그리드 기반 분산 객체지향 가상환경 시스템에서의 웹서비스 통신기법

이기호° 김형래 정창성
고려대학교
{yikiho°, nolight, csjeong}@korea.ac.kr

Web service communication method on Distributed Object-Oriented Virtual Environment System on Grid

Kiho Yi°, Hyunglae Kim, Changsung Jeoung
Korea University

요 약

그리드 기반 분산객체지향 가상환경 시스템인 DOVE-G는 많은 리소스를 요구로 하는 분산된 그리드 컴퓨팅 어플리케이션에서 효율적인 병렬프로그래밍 환경을 지원하는데 이에 웹서비스를 이용해 다양한 어플리케이션에서 고급 그리드 컴퓨팅 환경을 제공할 수 있다. 본 논문에서는 웹서비스를 이용해 DOVE-G의 implementation을 모르더라도 DOVE-G 서비스를 쉽게 이용할 수 있도록 해주는 Interface Object와 Agent Object 두 개의 DOVE-G Object를 이용한 통신기법을 제시한다.

1. 소 개

웹서비스[1]는 네트워크에서 기계간의 상호 운용을 지원하도록 설계된 소프트웨어 시스템으로 공개된 표준과 프로토콜을 이용하여 서로 다른 플랫폼에서 동작하는 다양한 어플리케이션 소프트웨어간의 상호 운용을 지원한다.

DOVE-G[2, 3]는 그리드 기반의 분산객체지향 병렬 프로그래밍 환경으로 그리드 환경에서 DOVE-G Object로 incapsulation된 그리드 서비스를 포함하는 어플리케이션이다.

각 DOVE-G Object는 서로 다른 하드웨어에 위치하고 있지만 DOVE-G 시스템은 DOVE-G Object를 하나의 가상공간에 있는 Object들로 보이게끔 도와 사용자에게 그리드에서 쉽게 사용할 수 있는 병렬프로그래밍 환경을 제공한다. 단 DOVE-G는 다양한 그리드 어플리케이션 사이의 통신에는 약하다는 단점을 가지고 있다. 만약 DOVE-G 시스템이 웹서비스와 통신할 수 있다면 DOVE-G Object들을 이용해 다양한 그리드 어플리케이션을 쉽게 개발할 수 있다.

본 논문에서는 두개의 특별한 DOVE-G Object를 생성해 다양한 그리드 어플리케이션에서 웹서비스와의 통신이 가능하게 한다.

2. DOVE-G 분산 객체

DOVE-G는 분산객체 각각의 Method Invocation 메커니즘을 이용해 여러 분산 객체들과 상호작용하는 분산 Object 모델을 기반으로 하며 Fig.1.에 이 메커니즘을 보여준다.

이 모델은 주어진 Task로부터 분할된 Subtask 들이 실행을 위해 여러 호스트로 동시에 수행을 요청하는 분산 Object들을 수집해 병렬 어플리케이션 모델을 만든다.

DOVE-G Object A는 두 종류의 Object를 포함하고 있는데 Implementation Object IM_A와 다른 원격 DOVE-G Object B를 위한 Interface Object IN_B이다. Interface Object IN_B는 B의 Implementation Object IM_B와 통신하는데 있어 인터렉션 포인트를 제공하는데 Interface Object IN_B가 마치 로컬 Object로 존재하듯 Implementation Object IM_A는 B의 Interface Object IN_B의 멤버 펑션을 호출하는 것으로 원격에 있는 다른 DOVE-G Object B의 Method Invocation을 만든다.

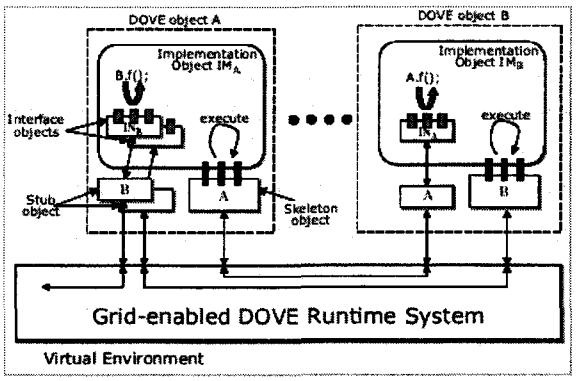

Fig.1.DOVE-G 분산객체 모델

Interface Object와 Implementation Object는 각각 스텁과 스켈레톤 Object에 연결되어 있는데 Interface Object에서 Method Invocation은 해당 스텁 Object의 인보케이션 메시지로 변환되어 DOVE-G 런타임 시스템으로 연결되어있는 Implementation Object에게 보내진다.

Implementation Object쪽에서 이 메시지는 스켈레톤 Object

를 통해 Implementation Object와 매치된 멤버 펑션을 호출하기 위해 디스패치하게 되며 멤버 펑션을 위한 응답 메시지는 스텁 Object로 되돌아가는 Implementation Object로부터 보내져 일반 펑션의 호출처럼 돌아오게 된다. 이 메커니즘은 Object의 장소에 구애받지 않고 DOVE-G Object에 접근할 수 있게 한다.

3. DOVE-G 에서의 웹서비스

웹 서비스는 XML에 기반을 둔 플랫폼과 구현언어에 독립적인 컴포넌트 기반의 분산 컴퓨팅 기술이다. 기존의 분산 컴퓨팅 기술이 폐쇄 환경에서 사용되었다면 웹 서비스는 인터넷을 기본 무대로 하는 개방형 환경에서 사용될 수 있는 분산 컴퓨팅 기술로 웹서비스를 이용하면 웹에서 쉽게 분산 컴퓨팅 플랫폼을 만들 수 있다.

DOVE-G는 C++ 라이브러리로 동작한다. 여기에서는 웹서비스 개발 툴킷으로 XML에서 C/C++까지의 까지의 언어를 지원하고 SOAP/XML 웹서비스를 C/C++로 쉽게 개발할 수 있는 gSOAP[4]을 DOVE-G Object의 웹서비스 통신에 이용한다.

4. 웹서비스 통신에서의 DOVE-G Object

웹서비스의 수행에는 두 가지 방법이 있는데 웹서비스 consuming과 웹서비스 providing이다.

웹서비스 consuming은 웹서비스를 요청하면 웹서비스 스텁이 웹서비스 provider로부터 WSDL을 파싱하는 것이고 웹서비스 providing은 웹서비스 consumer가 요구로 하는 웹서비스 기능을 WSDL로 정의해 implementation 하도록 하는 것이다.

DOVE-G 시스템에서 웹서비스 통신을 위해 제안하는 두 개의 특별한 DOVE-G Object는 웹서비스 consuming을 위한 Agent object와 웹서비스 providing을 위한 Interface Object이다. Fig.2.은 두 Object의 역할을 보여준다.

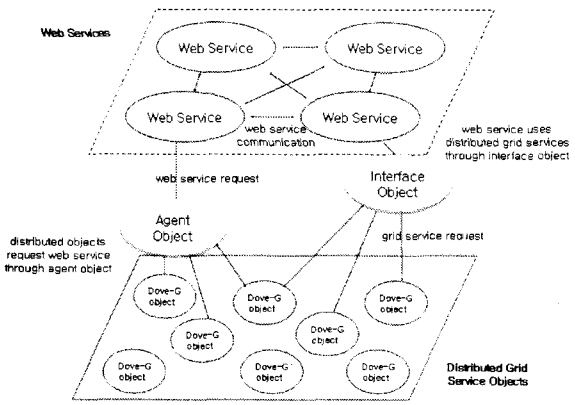

Fig.2. DOVE-G와 웹서비스 간 Interface Object와 Agent 이용한 통신

4.1. Agent Object

Agent Object는 웹서비스 consuming을 위한 DOVE-G Object로 DOVE-G 실행과 웹 서비스 통신 모듈을 모두 가지고 있으며 DOVE-G Object들이 사용하는 Remote Method Invocation 메시지를 SOAP 메시지로 변환한다. Agent Object를 사용해 DOVE-G Object는 웹서비스를 위해 디자인된 여러 다른 그리드 어플리케이션들과 통신이 가능하다.

4.1.1. Agent Object 생성

Agent Object는 DOVE-G Object의 일종으로 일반적인 DOVE-G Object의 생성방법과 생성방법이 비슷하며 웹서비스에 액세스하는 웹서비스 스텁 루틴들을 반드시 포함하고 있어야 한다. Fig.3.은 어떻게 Agent Object가 만들어지는지 보여준다.

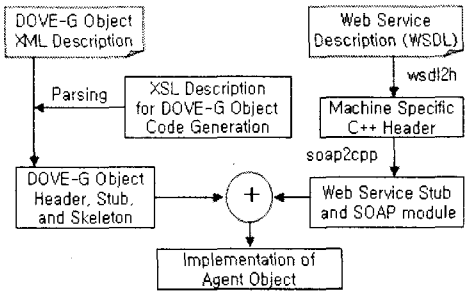

Fig.3. Agent Object 생성 과정

DOVE-G Object를 만들기 위해서는 이름, 데이터 타입, 가능을 XML로 정의해 주고 정의가 끝나면 DOVE-G Object 스텁과 implementation 가능을 정의한 XML 파일에 미리 정의한 XSL 파일을 적용한다. 여기서 Agent Object를 만들기 전에 Agent Object에서 어떤 웹서비스가 implementation 될지 결정해야 하는데 우리는 이 과정을 gSOAP 유틸리티를 이용하여 파싱된 WSDL에서 나온 웹서비스 스텁 펑션으로 알 수 있다.

Agent Object의 implementation에서는 XML로 정의된 DOVE-G Object에 대한 기술(description)과 두 gSOAP 유틸리티를 이용해 나온 WSDL로 된 웹서비스 커뮤니케이션 모듈이 요구된다.

4.1.2. Agent Object 실행

DOVE-G 시스템은 분산 객체들에 대한 단일 환경 통합과 같이 그리드 환경과 어플리케이션으로 분산 객체들의 단일뷰를 지원하는데 이것은 DOVE-G 아키텍쳐가 정의가 잘 된 implementation layer들로 구성되어있고 DOVE-G Object가 DOVE-G 서비스 layer와 런타임 시스템 layer를 통해 의사소통이 가능하기 때문이다.

DOVE-G 서비스 layer는 Object 매니저와 모니터 Object로 구성되어 Object 생성, Naming 서비스, 리소스 할당 등의 시스템 서비스를 제공하는데 명확한 컴퓨팅 환경을 만들기 위해 DOVE-G Object는 Naming 서비스로 각 호스트에 있는 Object 매니저를 증명한다.

Agent Object는 Object 매니저가 다른 DOVE-G Object를 생성하는 것과 비슷한 과정으로 생성되며 생성 후 부터는 DOVE-G Object는 Agent Object를 통해 웹서비스에 액세스할 수 있다. 비록 DOVE-G Object가 각 호스트에 나뉘어 위치

하지만 그들은 그들의 Object 매니저에게 요청하여 다른 DOVE-G Object를 참조하는 방법으로 서로를 연결한다. 만약 DOVE-G Object가 특정 웹서비스를 이용하고자 한다면 직접 웹서비스에 연결하지 않고 Agent Object 스텁 펑션에 웹서비스를 요청한다. Agent Object가 웹서비스에 질의를 하면 어떤 웹서비스가 요청한 기능과 맞는지 다른 DOVE-G Object를 찾고 Agent Object는 DOVE-G Object의 프락시에 웹서비스를 요청하는데 Agent Object는 웹서비스 요청뿐만 아니라 서비스 provider로부터 결과를 받는 역할도 함께 한다.

Agent Object는 웹서비스 응답메시지가 도착되면 그 결과(원래 요청했던 웹서비스의 결과)는 DOVE-G Object로 전달된다.

Fig.4.는 DOVE-G Object가 Agent Object들을 통해 웹 서비스를 이용하는 것을 보여준다.

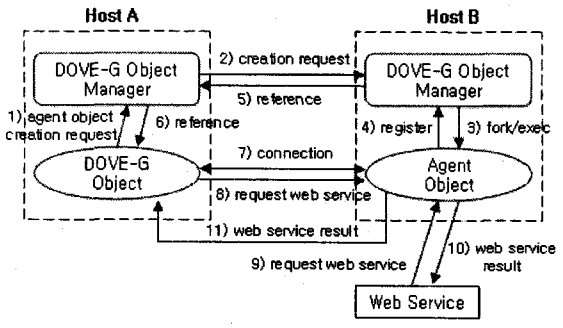

Fig.4. Agent Object 실행 과정

DOVE-G Object는 Agent Object를 이용하면 해당 웹서비스의 implementation을 신경 쓰지 않고 쉽게 웹서비스에 액세스할 수 있다

4.2. Interface Object

Interface Object는 웹서비스와 다른 DOVE-G Object와의 통신을 제공하는 역할을 한다. 고로 웹서비스 클라이언트는 Interface Object에 웹서비스를 요청하여 그리드 서비스의 implementation과 무관하게 DOVE-G 시스템을 사용할 수 있으며 Agent Object보다 생성과 실행이 복잡하다.

4.2.1. Interface Object 생성

Interface Object는 웹서비스 providing의 한 방법으로 웹서비스 클라이언트를 위한 WSDL로 된 웹서버 모듈을 가지고 있어야한다.

Interface Object는 DOVE-G Object와 웹서비스 provider에서 동작하는데 DOVE-G Object에서의 생성과정은 Agent Object와 유사하며 생성 시에는 XML파일로 이름, 데이터 타입, 펑션을 기술해 주어야한다.

스텁의 생성을 위한 파싱과 Interface Object 생성은 웹서비스 내부 구조적으로 Agent Object와 다르며 웹서버 모듈, Interface Object 모두 gSOAP 유틸리티를 사용하는데 gSOAP은 stand-alone HTTP 웹서버 모듈을 포함하고 있기 때문에 gSOAP 라이브러리를 이용하는 Interface Object에 웹서버 모듈을 삽입할 수 있다.

Fig.5.는 어떻게 Interface Object가 DOVE-G Object를 가지고 implementation 되는지 보여준다.

Interface Object를 만들 때는 펑션, 서비스 이름, 웹서비스에서의 C++ 헤더 파일 폼의 위치를 기술해야하는데 이 헤더파일은 gSOAP의 soap2cpp 유틸리티를 통해 웹서비스 스텁으로 만들 수 있으며 SOAP 통신간 모듈은 stand-alone HTTP 서버에 포함시킨다. 그리고 Interface Object의 서비스 기술(description)은 WSDL로 한다.

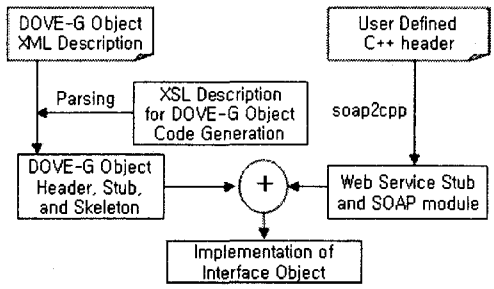

Fig.5. Interface Object 생성과정

4.2.2. Interface Object 실행

다른 DOVE-G Object처럼 Interface Object의 Naming과 생성은 object 매니저가 컨트롤 한다.

Interface Object는 DOVE-G Object의 참조를 여러 호스트에 위치한 Object 매니저에게 요청하여 다른 Object들과 대화가 가능하며 Interface Object는 웹서비스 클라이언트 관점에서 본다면 웹서비스(웹서버)이며 그것 또한 DOVE-G Object이다.

웹서비스 클라이언트는 Interface Object를 DOVE-G 시스템의 게이트웨이로 사용할 수 있는데 이는 DOVE-G 시스템과 implementation에 대해 알지 못해도 웹서비스 클라이언트는 Interface Object에 질의를 함으로써 DOVE-G 시스템을 사용할 수 있다는 것이다.

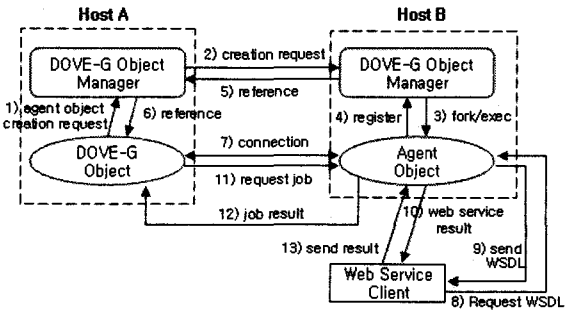

Fig.6. Interface Object 실행 과정

웹서비스 클라이언트는 누가 어떤 웹서비스를 사용하고자 하는지 Interface Object에 WSDL로 요청하고 웹서비스 클라이언트는 Interface Object로 보내진 WSDL로부터 웹서비스 스텁

펑션을 얻을 수 있다. 웹서비스 스텁이 implementation된 후 클라이언트는 특별한 파라미터로 Interface Object에게 웹서비스를 질의하게 되는데 이때 Interface Object는 XML SOAP 메시지 양식으로 웹서비스 요청을 받게 된다. 그 후 Interface Object는 클라이언트의 요청 메시지를 추출해 Remote Method Invocation 메시지로 바꾸어 다른 DOVE-G Object에 전달한다.

Fig.6.은 웹서비스 클라이언트가 어떻게 Interface Object가 다른 DOVE-G Object들과 통신하는지 보여준다.

DOVE-G Object는 Interface Object로부터 웹서비스의 처리를 요청 받고 그 결과를 Interface Object에게 돌려준다. 그리고 최종적으로 웹서비스 클라이언트에게 결과를 돌려준다.

5. 결 론

웹서비스는 어플리케이션과 비즈니스 프로세스간의 통신에 효율적인 통신방법을 제공한다. 그래서 우리는 그리드 기반 분산객체지향 가상 컴퓨팅 환경에서의 두 가지 웹 서비스 통신기술을 제안하였다.

Agent Object는 DOVE-G Object의 웹서비스 Accessing을 위한 것임과 동시에 다양한 그리드 어플리케이션들과 비즈니스 프로세스를 위해 DOVE-G 시스템을 웹서비스로 이용할 수 있게 한다.

Interface Object는 웹서비스 클라이언트가 DOVE-G 시스템의 implementation을 몰라도 높은 효율의 그리드 컴퓨팅 환경을 제공한다. 이로써 우리는 Agent Object와 Interface Object 두 agent를 이용해 웹서비스와 DOVE-G 시스템에서 많은 장점을 얻게 된다.

6. 참 고

[1] Web Service : http://www.w3g.org/2002/ws

[2] Y.J. Woo, and C.S. Jeong, "Distributed Object-Oriented Parallel Programming Environment on Grid", Lecture Note in Computer Science 2668, PaCT, May 2003,pp.562-570

[3] Y.J. Woo, and C.S. Jeoung, "DOVE-G: Design and Implementation of Distributed Object-Oriented Virtual Environment on Grid", Lecture Note in Computer Science 2763, International Conference on Computational Science and its Applications, September 2003, pp. 555-567

[4] gSOAP:http://www.cs.fsu.edu/~engelen/soap.html