# CORBA를 이용한 디지털도서관 분산환경의 통합

# Integrating Distributed Environment of Digital Library with CORBA

윤소영, 연세대학교 대학원 문헌정보학과

Yoon So Young, Dept. Lib. & Info. Science, Graduate School of Yonsei University

본 논문에서는 네트워크와 서버/클라이언트 기술의 도입으로 새로운 모습으로 발전하고 있는 디지털도서관의 분산환경을 통합하기 위한 방안으로 CORBA를 이용하는 모델을 제시한다. 본 모델은 인터넷의 웹 서버와 코바 정보저장소인 서버로 구성된 분산서버를 단일한 클라이언트에서 사용할 수 있도록 자바 애플릿을 클라이언트 환경으로 한다.

## 1 서론

디지털도서관은 인터넷의 확산과 클라이언트/서버 기술의 도입으로 새로운 모습으로 발전하고 있다. 기존의 단일 서버에만 존재하던 시스템 어플리케이션이 이제는 여러 클라이언트와 서버에 분산되고 있다. 즉 네트워크의 확산과 더불어 기존의 클라이언트/서버 환경을 적극적으로 발전시켜 여러 시스템을 지리적으로 분산되어 있는 곳에 분산시켜 각기 독립적인 기능을 수행하거나 필요할 때 협력하는 분산 시스템이 디지털도서관 구축에 있어서도 보편화되기 시작했다. 초기의 분산시스템은 소켓과 RPC (Remote Procedure Call) 기능을 이용하여 개발되었으나 이후에는 DCE (Distributed Computing Environment)등의 통합 분산 환경을 통해 개발되었다.

그러나 기존의 이러한 방법은 분산 환경 하에서 다양한 하드웨어와 응용 소프트웨어, 운영 체계, 데이터베이스 등을 일관되게 연결해 주지 못했다. 이러한 상황을 해결하기 위하여 OMG (Object Management Group)에서는 코바 (CORBA: Common Object Request Broker)를 발표하였다. 코바는 구현 언어나 개발 환경이라기보다는 객체 기술에 근거한 통합 기술이다. 따라서 실제 구현된, 또는 구현되는 언어에 상관없이 각 시스템들을 연결, 통합할 수 있는 방법을 제공한다.

본 논문에서는 이러한 코바를 이용하여 디지털도서관의 분산 환경을 통합하고자 한다.

## 2 디지털도서관 분산 환경

디지털도서관의 분산 환경뿐 아니라 일반적인 분산 컴퓨팅 시스템 (distributed computing system)에 의해 제공되는 서비스인 분산 처리는 여러 대의 컴퓨터를 네트워크로 연결하여 컴퓨터 상호간에 업무적 기능들을 네트워크를 통해서 성취하는 형태를 의미한다. 기술적인 관점에서는 컴퓨터 상호간에 주기억장치를 공유하지 않는 것으로, 정보 교환은 광역 변수 (global variable)를 통해 전달되지 않고 네트워크를 통한 메시지 교환으로 이루어지는 것을 의미한다. 사용자 관점에서의 분산 처리는 컴퓨터 통신망에 연결된 여러 자원들을 마치 사

용자 혼자서 사용하는 것처럼 지원해주는 것이며, 사용자들에게 컴퓨터들 사이의 관계가 투명 (transparent)하고 무관 (irrelevant)하게 보이도록 만드는 것을 말한다.

클라이언트/서버 기술은 분산 처리 기술의 특수한 모형으로 통신 망 기술의 발달과 개인용 컴퓨터나 워크스테이션 같이 성능도 좋고 규모도 커지는 클라이언트 급의 컴퓨터들이 등장하면서 출현한 시스템 구성상의 새로운 기술이다. 이 기술은 외국에서는 1980년도 말에 정보 공학분야에서 발표되어 이미 정보 통신 관련 분야에서 활용되고 있다.

분산처리 환경 하에서의 클라이언트/서버 응용 도구는 멀티미디어를 이용한 클라이언트/서버 형태의 응용 프로그램을 지원하며 분산처리 시스템 소프트웨어의 미들웨어 (midleware)[1]로서 기능을 수행한다. 클라이언트/서버 응용 도구에 대한 표준화된 참조 모델 (reference model)은 없으나, 분산처리 시스템 소프트웨어 기술 측면에서 살펴보면 [그림 1] 과 같이 4계층으로 분류된다.

| | |
|---|---|
| 사용자웨어 계층 | 계층 4 |
| 미들웨어 계층 | 계층 3 |
| 엔터프라이즈웨어 계층 | 계층 2 |
| 하드웨어 계층 | 계층 1 |

【그림 1】 클라이언트/서버 응용개발 도구 참조 모델

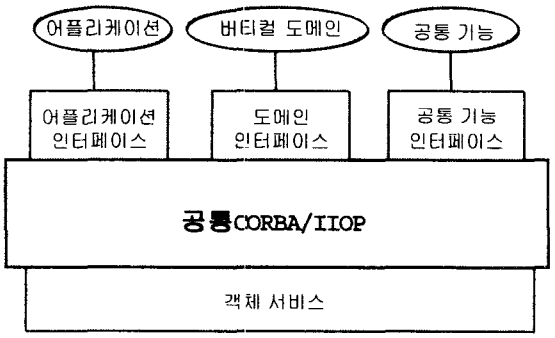
【그림 1】에서 미들웨어 계층은 클라이언트/서버 응용 개발 도구의 가장 핵심 부분으로서, 데이터 관리기능과 객체 관리기능, 그리고 통신 관리 기능으로 구성되어 있다. 객체 관리 측면에서는 사용자웨어에서 생성한 개체들을 관리하는 기능, 이종의 소프트웨어 사이에 데이터를 교환할 수 있는 기능이 있다. 통신 관리 측면에서는 상위 계층인 사용자 계층을 통하여 엔터프라이즈웨어 계층과 하드웨어 계층에 통신 방식으로 접근할 수 있도록 하는 기능과 함께 분산 시스템 소프트웨어에 접속하는 기능이 있다.

1) 미들웨어는 클라이언트/서버 시스템에서 클라이언트가 서버로 서비스 요청을 하는 일종의 가교 역할을 한다. 실제 서비스 (일반적으로 서버의 역할)와 어플리케이션 로직 (일반적으로 클라이언트의 역할)은 제외되며, 다양한 서비스와 관련해 여러 가지 미들웨어가 존재한다.

## 3 코바를 이용한 분산 환경의 통합

이기종 간의 클라이언트/서버 시스템을 효율적으로 통합, 관리할 수 있는 시스템의 요구가 본격화되면서, 여기에 부합해 객체지향 기술을 이용한 관련 표준을 제정하기 위하여 OMG라는 비영리 단체가 1989년 설립되었다. OMG는 현재 마이크로소프트사를 포함한 800여 개의 컴퓨터 관련 단체, 개발자, 사용자가 참가중이며, 객체지향 기술을 기반으로 이종의 분산된 환경에서 어플리케이션들이 서로 통합 할 수 있는 시스템 통합 표준 기술을 탄생시켰다. 이 표준이 바로 객체 관리 구조 (OMA: Object Management Architecture)이다. OMA는 어플리케이션 사이의 결합뿐만 아니라 객체의 생성, 소멸에서부터 저장, 트랜잭션 기능에 이르기까지 분산 객체 환경에 필요한 모든 서비스를 총칭하는 것이다. OMA의 특징은 특정 기술을 이용하여 구체화된 체품을 만드는 것이 아니라, 그 기술을 표현한 규격을 제시하여 업체들이 각자 구현하게 하는 것이다. OMA는 【그림 2】와 같이 어플리케이션, 객체 서비스 (Common Object Services), 공통기능 (Common Facilities), 버티컬 도메인 (Vertical domain)이라는 4개의 모듈로 구성되고 객체 요청 중개자 (ORB: Object Request Broker)는 각각의 모듈을 연결하기 위한 통신 메커니즘으로서 위치하고 있다.

· 어플리케이션 : OMG에서 정의된 규격으로 작성한 어플리케이션이나 클래스이다. 서버 객체와 클라이언트 객체로 구성된다. 어플리케이션 예를 들면 워드프로세서 등을 말한다.




【그림 2】 OMA의 구조

· 객체 서비스 : 객체의 기본적인 조작을 제공하기 위한 서비스이며, 어플리케이션은 ORB를 통해서 이들 서비스를 이용할 수 있다.

· 공통 기능 : 공통기능은 객체 서비스에 비해 보다 높은 수준의 기능을 제공하는 것으로 복합문서, 인쇄, 데이터 변환 등 범용적으로 어플리케이션이 공통으로 이용할 수 있는 기능이다.

· 버티컬 도메인 : 버티컬 도메인은 비즈니스용 어플리케이션을 대상으로 검토되고 있는 서비스이다. 일반적으로 사용할 수 있는 서비스와 특정화된 서비스가 있다.

· 객체 요청 중개자 : ORB는 컴퓨터 하드웨어의 시스템 버스처럼 객체 사이의 요청/응답을 전달하는 메커니즘이다. 이것은 객체가 다른 지역 객체 또는 원격 객체로부터 서비스를 요청하고 응답받을 수 있도록 해준다. ORB는 다른 업체 제품이라도 호환이 가능한 CORBA라는 객체 버스를 정의하고 있다. CORBA는 다양한 종류의 분산 미들웨어 서비스를 제공하며, 객체가 실행 시에 서로를 찾고 서로의 서비스를 호출할 수 있게 한다. CORBA는 기존의 RPC (Remote Procedure Call)나 MOM (Message Oriented Middleware), DB 저장 프로시저 등의 미들웨어에 비해 다양하다.

이들 OMA 기능 중 코바는 컴퓨터 내부의 버스처럼 서로 다른 어플리케이션 들 사이의 버스역할을 하는 모듈로서 OMA의 객체간 통신을 담당한다. 결국 코바는 OMA의 한 부분이고 ORB는 코바의 핵심 기술이다. 현재 OMG는 1990년 OMA를 발표한 이래 지금까지 코바 2.0 스펙을 발표하였으며, 3.0을 준비중에 있다.

코바를 이해하는 데 있어 중요한 점은 코바는 서버라는 용어를 사용하지 않는다는 것이다. 왜냐하면 코바는 객체지향 개념을 바탕으로 하여 원격지의 클라이언트가 원격지에 있는 서버를 호출하는 것이 아니라 객체의 메소드를 호출함으로써 서비스가 이루어진다.

따라서 코바에서는 서버가 아니라 구현 객체(Object Implementation)라는 용어를 사용한다. 이때 고려해 볼 점은 사용자가 클라이언트와 구현 객체 사이의 통신 부분을 직접 관여해야 하는가에 대한 여부와 구현시 사용하는 프로그래밍 언어에 따라 각기 다르게 작성해야 하는가 하는 문제이다.

이러한 점을 해결하고자 코바에서는 IDL (Interface Definition Language)이라는 표준 언어를 제공한다. 사용자는 IDL을 사용하여 원하는 어플리케이션을 작성할 수 있다. 일단 작성된 IDL은 코바에서 제공하는 IDL 컴파일러를 통해 컴파일된다. 이 결과 원하는 프로그램 언어로 작성된 클라이언크 코드와 구현 객체 코드를 얻을 수 있다. 이 코드는 C나 C++또는 자바 등의 사용자가 원하는 형태로 제공되며, 통신을 위해 필요한 모든 기능이 자동적으로 포함된다.

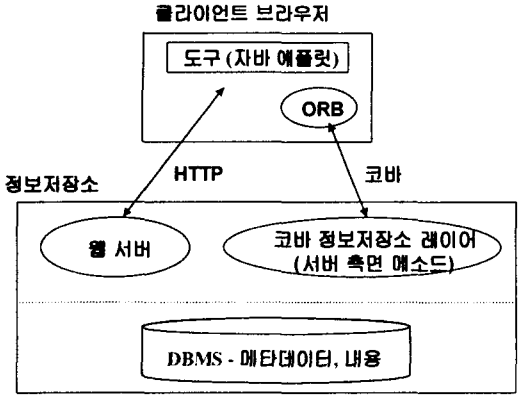
코바의 클라이언트에는 클라이언트 스터브와 어플리케이션 수행시 원하는 메소드를 호출하는 동적 호출 (dynamic invocation) 기능을 통해 구현객체를 호출할 수 있다. 클라이언트에 의해서 요청된 서비스는 코바를 통해 원격지의 객체에 전달되고 구현 객체에 의해서 처리된 결과는 다시 클라이언트에게 반환된다. 이때 호출시 전달되는 정보로는 호출시 대상이 되는

객체와 메소드, 전달 인자 등이 있다.

인터넷, 인트라넷이 보편화된 환경에서 코바를 웹에 연결하기 위한 방법으로 기존의 CGI를 이용하여 게이트웨이를 구성하는 CGI의 문제를 내포하고 있어 바람직하지 못하다.

다른 방법으로는 코바 IDL 컴파일러를 확장하여 해당 IDL 파일을 자바와 같은 웹에 적합한 코드로 생성해 주는 방법이 있다. 이 방법은 코바 IDL을 그대로 사용할 수 있다는 장점이 있다. 이 방법에서 기존의 웹 환경의 클라이언트/서버 사이 통신은 IIOP (Internet Inter ORB Protocol)를 사용한다.

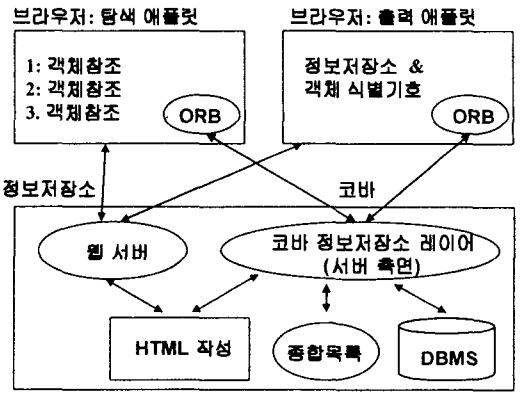
구축환경은 기존의 웹 환경과 동일하다. 단지 다른 점은 코바 IIOP를 지원하는 웹 브라우저와 코바와 연동 가능한 웹 서버에 있다. 개발된 클라이언트 프로그램은 코바 IIOP를 지원하는 웹 브라우저에 HTTP를 통해 다운로드된다. 다운로드된 프로그램은 IIOP를 통해 코바와의 서비스를 개시하게 된다. 【그림 3】은 분산환경을 위한 개략적인 구성도이며, 【그림 4】는 이를 구체한한 것이다.




【그림 3】 개략적인 분산환경 통합 구성도

## 4 향후과제 및 결론

본 논문에서는 최근에 디지털도서관의 새로운 환경인 분산처리 시스템, 코바를 이용한 분산 환경의 통합에 대해 다루었다.




【그림 4】 구체적인 분산환경 통합 구성도

현재 코바는 다양한 비즈니스 분야에서 시스템 통합을 위한 기반 기술로서 각광받고 있다. 특히 코바 IDL이 ISO/IEC를 통해 표준으로 제정되면 더 많은 분야에서 개방형 통합 기술로 채택될 것이다.

현재 디지털도서관은 인터넷을 기반으로 확산되고 있는 상황이다. 따라서 분산 객체를 지원하는 코바와 플랫폼 독립성을 지원하는 자바(JAVA)를 결합함으로써 기존의 시스템과 향후 개발될 시스템의 기반 기술로서 중요한 위치를 차지할 것이다.또한 액티브 X의 COM/OLE와의 연동에 대하여도 지속적인 연구가 필요할 것이다.

## 참고문헌


이상구 외 편역. 1998. 「CORBA & Java 분산객체 기술」. 서울 : 교학사.

최한석 외. 1996. CORBA 기능을 이용한 정보검색 시스템 통합에 관한 연구. 정보관리학회지 13(2): 223-242.

Vinoski, Steve. 1997. "CORBAL Integrating Diverse Applications Within Distributed Heterogeneous Environments," IEEE Communications Magazine 14(2).

Hurley, Bernie. 1998. "Interface and Repository Considerations For a System Architecture To Support A Distributed Digital Library". http://sunsite.berkeley.edu/moa2/papers/braindump2-03.html