COM+ 기반의 다중 계층 아키텍처 환경

이환진*, 최병엽**
고려대학교 컴퓨터과학기술대학원 미디어공학과
e-mail : {hwanjini, webserver}@korea.ac.kr

Multi Tier Environment based on Com+

Hwan-Jin Lee*, Byung-Youb Choi**
*Dept. of Media Engineering, Korea University

요 약

최근 사용자 요구 사항의 증대로 기존 2계층 아키텍처 기반 시스템에서의 제약 요인을 개선한 다중 계층 아키텍처 기반의 시스템으로 전환하고 있는 추세이다.
본 연구의 목적은 기존 2계층 아키텍처 기반의 시스템과 다중 계층 아키텍처 기반의 시스템에 대한 비교 우위의 분석을 통하여 보다 나은 시스템을 도출하고, 도출된 시스템의 구현 방안을 모색하는 데에 있다. 이에 본 연구에서는 2계층 아키텍처와 다중 계층 아키텍처의 구성 체계를 비교한 후 다중 계층 아키텍처의 비교 우위를 알 수 있었다. 또한 이의 구현을 위한 주요 개념인 COM+, 컴포넌트, 객체 등을 분석하였으며, 다중 계층 아키텍처 기반 시스템의 클라이언트 시스템 계층, 비즈니스 계층을 구현하기 위한 방안을 제시하였다.

Keyword : 2 Tier, Multi Tier Environment, COM+, Object, Component, OO, OOP

1. 서론

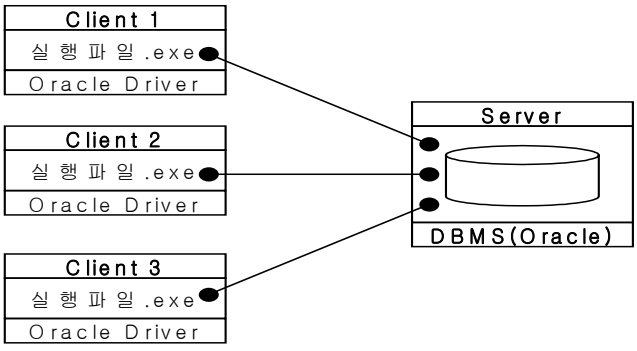
최근 들어 인터넷과 초고속 정보 통신망을 비롯한 여러 통신망의 발달로 고정된 영역에서 작업하던 사용자들은 공간의 제약없이 작업할 수 있는 환경을 요구하고 있다.

이에 기존 2계층 아키텍처 환경의 클라이언트/서버(C/S ; Client/Server) 시스템에서 웹 기반의 다중 계층 아키텍처 시스템으로 전환하는 기업이 점차 늘어나고 있으며, 특히 다중 계층 아키텍처 시스템은 클라이언트/서버 시스템에 비하여 상대적으로 시스템 개발시의 유연성 및 확장성 측면과 유지 보수 및 관리비 측면이 우수하기 때문에 시스템을 전환하는 사례는 지속화될 전망이다.

본 연구에서는 2계층 클라이언트/서버 시스템과 다중 계층 아키텍처 시스템의 주요 구성 요소를 분석하여 다중 계층 아키텍처 시스템의 비교 우위를 검토하고, 보다 저비용으로 기존 2계층 클라이언트/서버 시스템을 웹 기반 다중 계층 아키텍처 시스템으로 전환하는데 필요한 구현 방법을 제시하고자 한다.

2. 계층별 아키텍처 시스템의 비교

2.1 2계층 아키텍처



(그림 1) 2계층 아키텍처의 주요 체계

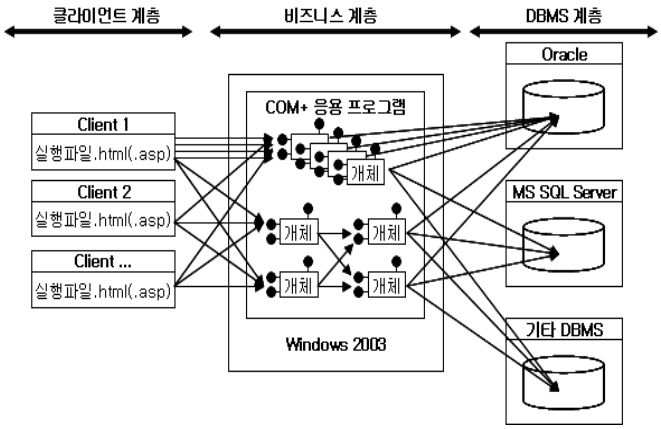
(그림 1)의 클라이언트 시스템은 사용자가 비즈니스 로직과 데이터 액세스 코드를 관리하는 인터페이스(Interface) 기능을 수행하고 있다.

각각의 클라이언트 시스템은 응용 프로그램((그림 1)의 '실행파일.exe' 참조)이 한 사용자당 한 커넥

션을 보유하여야 하며, 응용 프로그램이 종료될 때까지 해당 커넥션을 유지하고 있어야 하기 때문에 클라이언트 응용 프로그램의 개수가 증가될수록 데이터베이스 관리 시스템(DBMS ; DataBase Management System)의 연결에 따른 라이센스(License) 비용이 추가적으로 발생되며, DBMS 접속 계정을 만들고 DB 객체 참조에 대한 권한을 서버 시스템에서 개별적으로 설정(Table, Procedure 등) 해야 하므로 사용자 권한 관리가 복잡해지며, 클라이언트 시스템 응용 프로그램의 개선 및 기능 보강에 따른 프로그램 배포시에도 용이하지 않다.

서버 시스템에서는 데이터를 저장 및 관리하는 DBMS의 역할을 담당하며, 클라이언트 시스템의 요청에 의해 응답하고, 제한된 비즈니스 로직, 즉 함수(Function), 프로시저(Prodedure), 트리거(Trigger) 등의 기능을 수행하게 된다[1].

2.2 다중 계층 아키텍처



(그림 2) 다중 계층 아키텍처의 주요 체계

(그림 2)의 다중 계층 아키텍처는 크게 클라이언트 시스템, 비즈니스 로직, DBMS 이상 3부문으로 계층 분리시켜 해당 계층마다 최적화된 상태의 기능을 제공할 수 있으며, 시스템의 유지 보수 및 관리를 용이하게 지원함으로써 비용을 상대적으로 최소화할 수 있는 방식이다.

다중 계층 아키텍처 시스템의 주요 기능으로는 미들티어 상의 애플리케이션 서버 공유, 클라이언트 시스템의 관리, 안정된 분산 데이터 처리 등과 같은 형태로 구분할 수 있다.

1) 클라이언트 계층

클라이언트 계층은 2계층 아키텍처 환경의 사용자 인터페이스와 유사한 역할을 수행하지만 중요한 차이점으로는 최소의 인터페이스 로직만을 내장하여 사용자가 원하는 정보를 보여주고 입력하는 역할만을 담당한다. 웹 기반 시스템에서 클라이언트 응용 프로그램은 웹 브라우저에서 실행되며, 웹 브라우저는 HTTP(Hyper Text Transfer Protocol) 프로토콜을 사용하여 웹 서버에 처리사항을 요청하고 웹서버가 이 요청을 처리한 후 HTML(Hyper Text Markup Language) 페이지를 사용자에게 다시 전달하여 응답하는 구조로 실행되어 진다. 이를 통해 보안(Security)의 측면도 제고될 수 있다.

2) 비즈니스 계층

비즈니스 계층은 다중 계층 아키텍처에서 새롭게 제시된 개념으로 비즈니스적인 논리를 중앙 집중화하여 여러 클라이언트 응용 프로그램에서 업무 처리에 따른 서비스 재사용이 가능하도록 구현 가능하며, 개발시에 더욱 빠른 속도로 개발할 수 있는 환경을 제공함은 물론 운영 관리시 유지, 보수 관리를 용이하도록 지원한다. 이때 적용할 수 있는 미들웨어로 CORBA, TUXEDO, EJB 등이 있으며, 다중 계층 아키텍처 시스템에서 중요한 부분인 COM+가 적용되는 계층이다[2][3].

3) DBMS 계층

DBMS 계층은 데이터를 효과적으로 활용할 수 있도록 데이터의 저장, 관리 등을 수행하는 계층으로써 데이터를 관리하기 위해 필요한 필수 과정인 데이터의 등록, 수정, 삭제, 조회, 백업 등의 역할을 수행한다.

사용자는 DBMS의 접근을 통해 저장된 정보를 활용하여 통계 자료 산출 및 과거 발생된 사건에 대한 검색 등을 수행한다[4].

3. 다중 계층 아키텍처 환경 구현방안

3.1 구현 요소의 이해

1) COM+

인터넷의 사용이 폭발적으로 증가하면서 기업들도 기간 업무 시스템 중 일부를 웹 환경으로 이전하고자 하는 노력을 시작했다. 이는 워크스테이션급의 성능을 보유하고 있는 고성능 PC와 충분한 대역폭(Bandwidth)을 제공하는 네트워크에 의해 분산 컴퓨팅이 가능하게 되었기 때문이다. 다양한 이기종 컴퓨팅(Distributed Computing) 환경에서 데이터와 응용 프로그램을 분산 운용하면서 프로그램간의 상호 운영성과 사용자에게 분산의 투명성을 제공함과 동시에 원격 시스템 간의 자원 공유, 개방성, 시스템 간의 병렬성, 확장성 등이 보장되어야 한다는 문제가 대두되었다[5][6].

이와 같은 문제를 해결하기 위해 마이크로소프트(Microsoft)사에서는 윈도우 기반의 COM+라는 프레임워크를 개발하여 분산 객체를 운영할 수 있도록 하였다. COM+는 COM(Component Object Model)과 MTS(Microsoft Transaction Server)가 결합한

진보된 프레임워크 또는 컴포넌트 소프트웨어이며, 객체 지향의 장점과 컴포넌트 기반을 내장한 이진 형식의 소프트웨어 조각으로 구성된 상호 운영성 및 로드 밸런싱, 트랜잭션, 보안, 관리 등 다양한 장점을 내장하고 있다[7].

2) 컴포넌트(Component)와 객체(Object)

컴포넌트는 일반적으로 객체 지향 언어로 개발되고 있으나 반드시 그러한 것은 아니다. 그러나 객체는 반드시 객체 지향(Object Oriented) 프로그래밍만 가능하다. 객체 지향 언어로 작성된 한 개의 클래스가 곧 객체의 단위로 볼 수 있다. 또한 한 개의 컴포넌트는 많은 객체를 가질 수 있으며, 여러 객체들이 서로 연동하여 동작하지만 컴포넌트를 사용하는 클라이언트 입장에서 이것은 하나의 컴포넌트로 동작된다.

이에 비해 객체 개념은 컴포넌트 개념으로 적용될 수 있는데 하나의 컴포넌트는 객체가 갖는 특성을 가지므로 한 개의 컴포넌트를 한 개의 객체로 볼 수 있다. 객체는 일반적으로 소스 레벨에서 재사용을 의미하고, 컴포넌트는 바이너리 레벨에서의 재사용을 의미하기에 소스 레벨에서 객체를 수정할 때에는 재컴파일(Recompile)의 과정이 필요하지만 바이너리 레벨에서 컴포넌트를 변경할 때는 재컴파일의 과정이 필요없게 된다[8].

3) 컴포넌트 적용 과정

기존의 CORBA 객체 및 일반적인 객체 지향 프로그래밍을 통해 생성된 객체는 재사용을 표방하였으나 실제 컴포넌트로 사용하기에는 여러 문제점이 제기되었다.

우선 소스 형태의 클래스/객체는 컴파일을 해야 하므로 컴포넌트로 사용하기 어렵고, 객체들 간의 관계가 복잡한 대규모 프로젝트에서는 확장성이 떨어진다. 또한 서브 클래스가 슈퍼 클래스로부터 상속받기 위해서는 슈퍼 클래스의 내부를 분석해야 하므로 클래스의 캡슐화가 보장되지 않으며 확실한 정보 은폐 기능을 제공하지 않는다.

이러한 객체의 한계를 극복하기 위해 새로운 패러다임으로 등장한 것이 컴포넌트에 기반을 둔 소프트웨어 개발 방법론이며, 컴포넌트는 여러 객체로 구성되어 그 내부는 외부에서 볼 수 없도록 캡슐화되어 있고 소스 코드 수준이 아닌 실행 파일 형태로 제공된다.

컴포넌트는 포트(Port)를 통해 외부 시스템과 교류하기 위해 한 개의 컴포넌트를 사용하거나, 또는 여러 컴포넌트를 패키징(Packaging)하여 원하는 애플리케이션을 만들 수 있다. 컴포넌트 또는 패키징된 컴포넌트는 대상 시스템에 배치되어 실제 작업을 수행한다[9].

3.2 구현 방안

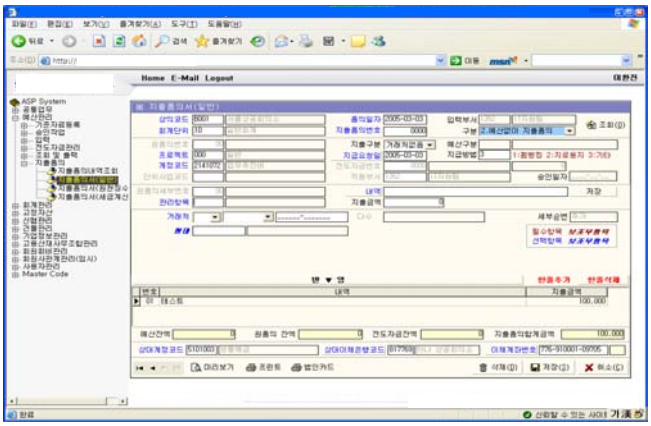
다중 계층 아키텍처 환경 구현 시 클라이언트/비즈니스/데이터베이스 각각의 계층을 15 : 55 : 30 등과 같은 비율로 배분하여 중간 계층인 비즈니스 계층을 최대한 활용할 수 있도록 구현한다. 다만 구현에 따른 여러 가지 환경적인 요소 및 개발 여건에 따라 비율 배분이 달라질 수 있으나 클라이언트 계층과 데이터베이스 계층의 처리 비율이 높지 않다면 적절한 배분으로 구축하더라도 특별한 문제는 발생하지 않는다. 또한 특정 모듈의 재사용성만을 높이기보다는 적절한 배분을 통한 밸런스 조절을 통해 개발 환경의 최적화를 유지하면서 재사용성을 높일 수 있도록 지향해야 한다.

1) 클라이언트 시스템의 응용 프로그램

클라이언트 시스템 계층은 웹을 통해 사용자가 접근하기 쉬운 환경을 제공하고 불요불급한 기능을 배제한 인터페이스로 구현함을 기본으로 하며, 개발언어는 델파이(Delphi)로의 ActiveX-Form 기술을 응용할 수 있다.

클라이언트 시스템과 비즈니스 로직 계층의 연결 시에는 Webconnection, ClientDataSet, DataSource 객체를 통한 비즈니스 계층의 애플리케이션 서버로 통신(웹서버와 HTTP 프로토콜)하며, 물리적으로 동일한 위치의 COM+ 응용 프로그램과 포트로 연동시켜 비즈니스 로직이 적용된 객체와 통신하며, 업무처리 프로세스를 수행 및 처리하도록 구현한다.

ActiveX-Form으로 구현된 프로그램은 브라우저를 통해 압축된 파일인 *.CAB 파일로 제공되며, 사용자의 브라우저를 통해 다운받아 사용자 계층(사용자 컴퓨터)에서 컨트롤 파일인 *.OCX가 레지스트리에 등록되어 업무 처리를 수행할 수 있도록 한다.



(그림 3) 클라이언트 시스템의 응용 프로그램 예시

(그림 3)은 개발툴인 델파이를 활용하여 클라이언트 계층의 ActiveX-Form으로 작성된 웹 응용 프

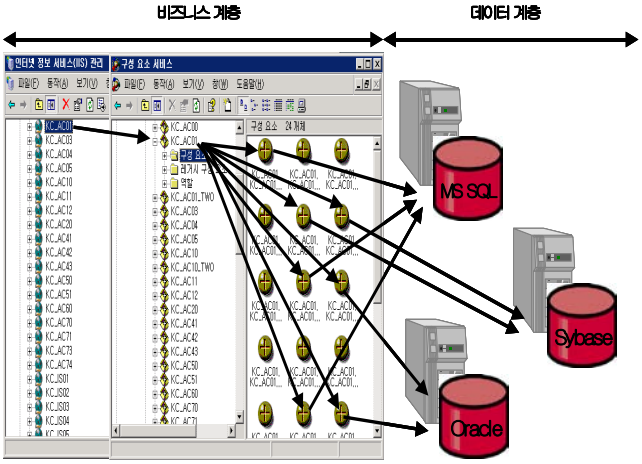
로그램의 예제 화면이다.

2) 비즈니스 계층

비즈니스 계층의 구현 시에는 어떠한 개발 툴로 개발하여도 무방하며, 개발시 고려사항은 업무 처리에 사용되는 기초 코드나 업무 처리 단위 및 재사용이 가능한 프로세스를 분석, 설계하여 사용 단위별로 최적화된 상태를 유지할 수 있도록 분리하여 구성한다. 또한 하나의 객체는 10개 이내의 클래스를 유지하고 각 클래스마다 처리되는 메소드(Method)는 15개 내외로 적용한다. 이는 여러 클라이언트 프로그램의 업무 처리 수행 시에 과부하의 발생을 억제하고 안정된 서비스를 제공하기 위한 일반적인 개발 방안이다.

COM+는 적시 활성화(Just in Time Activation) 기능을 제공하므로 해당 메소드 호출시 로드되었다가 결과를 응답 후 해제하므로 메모리 사용량 최소화 및 처리 속도 향상을 극대화 할 수 있으며, 마이크로소프트 윈도우 기반의 IIS(Internet Information Server)는 클라이언트 응용 프로그램의 요청에 의해 COM+ 응용 프로그램을 호출하고 해당 서비스를 수행한다.

COM+ 객체는 업무 처리 프로세스를 수행하며 비즈니스 계층의 DB 접속 정보를 통하여 DBMS 계층과 통신한다.



(그림 4) 비즈니스 계층과 데이터 계층 상호 작용

(그림 4)는 비즈니스 계층에서 수행되는 COM+ 응용 프로그램을 적용하여 IIS와 COM+ 응용 프로그램의 상호 작용, COM+와 이기종 데이터베이스간의 상호 작용을 구현, 적용한 계층 간의 상호 작용을 도식화한 것이다.

4. 결론

근래의 인프라 스트럭처(Infra Structure)의 발전에 따라 사용자들의 요구사항이 늘어나고 있으며 이를 만족시킬 대안인 웹 기반 다중 계층 아키텍처의 구조가 대두되고 있다.

본 연구에서는 우선 2계층 아키텍처 구조와 다중 계층 아키텍처 구조의 차이점을 비교하여 다중 계층 아키텍처 구조의 상대적 우수성을 검토한 후, 기존 2계층 아키텍처 기반의 시스템을 다중 계층 아키텍처 기반의 시스템으로 전환하기 위한 구현방안을 제시하였다.

다중 계층 아키텍처 기반의 시스템은 2계층 아키텍처 기반의 시스템에 비해 상대적으로 확장 용이성 및 개발 생산성 제고, 네트워크 트래픽으로 인한 성능 저하 최소화, 유지 보수의 용이성, 서버 시스템으로부터의 빠른 응답 등의 장점을 보이고 있다.

참고문헌

[1] 류형규, 이순천, 류시원, 신성호, 'UML 기반 객체지향 클라이언트/서버 구축', 홍릉과학출판사, pp.33-38, 2000.

[2] 권원상, 성진수, 이철성, 'IT 백두대간, 닷넷 프로그래밍 C#, VB.NET, ASP.NET', 한빛미디어, pp.35-38, pp.626-636, 2002.

[3] 백운기, 한상홍, 박준후, 'Delphi 5 Contact', 대림, pp.1316-1346, 2000.

[4] 정선환, 'Object 지향 이론', 대광서림, pp.9-12, 1994.

[5] Robert Orfali, Dan Harkey, 'Client/Server Programming with Java and Corba', Wiley Computer, pp.65-115.

[6] Seigel, Jon. 'Fundamentals and Programming', New York : John Wiley & Sons, 1996.

[7] David S. Platt, 'Microsoft Understanding COM+ Revised', Microsoft Press, 2000.

[8] 윤은영, 윤용익, 윤석환, '컴포넌트 기반 미들웨어 기술', 정보처리 제8권 제5호, p.39, 2001.

[9] 전게서, p.40.