Java RMI기반의 3-tier 클라이언트/서버 JDBC 드라이버의 설계

박정인 °, 전순미
인제대학교 전산학과

The Design of a 3-tier client/server JDBC Driver Based on Java RMI

Jeong-In Park °, Soon Mi Jun
Department of Computer Science, Inje University

요 약

본 논문에서는 클라이언트측에 별도의 JDBC드라이버 없이 JDBC API만 가지고도 여러 곳에 분산된 DB서버로 접근이 가능하게 하기 위해 RMI(Remote Method Invocation)를 이용하여 JDBC 3-tier모델을 위한 미들웨어를 제안하였다. 즉, JDBC를 이용한 Applet은 Download된 원래의 서버와 DB서버가 같은 경우에만 그 곳에 접속하여 DB에 액세스 가능한 현재의 제약을 Java RMI를 이용한 미들웨어를 구축하여 Client--middle ware--DB Server라는 3-tier 시스템으로 해결하는 방법을 제시하였다. 이 미들웨어는 서로 다른 데이터베이스 관리 시스템과의 연결을 위해 기존의 JDBC 드라이버를 이용하는 구조를 가질 수 있도록 설계하였다.

1. 서 론

Java로 클라이언트/서버 분산 컴퓨팅을 구현함에 있어서 클라이언트측의 applet이나 application은 원격지의 메소드를 호출할 어떠한 메커니즘도 갖고 있지 않다.[2] applet은 모든 작업을 local에서 수행하거나 아니면 원격지로부터 추가적인 applet을 물리적으로 가져와야 한다. 이러한 문제를 해결하기 위해 RMI가 이용된다. RMI는 직접적으로 분산객체 모델을 Java언어로 통합할 수 있다.

분산된 클라이언트 시스템과 데이터베이스 관리 시스템들 사이에서 응용 시스템의 구축을 위해 Java언어를 이용함으로써 효율적으로 구현할 수 있다. 그리고 구현 과정에서 이미 표준화된 데이터베이스 연결 프로토콜인 JDBC의 이용은 필수적이다. 하지만 이러한 Java언어의 많은 장점 이면에 몇 가지 제한 사항으로 인해 시스템 구축의 효율을 더욱 높일 수 있는 상황이 제한되고 있다. 그러한 제한 가운데 하나인 Java applet의 보안상 접근 가능한 시스템의 제한은 가장 큰 문제점으로 드러나고 있다.[2]

본 논문에서는 이러한 Java애플릿의 한계성을 극복하여 데이터베이스로의 접근을 용이하게 하고 Java애플리케이션과 애플릿에서 동일하게 적용 가능한 방법을 제시하여 설계한다. 이러한 접근을 위해 Java의 데이터베이스 연결을 위한 표준 API인 JDBC를 그대로 유지하고 이를 활용하되 JDBC의 하위 프로토콜을 RMI을 이용하여 설계한다.

이상의 연구에 대한 본 논문의 구성은 다음과 같다.

2장에서는 Java RMI에 대한 간략한 관련 연구를 기술한다.

3장에서는 Java와 데이터베이스 관리 시스템과의 표준연결을 지원하는 JDBC표준 API에 대하여 살펴보고 이를 분산 객체 환경 상에서 구현 가능한 방안을 알아보고, 4, 5장에서는 실제로 소개한 RMI JDBC 드라이버의 구성 및 설계에 대하여 기술한다

끝으로 결론에서는 문제점 및 향후 연구과제를 알아보고 글을 맺는다.

2. RMI

RMI객체는 원격 자바객체로서 그것의 메소드들은 다른 자바가상머신이나 네트워크로부터 호출될 수 있다. 그림 1은 RMI의 동작 메카니즘을 보여주고 있다.[2][5]

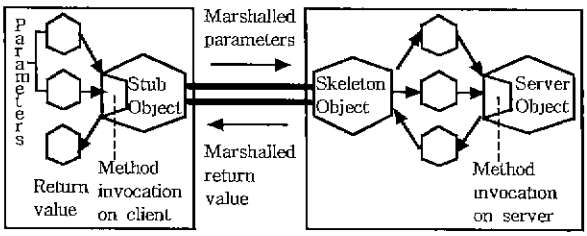

그림 1 RMI의 Stub와 Skeleton객체들

2.1 RMI의 구조

2.1.1 Stub

클라이언트 코드가 원격 객체상의 원격 메소드를 호출하길 원할 때 실제적으로 Stub라 불리는 객체내에 캡슐화 되어 있는 일반 자바 메소드를 호출한다. Stub는 클라이언트상에 존재하며, 원격 메소드에서 사용되는 전달인자를 취하고 이 전달인자를 바이트들의 블록으로 포장한다. 클라이언트 상에서 Stub메소드는 다음으로 구성되어 있

는 정보블럭을 형성한다.

- 사용되어질 원격객체의 식별자
- 호출될 메소드를 설명하는 메소드 번호
- 마샬링된 전달인자

2.1.2 Skeleton

서버측에서 패킷내에 포함되어 있는 정보를 인지하고 이 정보를 원격 메소드를 실행하는 실제 객체에게 전달하는 객체를 Skeleton객체라 한다. 특히, Skeleton은 모든 원격 메소드 호출에 대해 다섯 가지 기능을 수행한다.

- 전달인자를 언마샬(unmarshall)
- 서버상에 존재하는 실제 객체상에서 원하는 메소드 호출
- 서버상에서 호출의 결과나 예외상황을 감지
- 해당값을 marshalling
- marshalling된 형태의 값으로 구성되어 있는 패키지를 클라이언트상의 Stub에게 돌려 보냄

2.2 RMI의 동작 메카니즘

RMI를 이용하면 로컬의 자바객체 상에서 원격지의 RMI객체에 있는 메소드를 호출할 수 있다. 이런 측면에서 보면 RMI는 RPC(Remote Procedure Call)와 매우 유사하다. RMI는 매개변수로 또는 결과값으로 원격 객체의 참조에 의한 전달이 가능하다. 클라이언트가 서버를 호출할 때 RMI시스템은 여러 계층에 걸쳐 일을 수행한다. Stub는 다른 계층과 통신할 수 있도록 작성된 Java코드이며 Java RMI시스템은 자동적으로 몇 가지 도움을 주는 function을 사용할 수 있다. RMI클래스로부터 상속된 클래스는 Stub나 Skeleton을 구현한다. Stub는 클라이언트 코드를 위해, Skeleton은 서버를 위해 사용되며 Stub와 Skeleton계층에서의 작업(호출 또는 서비스 등)을 끝마치면 다른 두 계층(remote reference layer와 transport layer)으로 전달된다. 결과적으로 RMI Stub는 원격 객체에 대하여 클라이언트상의 proxy로 일하며 RMI는 이러한 proxy들이 잘 수행될 수 있도록 추가적인 security를 제공한다. RMI는 method invocation내의 파라메터로서 이러한 Stub를 전달하는 새로운 marshaling class를 정의한다.

RMI클라이언트는 제공된 인터페이스를 통하여 원격객체와 상호작용한다. 하지만 이러한 인터페이스를 구현한 class들과 직접적으로 상호작용하지는 않는다. 로컬에서의 자바호출과는 달리, RMI는 지역 객체를 전송할 때 pass-by-value방식을 사용한다. 지역 객체의 주소는 단지 하나의 virtual machine내에서만 유용한 것이기 때문에 참조에 의한 전달이 불가능하다. 그러나 사용자가 네트웍을 통해 서버의 객체에 접근할 때는 RMI는 실제적인 원격구현을 copy에 의해서가 아니라 reference에 의해 객체를 전달한다.[3][4]

3 JDBC

JDBC표준 API는 Java프로그래밍 언어를 통하여 데이터베이스 관리 시스템들에 연결하는 기능을 가진 응용 프로그램의 작성을 지원한다. 이러한 JDBC는 네트워크 환경에서 클라이언트/서버 구조를 기반으로 한 데이터베이스 관리 시스템과의 연동을 지원한다.

3.1 2-tier와 3-tier 모델

JDBC표준 API는 데이터베이스와의 연결을 위해 2-tier와 3-tier 연결 모델을 지원한다. 2-tier모델은 클라이언트 시스템의 Java애플리케이션이 네트워크상의 데이터베이스 관리 시스템이 존재하는 서버 시스템으로 JDBC드라이버를 이용하여 접근하게 된다. 이러한 구조를 그림2의 (a)에서 보이고 있다.

3-tier모델은 클라이언트 명령이 'middle tier'로 전달되고 이것이 다시 서버로 전달되어지고, 결과는 서버에서 'middle tier'로 전달되고 다시 클라이언트로 전송된다. 이러한 모델의 장점은 'middle tier'가 서버에 대한 접근 제어를 총괄하므로 공동 작업 자료를 효율적으로 관리할 수 있고 사용자는 추상화된 고차원 수준의 API를 제공받을 수 있는 유리한 점이 존재한다. 이러한 구조를 그림 2의 (b)에서 보이고 있다.[1]

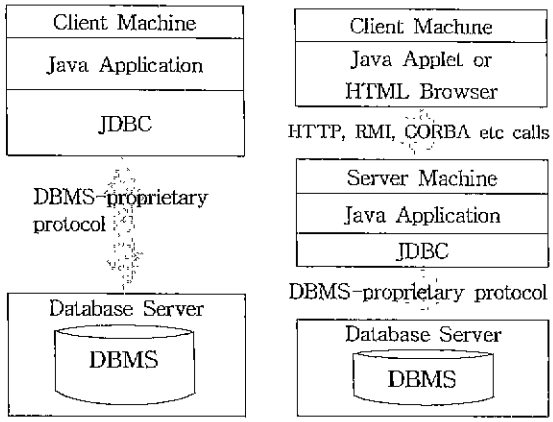

그림 2 JDBC의 2-tier 모델과 3-tier 모델

3.2 분산 객체 환경에서의 JDBC응용

앞서 살펴본 바와 같이 JDBC는 클라이언트/서버 구조의 2-tier방식을 기본으로 하고 있으며 미들웨어를 이용하는 3-tier의 경우에는 다른 네트워크 프로토콜을 이용하여 구성한다. 즉, Java RMI, CORBA, COM, HTTP등을 이용하게 된다. 이러한 3-tier모델의 경우 각 프로토콜의 이용에 관련된 표준이 없으므로 응용 시스템의 개발에서 임의로 선택하게 된다.[2][4]

기존의 JDBC 3-tier모델의 경우에 분산객체 환경을 이용하기 위해서는 미들웨어와 연결되는 부분에 대한 새로운 응용 프로그램의 구현시 빈복해서 새롭게 구현해야 하는 어려움이 생기게 된다. 또한 분산객체 환경을 도입하는 대부분의 시스템의 경우 대규모 구현이 발생하므로 반복 구현이 불가피하다. 그래서 RMI를 응용하기 위해 이미 표준화된 JDBC드라이버를 지원하는 프로토콜을 개발함으로써 데이터 베이스관리 시스템에 대한 일관된 접근과 반복구현을 최소화 함으로서 구현능률을 대폭 향상시킬 수 있다.

4. 개선된 3-tier모델

4.1 개선된 3-tier모델의 구성 및 설계

제안한 모델의 개선된 3-tier모델의 구성도를 보면 그림 3과 같다. 그림 3의 요소들 가운데 RMI를 중심으로 아래와 윗 방향으로의 화살표는 스트림의 흐름을 나타낸다. 이는 실제로는 원격 프로시져 호출은 클라이언트에서 이루어지므로 클라이언트에서 미들웨이 쪽으로의 요구가 발생하지만 사실적인 스트림의 흐름은 화살표의 방향과 일치한다. 클라이언트 내부에는 JDBC표준 API에 대한 호출을 적절한 스트림으로 변환시키는 모듈이 존재해야 하고 미들웨어의 내부에는 전

달받은 스트림을 해석하는 스트림 해석기와 이를 통해 해석된 내용을 적절한 JDBC표준 API로 전환시키는 모듈이 필요하다. 또한 결과의 반환을 위해서 JDBC표준 API에 의한 반환값을 스트림의 형태로 변환시키기 위한 변환모듈 역시 요구된다.[4][5][6]

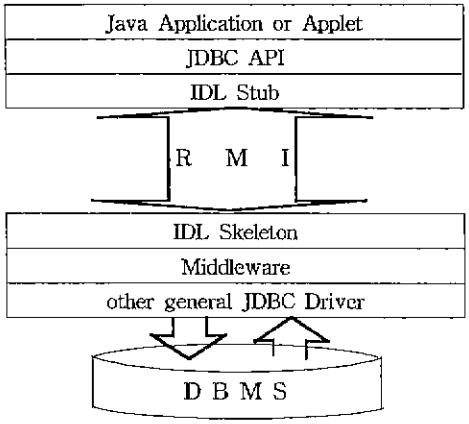

그림 3 RMI-JDBC 이용한 개선된 3-Tier 시스템

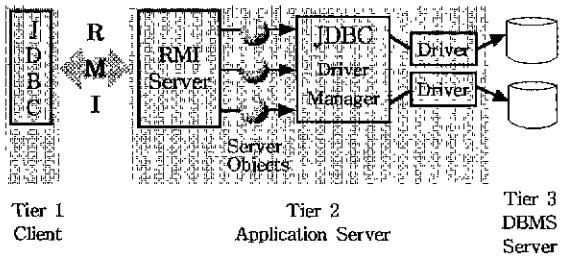

그림 4 JDBC 3-Tier 모델의 전체 구성도

4.2 동작 메카니즘

① Java응용 프로그램에서 데이터베이스 관리 시스템의 연결을 생성하는 JDBC표준 API를 호출

② 클라이언트측 모듈은 연결생성을 알리는 스트림을 형성

③ 연결생성을 지시하는 스트림을 RMI가 미들웨어에게 전송

④ 미들웨어는 전송자료를 데이터베이스 관리 시스템과의 연결을 요구하는 것임을 알고 전달된 데이터베이스 관리 시스템의 URL을 이용하여 데이터베이스 관리 시스템과의 연결을 생성

⑤ 미들웨어는 연결의 생성이 성공하면 클라이언트의 메소드 호출에서 사용된 스트림을 반환

⑥ 클라이언트는 연결의 생성이 성공했음을 알리는 반환값을 받는 경우 연결관련정보를 미들웨어로부터 RMI를 통해 받아들임

5. 제안된 시스템의 특징 및 장·단점

제안된 시스템은 middle-tier application으로써 적절한 JDBC드라이버를 통해 하나 이상의 데이터베이스와 연결할 수 있다.

클라이언트측은 여러 개의 이질적인 데이터베이스와 연결하기 위해 단지 하나의 client/server 프로토콜만을 사용하고, 이는 데이터베이스와의 연결을 기존의 표준내부에서 해결함으로써 표준상의 혼돈을 줄일 수 있다. 또한 JDBC API는 클라이언트측에만 사용하여 서버측의 JDBC드라이버를 통해 여러 데이터베이스 서버와 연결할 수 있다. 그리고 데이터베이스 서버로 접근을 하고자 할 때는 connection time시에 단지 JDBC URL만 변경하면 서버측의 코드를 변경하지 않고서도 손쉽게 엑세스 할 수 있는 장점이 있다.

CORBA나 다른 미들웨어로 설계된 시스템에 비해 순수한 자바코드로만 구성되어 있기 때문에 소스코드가 간단하고, 반복적인 코드의 구현을 감소시킬 수 있다[6] 이외에도 제안한 시스템은 Java응용 프로그램의 두 가지 형식인 애플릿과 에플리케이션 모두에 동일한 방법으로 적용되므로 applet security문제 외에는 두 가지 형식에 대한 별도의 고려와 구현이 필요치 않으므로 구현의 편리와 효율을 기할 수 있다. 그러나 이러한 RMI시스템은 다른 언어로 쓰여진 객체를 불러낼 수 없는 단점이 있긴 하지만 JDBC나 JDBC드라이버가 Java로 구현되어 있으므로 문제가 되지 않고 이 때문에 오히려 RMI외의 다른 미들웨어 사용을 피할 수 있다.

6. 결론 및 향후 연구 방향

본 논문에서는 RMI와 Java를 이용하여 데이터베이스를 이용한 응용프로그램작성 시 발생할 수 있는 문제점 가운데 데이터베이스 연결성에 관하여 살펴보았다. Java언어에서 기본적으로 제공하는 데이터베이스 연결에 관한 표준 API를 그대로 유지하면서 RMI를 활용하여 3-tier구조의 시스템 구성을 위한 방안을 살펴보고 이러한 구성을 위해 새로운 application server를 설계하여 Java가 갖고 있는 문제점 즉, JDBC를 이용한 Applet은 Download된 원래의 서버와 DB서버가 같은 경우에만 그 곳에 접속하여 DB에 엑세스 가능한 이러한 제약을 해결하였다.

앞으로의 연구 방향은 본 논문에서는 서버와 클라이언트사이의 호출시 스트림 입출력형식을 사용했는데, 이는 대량의 자료나 결과 튜플의 수가 큰 경우는 전송 효율이 떨어질 수 있다. Java언어 자체의 속도문제와 Java애플릿 등에서 이러한 문제의 발생이 나타날 수가 있으므로 대량의 스트림 전송방법에 대한 연구가 필요하다. 또한 이질적인 데이터베이스에 필요한 각각의 JDBC드라이버의 관리 기능에 대한 연구도 행해져야 할 것이다.

참고문헌

[1] C. J. Date, "An Introduction To Database Systems", Addison-Wesley Publishing Company, 1990.

[2] Gary Cornell and Cay S. Horstmann, "Core JAVA", The SunSoft Press, 1997.

[3] Mike Cohn외 5인, "DEVELOPER'S REFERENCE JAVA", Sams.net, 1997.

[4] Robert Orfali and Dan Harkey, "Client/Server Programming with JAVA and CORBA", John Wiley and Sons Inc., p.239~268, 1997.

[5] 왕창종 외 1인, "분산 객체 컴퓨팅 기술, CORBA 프로그래밍", 대림, p.10~23, 1998.

[6] 이진용 외 1인, "CORBA 분산 객체 환경을 위한 JDBC 하위 드라이버의 설계 및 구현", 한국 정보과학회 가을 학술발표논문집(III) Vol. 24, No. 2, p.201~204, 1997.