

이러한 계층 구조를 통하여 디버깅클라이언트는 양질의 사용자인터페이스를 제공할 수 있었으며 여러 개의 서버를 동시에 관리하면서 처리해야 하는 성능상의 요구도 만족시킬 수 있었다.

4. 통신모듈 - 이벤트 매니저

유니뷰 시스템의 서버와 클라이언트간의 서비스 요청과 요청된 서비스의 제공은 이벤트에 의해 이루어진다. 즉, 클라이언트의 요청 명령이 서버로 보내지거나, 서버의 응답이 클라이언트로 전달될 때, 각각 해당하는 이벤트가 생성되어 전달된다. 이러한 이벤트의 생성 및 전달은 이벤트 매니저라 불리는 통신모듈의 해당 루틴을 호출함으로써 이루어진다[4].

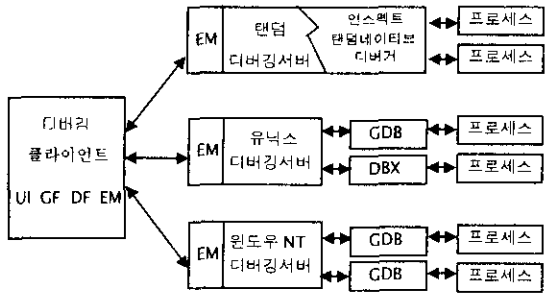
이벤트 매니저란 클라이언트와 서버간의 통신 메커니즘을 제공하는 라이브러리로서, 이벤트를 전달받기 위해 클라이언트와 서버는 자신들이 받을 이벤트들에 대해 이벤트 매니저의 해당 루틴들을 등록하고 이벤트가 도착되었을 때, 이벤트 매니저의 디스패처가 해당 루틴을 호출하게 된다.

이벤트 매니저는 ONC/RPC 와 XDR(eXternal Data Represenation)라이브러리를 사용하여 구현되었다. 일반적인 동기적 RPC 에서는 클라이언트가 서버로부터 응답을 받을 때까지 클라이언트의 입출력이 봉쇄되어 클라이언트는 다른 작업을 수행할 수 없게 된다. 유니뷰 시스템에서는 서버가 서비스를 처리하는 동안 클라이언트의 사용자 인터페이스가 봉쇄되지 않도록 하기 위하여 단방향 RPC 와 FRPC (Follow-up RPC)를 병행하여 사용하였다[6].

이벤트 매니저에서 사용하고 있는 프로토콜은 프로세서나 운영체제 등에 무관하게 TCP/IP 네트워크 상에서 동작하는 프로세스들간에 사용할 수 있도록 디자인되었으며, 플랫폼에 독립적으로 사용될 수 있도록 캡슐화되어 있어 특정 시스템 및 개방 시스템의 표준에 적용될 수 있도록 하였다.

5. 구현 상태

그림 2 는 현재까지 구현된 유니뷰 시스템의 구성을 보여준다.



[그림 2] 구현된 유니뷰 시스템의 구성도

디버깅클라이언트는 윈도우 NT 시스템상에서 비쥬얼 C++ 및 MFC 라이브러리를 이용하여 구현되었으며, 서버는 탠덤 기종과 유닉스 시스템 그리고 윈도우 NT 시스템 상에서 C++ 언어로 구현되었다.

서버의 후단부 디버거로 탠덤 기종에서는 네이티브 디

버거인 인스택트에 서버 기능을 확장하여 사용하였으며, 유닉스 및 NT 기종에서는 FSF(Free Software Fondation)의 gdb 를 사용하였다. 1 차 구현에서 gdb 를 후단부 디버거로 선정하게 된 이유는 gdb 의 소스 코드를 자유롭게 이용가능하다는 점과 gdb 자체가 높은 이식성을 가지고 있다는 점 때문이었다. 현재는 유닉스 서버의 후단부를 Sun 기종의 네이티브디버거인 dbx 로 확장하여 이식을 진행중에 있다.

구현된 유니뷰는 순차디버깅을 위한 고급의 인터페이스와 더불어 여러 개의 분산된 프로그램들을 한꺼번에 디버깅하면서 원격 호스트에의 연결과 관리, 디버깅되는 프로그램 간의 간편한 전환 기능 등을 제공하여 분산 멀티프로그램 디버깅을 위한 효과적인 인터페이스를 제공한다. 또한, 여러 프로그램들에 대한 동시제어 기능 및 분산 디버깅 작업의 동시 개시 기능, 통신 이벤트의 추적 기능 등 분산 디버깅을 효율적으로 수행하기 위한 여러 기능들을 제공한다. 이 중, 특히 통신 이벤트의 디버깅 기능은 분산 프로그램 간의 통신을 보여주고 해당 소스코드와의 연결을 통해 통신 과정을 사용자가 추적하면서 디버깅할 수 있게 해주어, 통신 과정에서의 에러 수정에 많은 도움을 준다[1].

6. 결론

최근 클라이언트-서버 구조의 분산시스템의 도입이 괄목하게 증가하고 있으나, 분산시스템의 개발에서 디버깅은 분산성과 비결정성으로 인하여 개발 기간 및 비용의 병목으로 남아있다. 본 연구에서는 클라이언트-서버 구조를 채택하여 분산디버거를 구현하고 클라이언트와 서버간의 인터페이스를 명확히 정의함으로써, 클라이언트의 단일화된 사용자인터페이스에 대하여 다양한 서버를 연동시킬 수 있게 되어 이기종 플랫폼간의 디버깅을 수행할 수 있었다. 이는 다른 분산디버거[5,7]에 비교하여 유니뷰시스템이 가지는 가장 큰 특징이다. 이러한 개방형 구조는 플랫폼이나 개수의 제한을 받지 않아 높은 수준의 시스템 확장성을 가진다. 앞으로 본 연구는 현재 RPC 에 기반하여 구현된 이벤트 추적 기능을 추가로 확장할 계획이며, JAVA 언어 등을 지원할 예정이다.

참고문헌

[1] 이은정 외 4 인, "분산처리 진단/교정 시스템의 이벤트 추적 기능의 설계 및 구현", 정보과학회 97 년 추계학술발표논문집, 제 24 권 2 호, pp.135-138, 1997.

[2] 정보통신부, "분산처리 진단/교정 소프트웨어 개발에 관한 연구", 1 차년도 연구개발 결과 보고서, 1997.

[3] 조영욱 외 4 인, "분산처리 디버거 유니뷰 서버의 구조," 정보과학회 97 년 추계학술발표논문집, 제 24 권 2 호, pp.327-330, 1997.

[4] 조영욱 외 4 인, "분산처리 진단/교정 시스템의 이벤트 매니저 개발," 정보처리학회 97 년 춘계학술발표논문집, 제 4 권 1 호, pp.724-728, 1997.

[5] D. Cheng, R. Hood, "A Portable Debugger for Parallel and Distributed Programs", Proceedings of Supercomputing'94, Washington, D.C., Nov.1994.

[6] J. Bloomer, Power Programming with RPC, O'Reilly & Associates, Inc., 1992.

[7] Suresh K. Damonaran-Kamal, Jeffrey S. Brown, "Towards Heterogeneous Distributed Debugging", Report No. LAUR-95-906, Los Alasmos Natinal Laboratory, 1995.