분산된 컴퓨팅 환경을 위한 자바 동기화

이상윤
대원과학대학 컴퓨터정보처리과
sylee@daewon.ac.kr

Java Synchronization for Distributed Computing Environment

Sangyun Lee
Dept. of Computer Information Processing, Daewon Science College

요 약

자바는 쓰레드 상호간의 동기를 프로그래밍 언어 자체의 기능으로 제공하고 있으므로 자바가 제공하는 동기화 메커니즘과 쓰레드는 병행처리를 수행하는 응용프로그램을 작성하는데 상당한 역할을 담당할 수 있다. 이에 따라, 병행처리와 관련된 자바의 기능을 분산된 컴퓨팅 환경에 적용하기 위한 많은 연구결과가 있다. 본 연구팀 에서는 자바 프로그램의 객체변환을 통하여 분산된 컴퓨팅 환경에서 동작하도록 지원하는 시스템을 발표한 바 있으나 분산처리와 관련된 기능을 지원하는 것으로 제한되었다. 병행처리를 수행하는 레거시 자바 프로그램을 분산된 컴퓨팅 환경에서 동작시키기 위하여 TORB(Transparent Object Request Broker)라고 명명된 이 시스템의 기능 확장을 시도하고 있으며 관련된 많은 문제들을 해결하였다. 본 논문에서는 서로 다른 컴퓨터에서 동시에 동작하는 자바 쓰레드간의 동기문제를 해결하기 위한 방안을 제시한다. 이는 단일 컴퓨팅 환경에서의 자바 동기화 메커니즘과 동일한 효과를 분산된 컴퓨팅 환경에서 얻을 수 있도록 지원하는 방안이며 TORB를 위하여 고안된 것이다.

1. 서론

자바의 쓰레드는 다중 처리 환경에서 하나의 프로그램 공간 내에서 독립적으로 동작하는 요소이므로 병행 프로그래밍을 위한 프로세스로 활용할 수 있다. 또한 자바는 쓰레드 상호간의 동기를 프로그래밍 언어 자체의 기능으로 제공하고 있으므로 자바가 제공하는 동기화 메커니즘과 쓰레드는 병행처리를 수행하는 응용 프로그램을 작성하는데 상당한 역할을 담당할 수 있다.[1]

가상기계에 기반을 둔 플랫폼 독립적인 실행환경, 객체지향 언어의 장점에 의한 프로그래밍 용이성, 범용성 있는 통신관련 API의 지원 등은 병행처리 프로그래밍과 관련된 자바의 장점을 분산된 컴퓨팅 환경으로 확장할 수 있는 가능성을 뒷받침해준다. 이에 따라 병행처리 응용프로그램의 작성에 적합한 자바의 장점을 분산된 컴퓨팅 환경에 적용하기 위한 많은 연구결과가 있다.[2,3,4]

이러한 연구의 일환으로, 본 연구팀 에서는 병행처리를 수행하는 레거시 자바 프로그램을 분산된 컴퓨팅 환경에서 수행할 수 있도록 지원하기 위하여 이미 발표한 분산처리 지원 시스템의 기능 확장을 시도하고 있으며 관련된 많은 문제들을 해결하였다. TORB(Transparent Object Request Broker)라고 명명된 이 시스템은 이미 작성된 자바 프로그램을 객체변환을 통하여 분산된 컴퓨팅 환경에서 동작하도록 지원하지만 분산처리와 관련된 기능을 제공하는 것으로 제한되었다.[5]

기능이 확장된 TORB는 서드파티 자바 패키지와 분산된 컴퓨팅 환경에서의 병행처리 운영을 위한 보조 프로그램들로 구성되며 범용의 가상기계에서 동작하므로 자바의 범용성을 분산된 컴퓨팅 환경에서도 그대로 유지할 수 있다. 또한, 독특한 객체변환 절차를 적용하여 프로그래밍 투명성의 범위를 확장 하였고, 통합된 대행객체를 통한 단일 객체변환 기법을 적용하여 실행코드의 배포를 자동으로 처리할 수 있도록 하였다. 이에 대한 효과로 개발자는 프로그램을 작성하는 동안 TORB를 활용하기 위한 부가적인 문법을 전혀 사용할 필요가 없고, 작성된 프로그램은 객체변환 절차를 거쳐서 TORB가 지원하는 보조 프로그램과 연동하여 분산된 컴퓨팅 환경에서 병행처리를 수행한다. 이때, 변환된 실행코드는 TORB가 제공하는 독자적인 클래스로더에 의해 자동으로 배포되어 동작한다.

본 논문에서는 서로 다른 컴퓨터에서 동시에 동작하는 자바 쓰레드간의 동기문제를 해결하기 위한 방안을 제시한다. 이는 단일 컴퓨팅 환경에서의 자바 동기화 메커니즘을 분산된 컴퓨팅 환경에서 동일한 효과로 유지시키기 위한 방안이며 병행처리와 관련된 TORB의 기능 확장을 위해 중요한 역할을 담당한다.

분산된 컴퓨팅 환경에서 자바의 동기화 기능을 유지하는 방안을 제시하기 위하여 본 논문은 다음과 같이 구성된다. 서론에 이어 2장 에서는 이미 작성된 자바 프로그램을 분산된 컴퓨팅 환경에서 동작시키기 위해 적용하는 변환작업과 TORB의 동작 메커니즘을 제시하고, 3장에서는 2장의 내용과 관련하여 자바 동기화 메커니즘을 분산된 컴퓨팅 환경에서 유지하기 위하여 적용된 방안을 설명하며 4장에서는 결론을 맺는다.

2. TORB(Transparent Object Request Broker)

2.1 분산된 컴퓨팅 환경에서의 병행처리

자바는 객체지향 프로그래밍 언어이므로 임의의 객체를 생성하고 접근(access)하는 방법으로 프로그램이 실행된다. 분산된 컴퓨팅 환경에서 생성된 객체가 다른 컴퓨터에 존재하는 것이라면 이 객체에 대한 접근조작은 네트워크를 통한 요청(request)과 응답(response)으로 처리되어야 한다. TORB에서는 네트워크를 통하여 전달되는 이러한 요청을 처리하고 응답하기 위해서 "TORB 서버"라는 서비스 프로그램을 도입하였다.

병행처리를 수행할 목적으로 이미 작성된 프로그램이 분산된 컴퓨팅 환경을 활용하기 위해서는 적절한 변환이 선행되어야 한다. TORB에서는 이미 작성된 자바 프로그램을 분산된 컴퓨팅 환경을 활용하는 버전으로 변환하는 후처리 프로그램을 제공한다. 이때 체계적인 변환기법과

분산된 컴퓨팅 환경에서의 자바객체에 대한 투명성을 지원하기 위해서 통합된 대행객체를 도입하였다.

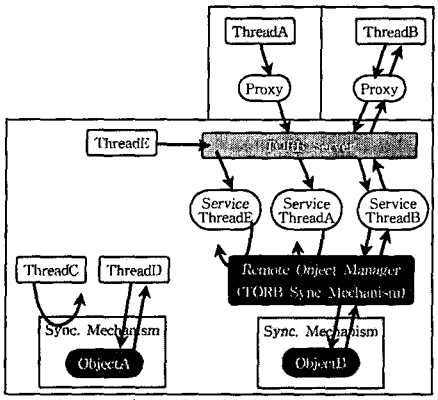

그림1 : 분산된 컴퓨팅 환경에서의 동기화

대행객체는 임의의 자바객체를 접근하는 일반적인 모든 동작을 대신하여 다른 컴퓨터에 존재하는 객체를 접근할 수 있도록 설계되었으며 후처리 프로그램에 의해 변환된 프로그램은 대행객체를 통하여 다른 컴퓨터에서 실행중인 "TORB 서버"에게 자바객체를 접근하기 위한 요청과 응답을 처리한다. 그림1을 참고하면 변환된 프로그램이 대행객체 및 "TORB 서버"와 상호작용을 통하여 분산된 컴퓨팅 환경에서 병행처리를 수행하는 메커니즘을 알 수 있다.

2.2 레거시 프로그램의 변환

TORB는 이미 작성된 자바 프로그램을 전혀 수정하지 않고 분산된 컴퓨팅 환경에서 수행하도록 변환하는 후처리 도구를 제공한다. "TORBC"라고 명명된 후처리도구는 자바를 위한 표준 실행환경에서 동작하며 단순한 형식의 설정정보에 의해 변환작업을 수행한다.

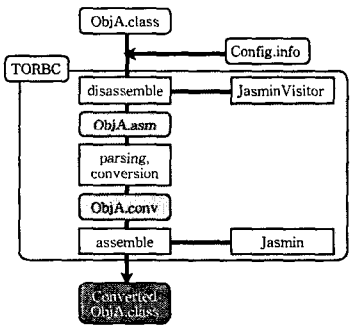

그림2 : TORBC에 의한 변환과정

"TORBC"에 의해 수행되는 변환작업은 BCEL의 Jasmin과 JasminVisitor을 활용하여 자바 어셈블리 수준에서 이루어지도록 구현하였으며, 변환하고자 하는 자바 클래스파일에 대한 자바 어셈블리 코드를 JasminVisitor에 의해 생성하고, 의미 분석과 변환과정을 통하여 수정된 자바 어셈블리 코드를 생성한 후, Jasmin에 의해 변환된 자바 클래스파일을 생성하는 절차로 구성된다.[6,7] 그림2는 "TORBC"가 이미 작성된 자바 프로그램을 분산된 컴퓨팅 환경을 활용하는 버전으로 변환하는 과정을 표현한 것이다.

그림2에서 "Config.info"는 변환의 대상이 되는 자바객체를 명시하기 위한 정보가 포함되며 원격 컴퓨터에서 생성되어야 하는 객체와 이들을 접근하는 다른 객체의 목록으로 구성된다.

3. 동기화 메커니즘

3.1 자바 동기화 메커니즘

단일 컴퓨팅 환경에서 자바 프로그램의 병행처리는 일반적으로 java.lang.Thread를 상속받은 객체들에 의해서 이루어진다. 이들은 하나의 프로그램 공간 내에서 독립적인 프로세스(쓰레드)로서 동시에 동작하며 병행처리를 수행한다. 동시에 동작하는 여러 개의 쓰레드가 동일한 객체에 대한 접근을 시도할 때 동기화 메커니즘이 필수적으로 요구된다. 키워드 "synchronized"는 자바에서 동기화 메커니즘을 지원하기 위해 도입된 것이며 그림3에는 자바 쓰레드의 동기화를 위해 두 가지 방법으로 사용되는 예를 보였다.[1]

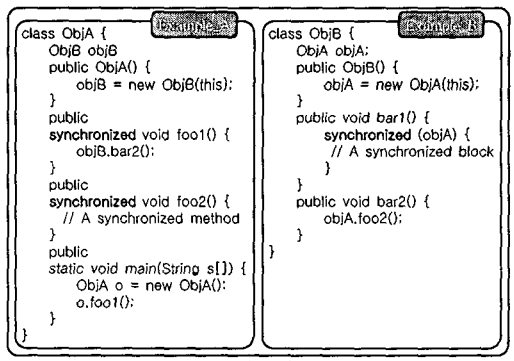

그림3 : 자바 동기화의 두 가지 방법

그림3에서 "Example_A"는 동기화 메쏘드를 적용하는 경우이고, "Example_B"는 동기화 블록을 적용하는 경우로서 임의의 쓰레드가 이 메쏘드 혹은 블록(임계영역)을 수행하는 동안 "ObjA"의 동일한 인스턴스를 접근하고자 하는 다른 쓰레드 들은 대기한다.

3.2 TORB를 위한 동기화 방안

TORB가 제공하는 컴퓨팅 환경에서는 하나의 프로그램 수행공간에서 객체들이 서로 다른 컴퓨터에 존재할 수 있으므로 자바의 동기화 메커니즘을 유지하기 위해서 적절한 조치가 필요하며 이는 두 가지로 정리할 수 있다. 하나는 서로 다른 컴퓨터에 존재하는 특정 객체에 대한 배타적인 접근권한을 적절히 관리하는 방법을 마련

하는 것이고, 다른 하나는 이러한 관리방법을 TORBC에 의해 변환된 프로그램에 적용할 수 있도록 변환하는 절차를 마련하는 것이다.

3.2.1 원격 객체 관리자

"Remote Object Manager"는 서로 다른 컴퓨터에 존재하는 객체에 대한 배타적인 접근권한을 적절히 관리하기 위하여 도입한 것이며 독자적인 동기화 메커니즘을 적용하여 동작한다. 그림1에는 원격 컴퓨터에 존재하는 "ThreadB"가 "ObjectB"에 대한 배타적인 접근권한을 얻었을 때, 다른 원격 컴퓨터에 존재하는 "ThreadA"와 로컬 컴퓨터의 "ThreadE"에 대한 접근제어를 "Remote Object Manager"가 처리하는 모습이 표현되어 있다. 특히, "ObjectB"와 같이 원격 컴퓨터에 존재할 가능성이 있는 객체에 대한 배타적인 접근제어를 "TORB 서버"를 통하여 처리되도록 하였으므로 로컬 컴퓨터에서 실행중인 "ThreadE"에 대해서도 동등한 방법으로 관리된다.

3.2.2 배타적 접근권한의 재획득

자바의 동기화 메커니즘에서는 하나의 쓰레드가 동일한 객체에 대한 배타적인 접근권한을 두 번 이상 획득하는 것을 허용한다. 이는 하나의 쓰레드가 자신이 이미 획득한 배타적인 접근권한 때문에 동일한 객체에 대한 새로운 배타적 접근권한을 얻을 수 없어서 생기는 교착상태를 피하기 위한 것으로서 그림3에 이러한 경우가 표현되어 있다.

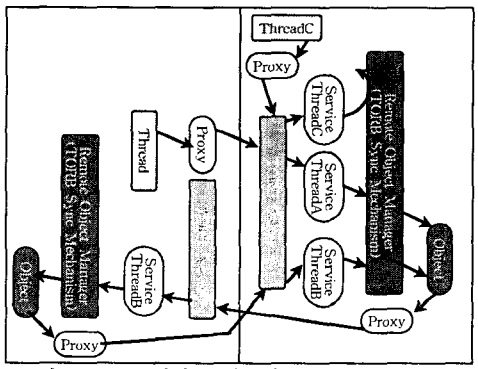

그림4 : TORB에서의 배타적 접근권한의 재획득

TORB에서는 서로 다른 컴퓨터에 존재할 수 있는 것으로 지정된 객체에 대한 배타적인 접근권한은 모두 "Remote Object Manager"에 처리되므로 동일한 쓰레드에서 순차적으로 발생한 접근조작인 것이 확인된 다면 그림3의 경우를 수용할 수 있다.[8] 특정 쓰레드의 식별정보는 "Thread.currentThread().hashCode()"를 이용해 얻을 수 있고, 하나의 메쏘드 호출이 종료될 때 까지 이 정보를 유지하는 것은 어렵지 않다. 그림4에는 "Remote Object Manager"가 배타적인 접근권한을 할당받은 쓰레드의 식별정보를 이용하여 동일한 객체에 대한 접근조작을 허용하는 경우와 그렇지 않은 경우를 표현하였다.

3.2.3 변환방안

그림1과 그림4에 표현된 바와 같이 다른 컴퓨터에 존재할 가능성이 있는 객체에 대한 배타적인 접근권한은 "Remote Object Manager"에 의해서만 처리되고, 자바의 자체적인 동기화 메커니즘은 "Remote Object Manager"의 기능을 위해 일부 사용된다. 이를 위해서 자바의 동기화 메커니즘이 적용된 코드를 "Remote Object Manager"가 처리하는 코드로 변환하는 절차가 필요하다. TORBC는 자바 어셈블리 수준의 변환처리를 수행하므로 이러한 변환을 수용할 수 있다.

4. 결론

자바의 쓰레드와 동기화 메커니즘은 병행처리를 수행하는 프로그램을 작성하는데 상당한 역할을 담당하므로 이러한 장점을 분산된 컴퓨팅 환경에 적용하기 위한 많은 연구결과가 있다. 이러한 연구의 일환으로, 본 논문에서는 이미 발표한 분산처리 지원 시스템의 기능 확장을 통해 자바 쓰레드를 이용한 병행처리와 이들 간의 동기화를 분산된 컴퓨팅 환경에서 투명하게 유지하는 방안을 제시하였다. 이 방안은 서로 다른 컴퓨터에 존재할 가능성이 있는 객체에 대한 배타적인 접근권한을 관리하는 "Remote Object Manager"를 도입하고, 이들에 대한 접근조작이 "TORB 서버"를 통해서만 이루어지도록 후처리를 적용하여 실현될 수 있었다.

참고문헌

[1] Doug Lea, "Concurrent Programming in Java - Design Principles and Patterns", Addison-Wesley, Reading, Mass., 1996

[2] Weimin Yu and Alan Cox, "Java/DSM: A Platform for Heterogeneous Computing", In Concurrency: Practice & Experience, Vol. 9, No. 11, pp. 1213-1224, 1997

[3] Yariv Aridor, Michael Factor and Avi Teperman, "cJVM: a Single System Image of a JVM on a Cluster", In Proceedings of ICPP 99, IEEE, 1999

[4] Philippsen M, Zenger M. "JavaParty - Transparent remote objects in Java", Concurrency: Practice and Experience, 9(11):1225--1242, 1997

[5] 이상윤, 김승호, "프로그래밍 투명성을 지원하는 분산 프로그래밍 도구의 설계", 한국정보과학회 논문집 제 31권 3호, pp. 259-268, June 2004

[6] M. Dahm. "Byte code engineering with the BCEL API". Technical Report B-17-98, Freie Universitat Berlin, Institut fur Informatik, April 2001

[7] J. Meyer, T. Downing. "Java Virtual Machine". O Reilly, 1997

[8] Bernhard Haumacher, Thomas Moschny, Jurgen Reuter and Walter F. Tichy, "Transparent Distributed Threads for Java", International Parallel and Distributed Processing Symposium, 2003