활용 절차의 투명성을 제공하는 분산 처리 지원 도구

이상윤
대원과학대학
sylee@daewon.ac.kr

KISS Korea Computer Congress 2005

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

요 약

여러 대의 컴퓨터가 협조해야만 달성할 수 있는 응용을 위해 분산 처리 모델은 성공적인 해결책을 제공한다. 이는 분산 처리 모델이 여러 대의 컴퓨터를 통합하여 운영하는 체계적인 절차와 서비스를 제공하기 때문이다. 본 연구팀에서는 여러 대의 컴퓨터를 통합하여 활용하기 위하여 이미 익숙한 컴퓨팅 환경(단일 컴퓨팅 환경)을 위한 절차와 서비스를 적용하는 방안을 연구하고 있으며 이미 발표한 TORB(Transparent Object Request Broker)는 프로그래밍 투명성의 제공을 통하여 분산된 컴퓨팅 환경을 활용하기 위한 프로그램을 작성하는 것에 대한 투명한 서비스를 지원한다. 단일 컴퓨팅 환경에서는, 작성된 응용 프로그램을 기동하는 것이 무시하여도 좋을 만큼 간단한 절차이다. 그러나 분산된 컴퓨팅 환경에서 이를 간단한 절차로 수용하는 것은 쉬운 일이 아니며 기존의 분산 처리 모델에서는 체계적인 지원을 고려하고 있지 않다. 본 논문에서는 여러 대의 컴퓨터를 통합하여 활용하기 위하여 작성되어 분산 처리를 수행하는 응용프로그램을 단일 컴퓨팅 환경에서와 동일하게 취급하는 투명성을 제공하기 위한 방안과 이 기능의 수용에 대한 효과를 제시한다.

1. 서 론

여러 대의 컴퓨터가 협조해야만 달성할 수 있는 응용을 위해 분산 처리 모델은 성공적인 해결책을 제공한다. 이는 분산 처리 모델이 여러 대의 컴퓨터를 통합하여 운영하는 체계적인 절차와 서비스를 제공하기 때문이다.[1] 객체로 표현되는 언어 요소가 생성 및 동작되는 장소에 대한 개념은 자바와 같은 객체 지향 언어로 작성된 프로그램을 실행하기 위하여 단일 컴퓨팅 환경과 분산된 컴퓨팅 환경이 제공하는 패러다임의 대표적인 차이점으로 평가할 수 있다. 객체의 생성 및 동작하는 장소의 개념은 분산된 컴퓨팅 환경이 제공하는 패러다임에 추가된 것이며 새로운 형태의 응용을 가능하게 한다. 분산된 컴퓨팅 환경의 응용에서, 객체에 대한 장소의 개념은 명시적인 방법이나 암시적인 방법으로 모두 활용될 수 있으며 각각의 경우는 다음과 같은 의미를 가진다. 명시적인 활용은 본질적으로 분산된 컴퓨팅 환경을 구성하는 여러 대의 컴퓨터가 참여해야만 구현될 수 있는 응용을 위한 것이고, 암시적인 활용은 단일 컴퓨팅 환경을 위하여 이미 존재하는 프로그램이 분산된 컴퓨팅 환경을 통하여 성능 향상을 얻는 것이다. 한편, 객체의 위치를 명시하지 않고 분산된 컴퓨팅 환경을 활용하는 패러다임은 기존의 분산 처리 모델에서는 고려하지 않은 개념이다.

그림1은 객체에 대한 장소의 개념을 활용하는 두 가지 패러다임을 모두 지원하기 위하여 설계된 분산 처리 모델을 표현한 것이며 프로그래밍 투명성을 통하여 이를 실현하고 있다. 그림1에서 아래 방향으로 진행하는 두 개의 화살표는 새로운 분산 처리 모델을 활용하는 두 가지 방법을 표현하고 있으며 각각의 의미는 다음과 같다. 먼저, 왼쪽의 화살표는 객체에 대한 장소의 개념을 암시적으로 활용하는 것을 표현한 것으로서, 객체 지향 언어로 작성된 응용프로그램을 전혀 수정하지 않고 프로그래밍 투명성의 지원에 의한 변환 절차가 적용된 후 분산된 컴퓨팅 환경에서 수행됨을 의미한다. 다음, 오른쪽의 화살표는 객체에 대한 장소의 개념을 명시적으로 활용하는 것을 표현한 것으로서, 객체 지향 언어로 응용프로그램을 작성할 때, 객체의 위치가 실행 시간에 결정되는 경우에 예약된 메서드를 호출(단일 컴퓨팅 환경을 위한 프로그래밍 기법임)하여 프로그램을 작성하며, 작성된 프로그램은 객체에 대한 장소가 고정되는 경우에 대한 변환 절차가 추가로 적용된 후 분산된 컴퓨팅 환경에서 수행됨을 의미한다.

본 연구팀에서는 여러 대의 컴퓨터를 통합하여 활용하기 위하여 이미 익숙한 컴퓨팅 환경(단일 컴퓨팅 환경)을 위한 절차와 서비스를 적용하는 방안을 연구하고 있으며 이미 발표한 TORB(Transparent Object Request Broker)는 프로그래밍 투명성의 제공을 통하여 분산된 컴퓨팅 환경을 활용하기 위한 프로그램을 작성하는 것에 대한 투명한 서비스를 지원한다.[2]

단일 컴퓨팅 환경에서는, 작성된 응용 프로그램을 기동하는 것이 무시하여도 좋을 만큼 간단한 절차이다. 그러나 분산된 컴퓨팅 환경에서 이를 간단한 절차로 수용하는 것은 쉬운 일이 아니며 기존의 분산 처리 모델에서는 체계적인 지원을 고려하고 있지 않다. 그림1에 표현된 분산 처리 모델은 분산된 컴퓨팅 환경을 활용하기 위한 프로그램을 작성하는 동안 적용되는 프로그래밍 투명성뿐만 아니라 작성이 완료된 프로그램을 기동하는 절차에 대한 투명성을 동시에 고려하고 있다.

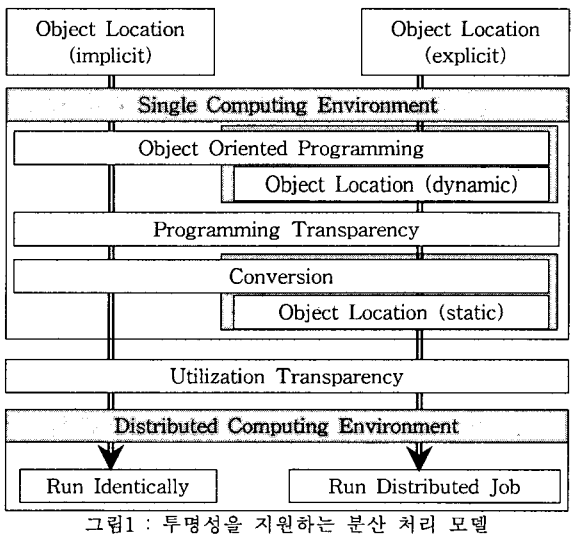

그림1 : 투명성을 지원하는 분산 처리 모델

본 논문에서는 여러 대의 컴퓨터를 통합하여 활용하기 위하여 작성되어 분산 처리를 수행하는 응용프로그램을 단일 컴퓨팅 환경에서와 동일하게 취급하는 투명성을 제공하기 위한 방안과 이 기능의 수용에 대한 효과를 제시한다.

2. 실행 코드 배포의 투명성

2.1 분산 처리를 위한 실행 코드의 배포

어떠한 분산 처리 모델을 적용하더라도 분산된 컴퓨팅 환경을 활용하는 프로그램의 실행코드는 수행되기 전에 각각의 컴퓨터로 배포되어야 한다. 프로그램의 실행 코드를 분산 처리에 참여하는 컴퓨터에 배포하고 활용하는 문제를 해결하기 위한 방안은 그림2과 같이 세 가지를 고려해 볼 수 있다.

A) No Support	Using existing transmission method(FTP. HTTP) e.g., HORB
B) By artificial defined service	Using naming and trade service after register to implementation repository e.g. CORBA
C) By dynamic distribution	Using specially contrived java ClassLoader e.g., TORB

그림2 : 실행 코드 배포를 위한 방안

- ◆ No Support : 실행 코드의 배포 및 활용과 관련된 기능을 제공하지 않는 경우이며 프로그램을 기동하기 전에 별도의 전송 기법(FTP, HTTP 등)을 활용하여 배포하는 것을 전제로 하는 방안이다.
- ◆ By artificial defined service : CORBA 권고안에 명시된 객체 서비스에 포함되는 방안이며, 실행 코드를 구현 저장소(Implementation Repository)에 등록한 후 프로그램을 기동하여야 하며 네이밍서비스와 트레이드서비스를 통하여 운영된다.[3]
- ◆ By dynamic distribution : 자바가 지원하는 클래스로더 기능을 적용하여 실행 코드의 동적인 배포를 처리하는 방안이다. 자바는 클래스로더를 통하여 디스크에 저장된 실행 코드(클래스파일)의 적재를 동적으로 처리하고, 필요에 따라 사용자가 정의한 클래스로더를 적용하도록 허용하므로 네트워크를 통한 실행 코드의 배포를 처리하기에 최적의 환경을 제공한다.

객체에 대한 장소의 개념을 명시적으로 활용하여 본질적으로 여러 대의 컴퓨터가 상호 협조하는 응용을 위한 분산 처리의 경우에는 실행 코드의 배포 대상을 명시할 수 있으므로 그림2의 A, B와 같이 프로그램 기동 전에 실행 코드를 배포하는 방안이 타당하다. 그러나 객체에 대한 장소의 개념을 암시적으로 활용하는 응용인 경우에는 프로그램이 기동된 이후에 실행 코드의 배포 대상이 결정되므로 그림2의 C와 같이 요청되는 실행 코드를 동적인 방법으로 배포하는 방안을 적용하여야 한다. 또한, 그림2의 C는 A와 B를 대체 할 수 있으며 "TORB"와 같이 자바로 구성되는 단일 프로그래밍 언어를 위한 분산 처리 모델의 경우에는 클래스 로더를 이용하여 실현 가능하다.

2.2 클래스 로더

자바 응용 프로그램을 구성하는 실행 코드(클래스 파일)는 기동된 이후에 자바 가상 기계에 의하여 운영되는 클래스 로더에 의해 요구되는 상황에 따라 메모리에 적재된다.[4] 사용자 정의 클래스 로더를 허용하는 자바의 이러한 메커니즘을 적용하면, 프로그램을 기동한 후에 요구되는 상황에 따라 다른 컴퓨터에 존재하는 실행 코드를 네트워크를 통하여 전송받아 메모리에 적재하도록 구성할 수 있다.

2.3 TORB를 위한 클래스 로더

그림1에 표현된 분산 처리 모델은 분산된 컴퓨팅 환경에서 객체에 대한 장소의 개념을 활용(명시적, 암시적)하는 두 가지 패러다임을 모두 고려하고 있으며 "TORB"는 이를 실현하는 분산 처리 지원 도구이다.

객체의 위치를 명시하지 않고 분산된 컴퓨팅 환경을 활용하는 패러다임을 지원하기 위해서는 실행 코드의 동적인 배포가 암시적으로 이루어져야 하며 사용자 정의 클래스 로더를 통한 동적인 배포를 통하여 실현하였다. 이에 따라, 실행 코드가 존재하지 않는 원격 컴퓨터에게도 객체 서비스를 요청할 수 있으며 원격의 컴퓨터에서는 "TORB 서버"가 실행 중 이므로 사용자 정의 클래스 로더를 통하여 상황별로 필요한 실행 코드를 동적으로 전송받아서 수행한다. 이때, 실행 코드가 존재하는 컴퓨터를 명시할 수 없으나 프로그램을 최초로 기동한 컴퓨터에 실행 코드가 존재하는 사실을 이용하면 적절한 코딩 기술을 통하여 쉽게 찾을 수 있다.

사용자 정의 클래스 로더와 관련된 자바의 메커니즘을 수용하여 자바 프로그램의 실행 코드를 상황별로 동적으로 배포하는 것이 가능해짐에 따라 객체에 대한 장소의 개념이 명시적이든 암시적이든 분산된 컴퓨팅 환경을 활용할 목적으로 작성된 프로그램은 적절한 변환이 적용된 이후에 단일 컴퓨팅 환경과 동일한 절차를 따라 기동할 수 있으며 그림2의 A, B와 같은 절차를 대체할 수 있다.

3. TORB를 이용하는 병렬 처리

자바가 제공하는 쓰레드와 동기화 메커니즘을 이용하면 병렬 처리를 수행하는 응용 프로그램을 쉽게 작성할 수 있으며 "TORB"에 의한 분산 처리를 통하여 수행시간 단축의 효과를 얻을 수 있다. 이는 객체의 위치를 명시하지 않고 분산된 컴퓨팅 환경에 참여하는 여러 대의 컴퓨터를 동시에 활용하는 대표적인 응용이며 그림1에 표현된 활용의 투명성을 제공하기 위해서는 참여하는 컴퓨터의 선정이 암시적인 방법으로 이루어져야 한다. 또한, 복잡한 작업을 여러 개의 부분 작업으로 분할하여 분산되어 있는 컴퓨터를 통하여 동시에 수행한 후 이들을 통합하는 것은 병렬 처리에 대한 장단점(프로세서의 투입의 편리성과 통신의 오버헤드)이 극대화 되는 것이며 응용 형태에 따라서 효과적인 성능 향상을 제공할 수 있다.[5]

3.1 원격 컴퓨터의 목록

개발자가 작성한 프로그램이 수행되는 동안 프로그램 요소(객체)가 분산된 컴퓨팅 환경을 구성하는 서로 다른 컴퓨터를 통하여 수행되도록 지원하고 이들을 활용하는 암시적인 체제가 필요하다. 그림3에 표현된 "TORB 서버 목록 관리"의 개념은 가용한 "TORB 서버"들의 목록을 관리하고 분산 처리에 참여시키는 암시적인 체제를 제공하기 위한 것이다. 그림3에는 객체의 위치를 고려하지 않고 작성된 응용 프로그램(주로 병렬 처리를 수행하는)이 분산된 컴퓨팅 환경에 참여하는 서로 다른 여러 대의 컴퓨터를 암시적으로 활용하는 절차를 표현하였으며 다음은 이에 대한 해설이다.

- ◆ 그림3의 "TORB 서버"에 표현된 두 개의 상자는 네트워크에 존재하는 다른 "TORB 서버"의 URL정보를 보관하는 용도로 도입된 것이며 상단의 상자는 "TORB 서버"의 목록을 관리하는 서비스 프로그램의 URL이 보관되고, 하단의 상자에는 최소한 자신을 포함해서 네트워크에서 활동 중인 "TORB 서버"의 URL목록이 보관된다.
- ◆ 각각의 "TORB 서버"는 "Req_URL"라는 요청을 적절히 처리하여 활동 중인 "TORB 서버"의 URL을 "Rep_URL"을 통하여 응답해야 하며 먼저 상단의 상자에 보관된 정보를 이용하여 서비스 프로그램에 요청을 위임해 본 후 성공하면 그 결과를 응답하고 실패하면 하단의 상자에 보관된 URL중 하나를 응답한다.
- ◆ "TORB 서버"는 명시적인 방법과 암시적인 방법으로 기동되며 명시적인 방법으로 기동할 때, "TORB 서버"의 목록을 관리하는 서비스 프로그램의 URL을 지정할 수 있다. 목록 관리를 위한 서비스 프로그램의 URL이 지정되지 않으

먼 "TORB://LOCALHOST:기본포트"의 기본 URL이 지정된다.

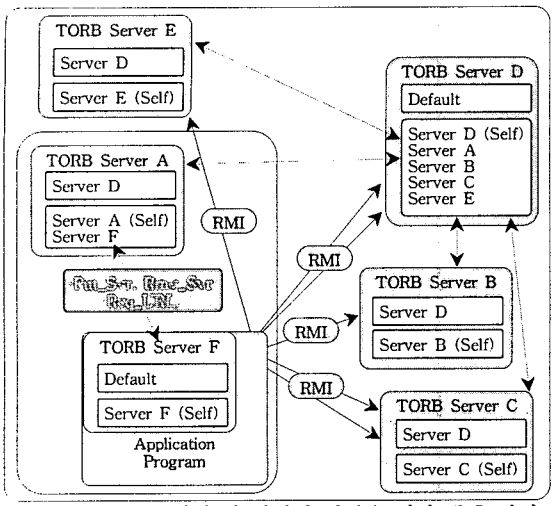

그림3 : "TORB 서버"의 암시적 할당을 위한 목록 관리

◆ "TORB 서버"의 목록은 이를 관리하는 서비스 프로그램에 네트워크를 통하여 전달되어 오는 메시지를 통하여 관리된다. 이 목록에 등록하기 위한 메시지 "Put_Svr"는 해당 서비스 프로그램의 URL을 지정하여 다른 "TORB 서버"를 명시적으로 기동할 때 발생되고 삭제하는 메시지 "Rmv_Svr"는 임의의 "TORB 서버"를 선정하는 과정에서 응답하지 않는 "TORB 서버"가 발견 될 때 발생된다. 한편, 그림3에서 점선으로 표현된 화살표는 모두 "Put_Svr", "Rmv_Svr", "Req_URL", "Rep_URL"의 네 가지 메시지를 주고받는 것을 의미한다.

◆ "TORBC"에 의해 적절히 변환된 응용 프로그램이 기동할 때, 그림3에 표현된 것과 같이 동일한 실행 공간(컨텍스트)에 "TORB 서버"가 기동 되며 이는 다른 컴퓨터에서 이 컴퓨터에 존재하는 객체에 접근하는 서비스를 담당한다. 이때의 "TORB 서버"는 암시적으로 기동 되므로 "TORB 서버"의 목록을 관리하는 서비스 프로그램의 URL이 기본 값(TORB://LOCALHOST:기본포트)이 된다. 응용 프로그램에서 원격 객체를 생성할 때 마다 기본 값으로 설정된 URL을 이용하여 "Req_URL"를 통한 요청과 "Rep_URL"을 통한 응답이 이루어지며 이를 위하여 같은 컴퓨터에 별도의 "TORB 서버"를 미리 실행하여 독립적인 객체 서비스와 함께 목록을 관리하는 서비스를 수행한다. 또한 별도의 "TORB 서버"가 기동 되어 있지 않은 경우에는 자기 자신에게 요청과 응답을 처리하므로 일관성이 유지된다.

이상의 해설을 종합하여 그림3에 표현된 전반적인 동작을 다음과 같이 설명할 수 있다. "Server D"는 최초에 명시적으로 기동하였고 "TORB 서버"의 목록을 관리하는 서비스 프로그램의 URL을 명시하지 않았으므로 상단의 상자에 기본 값이 저장되어 있다. 다음에 "Server A", "Server B", "Server C", "Server E"가 순차적으로 기동하였고 이들은 모두 "Server D"를 "TORB 서버"의 목록을 관리하는 서비스 프로그램으로 취급하여 지정하였으므로 "Server D"의 하단 상자에 이들의 목록이 같은 순서로 등록되어있다. "Server A"가 기동 되어 있는 동일한 컴퓨터에서 "TORBC"에 의해 변환된 응용 프로그램이 기동되었고 이때 암시적으로 기동된 "Server F"가 "Server A"의 하단상자에 등록된다. 이후 응용 프로그램에서 원격 객체의 생성을 시도할 때 마다 메시지 "Req_URL"이 발생하고 이 메시지는 "Server D"에게 위임되어 "Server A", "Server B", "Server C", "Server D", "Server E" 중 하나가 선택되어 메시지 "Rep_URL"을 통하여 응답 된 후 적절한 객체 서비스가 이루어진다.

한편, "Server D"가 기동되어 있는 동일한 컴퓨터에서 응용 프로그램을 기동하였을 경우에도 다음과 같이 일관성이 유지된다. 응용 프로그램과 동일한 실행공간에 기동된 "TORB 서버"를 위한 메시지 "Req_URL"은 동일한 컴퓨터에 실행 중인 "Server D"에게 전달된 후 "Server D"의 상단상자에 보관된 URL(즉, 자기자신)에게 위임하여 "Server A", "Server B", "Server C", "Server D", "Server E" 중 하나가 선택되어 메시지 "Rep_URL"을 통하여 응답 된 후 적절한 객체 서비스가 이루어진다.

3.2 관련 연구

분산된 컴퓨팅 환경을 통하여 병렬처리의 효과를 얻기 위한 대표적인 시도로 "javaParty"가 있으나 본 논문에서 제시된 결과와 같이 실행코드의 동적인 배포와 활용 절차의 투명성을 고려하지 않고 있다. 즉, "javaParty"를 활용하기 위해서는 제공되는 API를 활용하는 방법을 학습하여 프로그램 작성시 적용하여야 하며, 매뉴얼에 지시에 따라 서버를 기동하고, 완성된 프로그램의 실행코드는 분산처리에 참여하는 모든 컴퓨터에 별도의 방법으로 배포한 후 프로그램을 기동할 수 있다.[6]

4. 결론

본 연구팀에서 이미 발표한 TORB는 프로그래밍 투명성의 제공을 통하여 분산된 컴퓨팅 환경을 활용하기 위한 프로그램을 작성하는 것에 대한 투명한 서비스를 지원한다. 또한, TORB가 제공하는 프로그래밍 투명성과 활용 절차의 투명성을 체계적으로 정의하기 위하여 그림1과 같은 분산 처리 모델을 설계하였다.

본 논문에서는 그림1과 같은 분산 처리 모델을 실현하기 위하여 프로그래밍 투명성과 더불어 활용 절차의 투명성을 추가로 제공하기 위한 방안을 제시하였다. 사용자 정의 클래스 로더와 관련된 자바의 메커니즘을 수용하여 실행코드의 배포절차를 투명하게 하였고, 분산된 컴퓨팅 환경에서 객체의 위치개념을 암시적으로 활용하는 응용을 투명하게 처리하기 위한 메커니즘을 제시하였다.

활용 절차의 투명성을 제공하기 위한 이러한 기능을 수용하게 됨에 따라 "TORB 서버"를 기동하는 것 외에는 분산된 컴퓨팅 환경을 활용하기 위하여 별도의 절차를 학습할 필요가 없으며 단일 컴퓨팅 환경을 활용하는 절차와 동등한 방법을 적용한다.

참고문헌

[1] S. Maffeis, "Run-Time Support for Object-Oriented Distributed Programming", PhD thesis, University of Zurich, 1995.

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

[3] J. Siegel, "CORBA Fundamentals and Programming", John Wiley & Sons, 1996

[4] Sun Microsystems, "Understanding Extension Class Loading" On-Line Document, http://java.sun.com/docs/books/tutorial/ext/basics/load.html

[5] Inria Sophia Antipolis, "C3D - A distributed raytracer for benchmarking Java RMI & Serialization", On-Line Document, http://www-sop.inria.fr/sloop/javall/apps/c3d.html

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