□사례발표□

실무 사례 중심의 클라이언트/서버 구축 방법

홍 정 화†

◈ 목 차 ◈

1. 서 론

컴퓨터의 기술은 비약적인 발전을 거듭하여 왔다. 시대별로 보면 60년대 일괄처리(batch processing) 시대를 거쳐 70년대 시분할 처리(time sharing)가 가능하여 지면서 한 컴퓨터에 여러 사용자가 사용 가능하게 되었고, 80년대는 개인용 컴퓨터가 비약적인 성능향상을 거듭하게 되었다. 특히 인텔의 CPU칩은 8086에서 시작하여 펜티엄까지 거의 2-3년을 주기로 4배 이상의 성능이 향상된 마이크로프로세서의 신제품이 나왔고, 이와 함께 점점 개인용 컴퓨터는 더미(dummy) 형태에서 지능형(intelligent) 컴퓨터를 각 개인이 소유하게 되었다. 또한 개방형 시스템인 UNIX에 RISC칩, 병렬처리의 도래는 기존 IBM 대형 시스템에서 가능하던 업무처리 환경을 보다 값싼 시스템으로 구축 가능하게 됨으로써 클라이언트/서버 및 분산처리 환경이 가능하게 되었다. 궁극적으로 이와 같은 컴퓨터 기술의 발전을 바탕으로 90년대에는 다운 사이징을 모토로 한 클라이언트/서버 및 분산처리 시스템의 시대가 시작 구현 되었다. 물론 이와 같이 클라이언트/서버 환경이 도래하게 된 원인은 기술적인 측면 뿐 아니라 IBM에 의해 주도되던 중앙처리 방식의 폐쇄된 컴퓨터 처리 환경으로 인하여 한 회사의 솔루션에 종속되어 가는 운영상에 문제, 비용문제 등이 복합적으로 결부된 결과라 할 수 있다.

클라이언트/서버 컴퓨팅이란 지능형 컴퓨터를 가진 사용자가 네트워크 상에서 분산 자원을 공유할 있는 소프트웨어 기반 구조(software-based architecture)라고 가트너 그룹에서는 정의하고 있다. 여기에는 클라이언트/서버를 구성하는 4가지 기본 요소가 존재하게 된다. 단말 사용자가 접촉하는 지능형 개인 컴퓨터, 사용자의 요청을 서비스하는 서버, 클라이언트 서버간을 연결하는 네트워크, 이 세가지를 구현할 수 있는 소프트웨어로 구성 되고 특히 SQL 언어는 클라이언트에서부터 서버에서 구동 중인 DBMS(Database Management System)로 서비스 요청을 보내게 되는 언어이다.

클라이언트/서버를 채택함으로써 얻을 수 있는 이득은 첫째, 컴퓨터와 소프트웨어의 계속적인 가

† 정회원 : 한국오라클(주) 기술본부 제조전문팀 차장

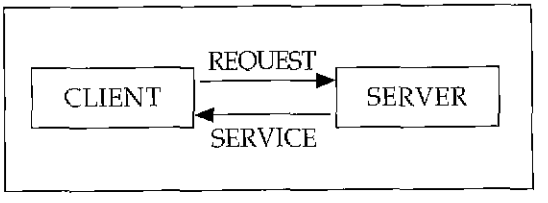
격 하락으로 인한 비용 절감으로 이는 개방형 시스템이 가져다 주는 혜택이라 할 수 있다. 둘째, 개방형 산업 표준을 준수 함으로써 한 회사의 하드웨어나 소프트웨어에 종속되지 않고 자신의 환경에 최적인 각 컴퍼넌트를 구성할 수 있다. 셋째, 클라이언트/서버는 GUI를 100% 활용함으로 시스템 사용에 대한 교육의 감소, 확산의 용이성 및 데이터에 대한 접근의 쉬워짐으로 인한 데이터 활용이 더욱 커진다. 마지막으로 작업을 분산함으로서 최종 사용자의 업무 능률을 향상 시키고 전산실 요원에게 보다 가치 있는 미션크리티칼(mission-critical) 업무에 집중하도록 함으로써 전체적인 업무 효율을 높일 수 있다.

본 고에서는 21세기를 눈앞에 둔 시점에서 기존 90년대를 풍미하던 기존 클라이언트/서버 환경을 실무구축 사례 중심으로 분석하여 보고 점점 과속 급변하는 컴퓨터 기술의 조류 속에서 클라이언트/서버 환경의 발전 방향을 점쳐 보려 한다.

본 고는 크게 4장으로 구분되어 있고, 1장 서론 부분에서는 시대별 컴퓨터 시스템의 발전 및 클라이언트/서버 환경 탄생 배경 장점 및 본 고에서 논하고자 하는 범위를 지정하고, 2장에서 그 동안 꾸준히 논의 되어온 클라이언트/서버 구축 방법론인 1 계층, 2계층, 3계층 등 각기 방식의 차이점 및 장단점 분석하여 보도록 하고, 3장에서는 실제 구축 운영중인 대표적인 3 가지 클라이언트/서버 구축 사례를 분석하여 효과적인 클라이언트/서버 시스템 구축 방법과 허 와 실을 분석하고, 끝으로 4 장에서는 향후 클라이언트/서버의 발전 방향의 대안인 네트워크 컴퓨팅에 대한 소개 및 그 타당성을 논하여 보고 결론을 내려 보도록 하겠다.

2. 클라이언트/서버 구축 방법론

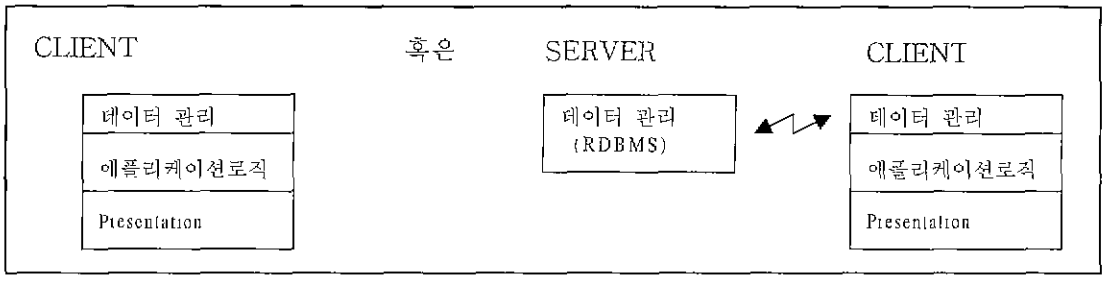
클라이언트/서버 구조란 아래 그림처럼 기본적으로 클라이언트 프로세스와 서버 프로세스 간의 통신으로 업무를 처리하는 방식이다. 즉 요구조건을 내세우는 클라이언트는 개인용 컴퓨터에 존재하고 이에 대한 서비스를 제공하는 서버는 중형 이상의 컴퓨터 존재하여 클라이언트의 요구를 해결해 줌과 동시에 확보된 개인용 컴퓨터의 성능을 이용하여 중앙처리 방식에서는 상상할 수 없던 GUI를 제공함으로서 사용자가 손쉽게 시스템을 운영할 수 있을 뿐 아니라 개인에 대한 처리 권한을 최대한 부여하여 필요에 따라 독립적으로 데이터를 가공 재생산하여 결국은 중앙 처리로 인한 처리 시간의 지연을 줄일 수 있게 한다.



(그림 2-1) 클라이언트/서버 처리

위의 설명처럼 아무리 클라이언트/서버 환경에 장점이 존재한다 할지라도 반드시 간과 하지 않아야 될 사항은 클라이언트/서버 간의 통신에 의해 처리되는 방식으로서 만일 이들 간에 과다한 요구(request)와 서비스(service)가 발생한다면 네트워크 체증으로 소기의 목적을 달성하기도 전에 운영할 수 없는 시스템이 될 수 있다.

클라이언트/서버 구축 시에는 여러 가지 컴포넌트가 존재하여 시스템을 구축 하게 되는데 첫째로 데이터를 저장 서비스하는 DBMS(Database Management System)으로 Oracle, Sybase, Informix, Ingress, DB2등의 관계형 데이터 베이스가 주종을 이루고, 둘째로 클라이언트 어플리케이션을 구축 표현하여 주는 클라이언트 툴로서 오라클사의 developer/2000, 사이베이스사의 Power builder,



(그림 2-2) 1 계층 처리 구성도

마이크로사의 Visual Basic, Delphi등이 대표적인 클라이언트 툴이다. 작업 분야별로 클라이언트/서버 환경을 구성하는 요소를 구분하면 데이터 요구에 대한 서비스를 제공하는 데이터 관리 부분과 최종 사용자에게 화면을 제공하는 표현 부분과 어플리케이션 로직 등 3부분 나눌 수 있으며, 관계형 데이터베이스를 데이터 관리 부분으로 사용할 경우 SQL(Structured Query Language)을 이용하여 표현 부분인 클라이언트에 원하는 데이터를 보내 주게 된다. 이때 데이터 관리 부분, 어플리케이션 로직 부분, 표현 부분의 물리적 위치와 분리 여부에 따라 1 계층, 2 계층, 3 계층, 2 계층 3계층 혼합형으로 나뉘어지고 각자 발생되는 트랜잭션의 특징 및 업무성격에 따라 적용 범위가 결정 된다.

2장에서는 각 계층별 구성 방법과 장단점에 대해서 알아 보도록 하겠다.

2.1 1 계층 구조

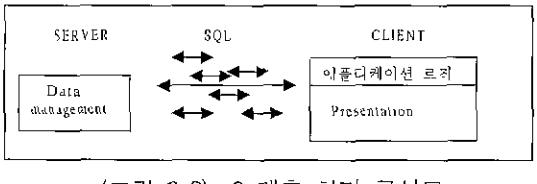
1 계층 구조는 클라이언트에 표현 부분 및 어플리케이션 로직, 데이터 관리 부분이 존재하는 형태로서 아래 (그림 2-2)처럼 표현된다.

(그림2-2)와 같은 구조는 일반적인 클라이언트/서버 환경에서는 극히 제한적으로 나타나고 실지로 현업 환경에 적용하기는 아무리 클라이언트 성능이 좋아지더라도 클라이언트의 한계로 인하여 현실적으로 불가능한 구조이다. 뿐만 아니라 각 클라이언트가 데이터를 저장 관리함으로써 데이터에 대한 보안 활용도 측면에서 많은 문제가 발생 할 수 있다. 물론 하나 사용자만이 구축 사용하고자 하는 경우에 문제가 없지만 여러 사용자가 동시에 사용하는 클라이언트/서버 환경에는 부적합하다 할 수 있다.

2.2 2-계층 구조

가장 많이 적용되는 구조로서 구현이 쉽다는 장점이 있다. 즉 기존 3GL 프로그래밍 방식과 비슷하게 클라이언트 프로그래밍함으로써 구현할 수 있다. 서버에는 데이터를 관리하는 DBMS 등이 존재하여 데이터에 대한 서비스를 담당하고 클라이언트에는 모든 어플리케이션 로직과 프리젠테이션 부분이 존재하는 구조이다. 이는 클라이언트에 모든 어플리케이션 로직이 구현 됨으로써 한 단위 업무를 처리할 때 클라이언트와 서버 간에 많은 네트워크 및 자원에 대한 부담이 가중됨으로 많은 사용자 동시에 사용하기는 구조적인 제약이 따르게 된다.

아래 (그림 2-3)은 정형적인 2-계층 방식을 표현한 그림이다.



(그림 2-3) 2-계층 처리 구성도

2-계층 구조에서 네트워크 체증이 많아지는 이유는 필드 검증(validation), 필드 간의 계산 등 클라이언트 자체가 처리해야 효과적인 부분 뿐 아니라 DBMS 차체가 처리해야 효과적인 비즈니스 측면의 데이터 처리 중심 로직 까지 네트워크에 의존하여 모든 어플리케이션을 클라이언트에서 코드 처리 함으로서 발생하게 된다. 그러나 클라이언트에서 작성하는 프로그램밍 작업은 기존 호스트에서 작성 하던 프로그램에 비해 훨씬 쉽게 구현 가능하게 하는 장점도 있다.

이와 같은 구조는 중소 규모의 20-30명 정도 사용하는 환경에 적합하다 할 수 있다. 바로 2-3절에서 논의하게 되겠지만 비교적 적은 사용자가 사용하는 시스템에 긴 개발 기간과 위험 부담을 들이는 것보다는 빠른 시간 내에 개발 활용하는 것이 비용 측면에 효과적이고 현실적이다.

2.3 3-계층 구조

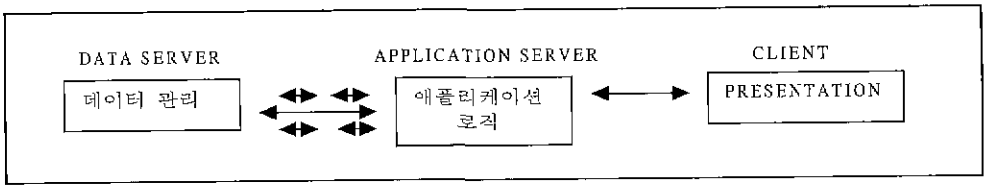
위 2-2절에서 논의 된 것처럼 2-계층 구조는 네트워크 및 클라이언트 자원의 제약으로 인해 많은 사용자가 사용하기에는 많은 문제가 따른다. 특히 네트워크 환경이 다양하게 분포 되어 있는 대기업의 기간 시스템이나 금융사의 예금 입출금 업무처럼 동시에 같은 어플리케이션 로직이 수행되는 환경에서는 사용 불가능 하다. 따라서 이와 같은 단점을 보완하기 위해 아래 (그림 2-4)과 같이 데이터 관리, 애플리케이션 로직, 프리젠테이션을 분리하는 3-계층 구조를 가져 감으로 클라이언트와 서버 간의 절대적인 네트워크 체증을 줄이게 하여 많은 사용자를 서비스하게 한다.

전형적인 3-계층 구조는 클라이언트가 단지 프리젠테이션 기능만을 수행함으로써 클라이언트가 더미(dummy)화 함으로서 클라이언트 화면의 필드에 잘못된 변경이 발생하게 되는 경우 화면 단위의 블록 모드 RPC(Remote Procedure Call) 방식으로 수행 되므로 모든 화면의 필드가 입력 될 때까지 에러를 기다리게 된다. 시스템 개발 측면에서 보면 운영하고자 하는 모든 로직이 사전에 작성 구축하여 어플리케이션 서버에 올려져야 함으로써 기본적으로 클라이언트 툴에서 4GL을 사용하여 쉽게 구현할 수 있는 로직까지 3GL을 사용하여 구축되어야 함으로써 상대적으로 2-계층에 비해 개발 생산성이 크게 떨어지게 된다. 뿐만 아니라 자주 반복되는 로직이 아닐 경우 과다한 어플리케이션이 서버에 운영되어야 함으로서 경우에 따라 어플리케이션 서버에 대한 활용도가 떨어지게 된다.

3-계층을 구축하기 위한 필수 요소로서 top-end, tuexido, encina와 같은 TP-모니터을 사용하게 되는데 이와 같은 미들웨어에 대한 분석을 하여보면 아래와 같다.

기존 중앙처리 방식의 운영 환경에서 개방형 클라이언트/서버 환경을 접목하기 위해 UNIX 솔루션을 도입하게 되었고, 이에 따라 데이터베이스 간의 연동 및 네트워크 프로토콜을 연동할 수 있는 해결책으로 미들웨어인 TP-모니터가 필요하게 되었다.

미들웨어란 네트워크 프로토콜에 대한 인터페이스를 포함하고 각종 어플리케이션에 대한 표준

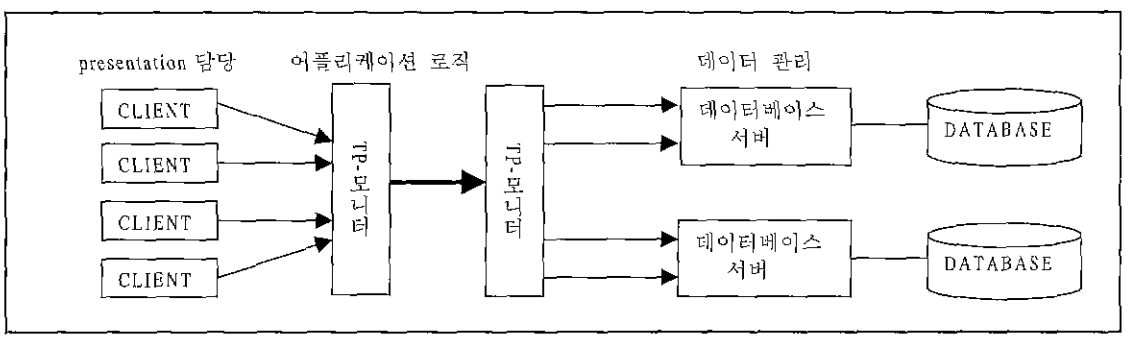


(그림 2-4) 3-계층 처리 구성도

화된 인터페이스를 제공하는 일종의 API로서 대부분의 클라이언트/서버 환경에서는 SQL을 기본으로 하는 관계형 데이터 베이스가 사용되기 때문에 대개의 TP-모니터는 이질적인 네트워크와 데이터베이스 환경에서 데이터를 투명하게 연결해 주는 역할 뿐 아니라 클라이언트/서버가 안고 있는 네트워크 집중 및 로드 분산을 수행하도록 함으로서 3-계층 구조를 구축하는데 있어서 주요 구성 요소가 될 수 있다.

기본적으로 TP-모니터는 어플리케이션 로직을 분리시켜 클라이언트와 서버간의 네트워크 체증을 줄임으로서 3-계층의 장점을 살릴 뿐 아니라, 2단계 커밋(2-phase commit)을 제공하여 트랜잭션의 무결성을 보장하고, 전체적으로 서버에 대한 부하가 균등 하도록 하여 OLTP 업무의 처리량이 최대로 유지되게 하는 로드 분산 기능, 일정한 응답 시간을 보장하는 스케쥴링 기능, 이기종 데이터베이스 간의 연결 기능 등으로 인해 개방형 클라이언트/서버 시스템으로 전이 되는 과정에서 중요한 요소가 될 수 있지만, 반면 TP-모니터 도입에 따른 비용도 만만치 않을 뿐 아니라 대부분의 어플리케이션 로직이 개발 툴을 이용한 생산성을 보장 받을 수 없음으로 인한 제약 및 개발 기간의 연장으로 인한 비용 또한 구입 비용 못지 않다. 또한 극단적인 OLTP 업무에는 많은 효율을 거둘 수 있을 지 모르지만 일괄처리 작업이나 분석을 요하는 긴 트랜잭션에는 오히려 성능 향상에 심각한 장애를 가져올 수 있다. 실제로 지난 수년간 단지 업무의 특성은 고려하지 않고 막연히 벤더의 말에 따라 성능 측면만 고려하여 TP-모니터를 도입하여 실패한 사례가 무척 많았다. 적어도 온라인 업무가 100 TPS(Transaction Per Second) 이상인 규모의 회사 업무에 적합하다 할 수 있다.

실지 TP-모니터의 도입하여 실패 할 경우 많은 피해를 예상할 수 있는데 이는 개발 방법부터 정통적인 클라이언트/서버와 너무나 상이하여 실패에 대한 복구가 사실상 어렵다. 실지 클라이언트/서버 환경을 사용하면 성능 부분에 많은 문제가 발생하는 원인은 네트워크과 같은 물리적이거나 클라이언트/서버 구조상의 문제보다는 집합 개념을 갖고있는 클라이언트/서버 처리 언어인 SQL을 부적절한 사용에 기인 하는 경우가 보다 많으므로 성능 향상은 TP-모니터라는 등식이 성립 되는 것은 아니다. 따라서 2-계층으로 클라이언트/서버 구조를 가져 갈 것인가 3-계층으로 운영할 것인가 TP-모니터를 사용 할 것인가는 업무의 성격, 어플리케이션 튜닝 등을 검토 및 수행 후 전문가의 조언에 따라 신중하게 고려하는 것이 효과적일 것이다.

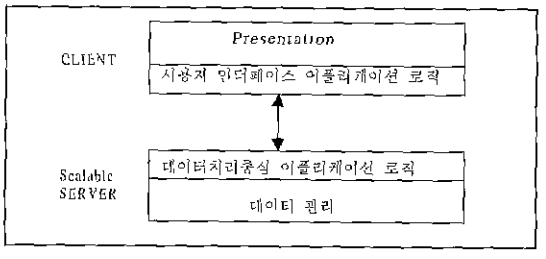


(그림 2-5) TP-모니터를 이용한 3-계층 처리

2.4 2-계층와 3-계층이 혼합된 클라이언트/서버 구조

여기 2-4절 논의 되는 혼합형 클라이언트/서버 구조는 2-계층의 생산성과 3-계층의 확장성을 동시에 얻을 수 있는 또 다른 형태의 클라이언트/서버 구조로서 충분한 검토의 효과가 있다 할 수 있겠다. 구조적으로는 2-계층을 따르지만 어플리케이션 로직을 나누어 수행할 수 있는 툴의 특성을 이용한 방법이라 하겠다.

즉 네트워크의 사용을 최대로 줄이면서 클라이언트 자원을 최대로 늘리는 또 다른 방안으로서 어플리케이션 로직을 클라이언트에서 처리하는 것이 효과적인 필드 확인(validation) 부분과 서버 쪽에서 수행하는 것이 효과적인 데이터 처리 중심 어플리케이션으로 나누어 각 어플리케이션 로직을 클라이언트와 서버로 나누어 처리하는 방식으로 아래 (그림 2-6)으로 표현 될 수 있다.



(그림 2-6)

이는 바로 클라이언트 툴에서 원하는 어플리케이션을 작성함으로써 2-계층의 장점인 개발 생산성을 높이고, 클라이언트에서 개발된 모듈중 서버에서 처리하는 것이 효과적인 데이터처리 중심의 부분은 서버에서 처리하게 하는 방식으로 어플리케이션 로직의 작업 특성에 따라 분할 처리하는 방식이다.

이와 같은 구조는 오라클사에서 스마트 클라이언트(Smart Client) 이름하에 ERP에 적용하는 방법론으로서 사용자 인터페이스와 관련된 부분은 모두 클라이언트 컴퓨터에서 처리하고 자료 중심 처리는 서버 컴퓨터에서 담당 함으로서 3-계층 구조에서 클라이언트가 아무런 어플리케이션 로직을 처리하지 못하는 단점과 중간층(Middle tier)을 위한 추가적 투자를 줄일 수 있을 뿐 아니라, 순수한 2-계층의 패트 클라이언트(Fat-client) 구조에서 모든 응용 프로그램을 수행함으로써 발생할 수 있는 과도한 네트워크 부하 줄이는 양자의 효과를 거둘 수 있다. 이러한 아키텍처는 클라이언트와 서버 모두에게 충분한 강점을 보유한다. 클라이언트 컴퓨터는 그래픽과 마우스의 지원을 제공해 줌으로써 데이터 프리젠테이션을 최적화할 수 있게 된다. 마찬가지로 서버 컴퓨터는 많은 프로세싱 능력과 메모리, 그리고 충분한 디스크 스페이스를 제공해 줌으로써 데이터 프로세싱 저장에 있어 최적화된 기능을 가지게 되는 것이다. 그림 2-6에서 보여 주듯이 클라이언트 상에서 사용자 인터페이스 프로세싱은 프리젠테이션을 포함하고 있음을 알 수 있다. 또한 프리젠테이션을 지원하기 위해 필요한 어플리케이션 로직을 나타내주고 있다. 프리젠테이션은 디스플레이 관리와 네비게이션으로 구성되어 있으며 오라클의 Developer/2000과 같은 툴을 사용함으로써 실행될 수 있다. 오라클 Developer/2000은 데이터베이스 내의 테이블을 입력 수정 삭제 조회하는 FORMS와 수치 데이터를 도표화 할 수 있는 GRAPHICS등으로 구성되어 있는데 위의 예제를 가장 잘 표현해 줄 수 있는 대표적인 툴이라 하겠다. 또한 그림 2-6에서 나타내듯이 서버상에서 수행되는 데이터 처리 중심의 데이터 지향 어플리케이션 프로세싱을 들 수 있는데 이는 업무 규칙 프로세싱을 위한 RDBMS 내의 내장 어플리케이션을 사용해 실행 될 수 있다. 내장 어플리케이션은 Embeded SQL를 사용하여 작성할 수 도 있지만 생산성이

문제가 되므로 오라클 4GL언어인 PL/SQL을 사용함으로써 손쉬운 작성과 클라이언트 툴과 서버 상에서 동시에 적용 가능함으로써 툴에서 프로그램을 작성하여 서버에 내장시킴으로써 클라이언트에서 작성한 프로그램을 필요에 따라 서버와 클라이언트로 적절히 분할 할 수 있다.

3. 사례 연구

위의 2장에서는 클라이언트/서버의 대표적인 구축 방법론과 이에 대한 장단점을 분석하여 보았다. 이번 3장에서는 2장의 이론적 배경을 기반으로 실제 구축한 3가지 사례를 중심으로 클라이언트/서버의 효율적 구축 방법을 설명하여 보겠다. 첫번째 사례는 조회 중심의 시스템으로 약 15년 이상 IBM 호스트에서 시스템을 구축, 사용한 시스템의 2-계층을 이용한 클라이언트 서버 적용을 설명하고, 두 번째 사례에서는 사용 유저가 15000명에 분(minute)당 동시 사용자가 1500 이상인 제조 업체에서 그룹웨어를 3-계층 클라이언트/서버 환경으로 효과적으로 적용한 사례을 설명하고, 세 번째 사례에서는 약 4000의 직원을 관리하는 인사/급여/회계 시스템에서 클라이언트/서버를 적용 실패한 사례에 대해 알아보겠다.

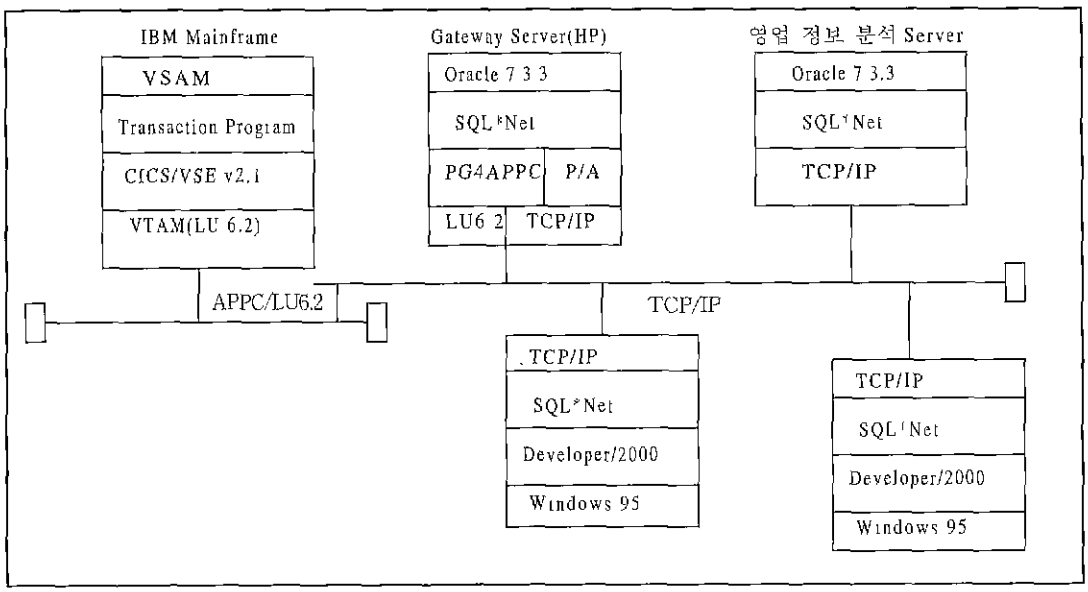
3.1 사례 1 영업 정보 분석 시스템

사례 1에서 소개하는 회사는 제약/음료 분야 회사로서 전산 분야에 일찍부터 투자하여 전국적인 영업망에 300여명의 영업사원이 IBM 중심의 시스템에서 영업 실적 데이터를 입력하고 분석하는 데에는 전혀 문제없이 사용하고 있었다. 그러나 신규 회사의 참여로 생산하고 있는 제품에 대한 경쟁이 치열하여 짐으로서 새로운 제품의 개발 계획, 생산 계획 수량 시기, 심지어 음료 분야와 날씨와의 관계, 제약 제품과 광고와의 관계 등이 제품 탄생에서 소멸되는 전단계에서 정보가 필요하게 되었다.

이는 기존에 생각할 수 없었던 부분이 회사의 생존 발전을 위해 중요한 이슈로 등장하게 되었다는 것을 의미한다. 이에 대한 작업으로 이 회사는 치열한 경쟁 환경에서는 생산에서 판매까지 업무에 대한 프로세싱의 재구축이 필요하다고 결론 내림과 동시에 기존 기업 내의 정보 인프라로는 필요한 시기에 정확한 정보 제공이 어렵다고 결론 내리게 되었다. 특히 영업정보 데이터에 대한 시각적인 분석을 통하여 빠른 의사 결정을 내릴 수 있는 시스템의 추가가 시급하게 되었다. 기존에 운영계 시스템의 근간인 IBM으로는 직관적으로 정보를 제공할 수 있는 GUI 표현에 한계가 있다고 판단하고, 개방형 시스템으로의 추가 인프라 구축이 절실하게 되었으며 궁극적으로는 전 시스템을 개방형으로 전이할 수 있는 기반을 제공하는 시스템 환경이 필요하게 되었다.

위의 구성도는 크게 LU6.2을 사용하는 IBM과 TCP/IP을 사용하는 영업 정보 분석 시스템간의 데이터 연동을 위해 Gateway 서버를 두고, IBM 호스트의 VSAM으로 구축된 영업 실적 정보를 실시간 또는 주기적으로 Unix 환경의 영업정보 분석 시스템의 오라클 데이터베이스에 반영하여 실적 분석에 활용한다.

영업 정보 분석 시스템의 발생 트랜잭션의 특징은 시점별 조직별 제품별 분석하는 조회 중심의 쿼리(query)가 주로 발생하므로 TP-모니터와 같은 미들웨어는 채택하지 않고 SQL*Net을 통하여 직접 서버에서 데이터를 검색하여 가져오고 이와 같은 어플리케이션은 생산성이 높은 developer/ 2000을 사용하여 약 2개월의 기간에 거쳐 업무분석, 프로토타이핑(prototying), 개발까지 총 30여 본의 프로그램 개발을 3명의 인원으로 이룰 수 있었다.



(그림 3-1) 사례 1 시스템 구성도

비록 프로그램 개수는 적지만 이는 IBM에서 개발 할 경우 약 150이상의 프로그램 본 수로서 클라이언트 툴의 GUI 장점을 충분이 활용하여 하나의 프로그램이 주제별 기능을 복합적이고 효율적으로 처리하게 함과 더불어 사용이 쉽도록 하는 성과를 거두었다. 기존 IBM에서 운영 시에는 비정기적으로 발생하는 자료 보고서를 전산실에 신청하여 적어도 5-6일 기간이 걸려서 원하는 보고서를 현업이 이용할 수 있었으나 원하는 데이터를 유연하게 GUI 화면에서 조회하여 EXCEL에 다운로드하여 사용자가 가공하여 사용함으로써 전산실에 요구하던 전표 횟수를 크게 줄일 수 있었다. 물론 이는 사용자가 EXCEL와 같은 툴을 새로이 익혀야 한다는 부담은 늘지만 훨씬 큰 업무에 생산성을 높일 수 있다. 궁극적으로 이는 클라이언트 환경의 GUI 어플리케이션의 유연성 때문에 가능한 것이다.

이 사례 시스템의 특징은 클라이언트/서버로의 성급한 전이 보다는 기존 IBM 투자를 최대한 보호하면서 전형적으로 GUI를 필요로 하는 부분만 우선적으로 클라이언트/서버 환경을 도입함으로써 위험 부담을 줄이고, 게이트웨이(Gateway)을 이용하여 실시간으로 IBM VSAM 데이터를 이용함으로써 데이터에 대한 통합도 이루었다. 시스템 환경 구축에 있어도 비교적 사용자가 많은 시스템임에도 복잡한 조회 중심의 트랜잭션 특성에 맞게 2-계층 장점을 최대한 활용함과 동시에 처리 속도 문제를 시스템 구조 보다는 어플리케이션 내의 SQL 튜닝으로 많은 효과 봄으로서 짧은 시간 내에 성공적으로 시스템을 오픈 사용할 수 있었다

3.2 사례2 : 그룹웨어

사례 2에서 설명할 회사는 예상 사용자가 15000명이 넘는 대규모 사업장으로서 1970년대에 조선사업을 시작하여 프랜트, 해양, 엔진, 중전기, 중장비등 7개 사업부를 거느린 종업원 27000명의 회사로서 사무자동화를 통한 기업 생산성 향상이

절실히 요구 되고 있었다. 특히 사업장간의 전산 환경 불일치, 공문서 결재, 기존 시스템 조작의 불편 등으로 클라이언트/서버 환경의 그룹웨어 도입이 절실이 요구 되었다.

특히 예상 사용자가 적어도 15000명이 넘으므로 가장 중요하게 검토해야 할 부분은 안정성이었다. 이런 점에서 기존의 2-계층 방식의 클라이언트/서버 방식을 보완한 3-계층 방식이 적절하다고 판단하게 되었다.

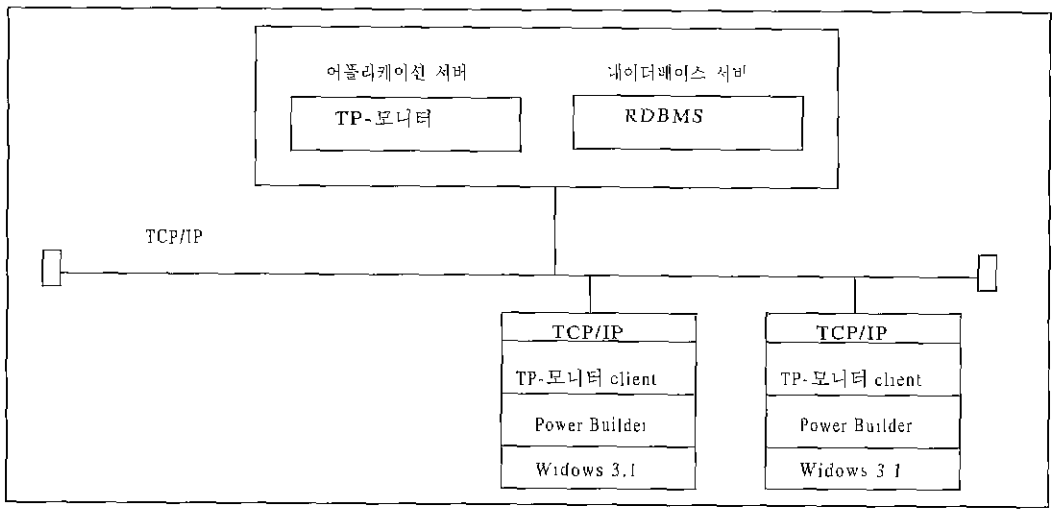
클라이언트에서 요구하는 작업이 바로 채택된 오라클 데이터베이스 서버로 바로 통신하지 않고 먼저 어플리케이션 서버인 그룹웨어 오피스로 요구를 하여 어플리케이션 서버와 데이터베이스 서버가 통신하여 수행된 결과를 클라이언트에 보냄으로써 클라이언트와의 직접 통신으로 인한 접속 한계를 줄이도록 기본적인 클라이언트/서버 구조를 가져갔다. 특이할 만한 사항은 그룹웨어에서 발생할 수 있는 트랜젼션 특징을 잘 대응한 접속관리자 서버를 따로 구축한 사실이다. 데이터베이스 서버 입장에서 보면 비록 부담이 적은 작업이지만 일반적인 업무와는 비교할 수 없을 정도 짧은 시간에 많은 접속이 요구 된다는 사실이다. 이는 DBMS 내의 접속 자체가 부담이 되므로 접속관리 서버를 구현하여 요청이 없는 사용자는 일시적으로 드롭하여 더 많은 사용자를 지원하는 기술을 이용한 것이 UNIX 단일 서버에서 1500여 동시 사용자를 서비스가 가능하도록 하였다. 또한 유지 보수를 위한 배치성 작업(구성원의 직위에 따른 문서 권한 등)은 오라클의 프로시듀얼 언어인 PL/SQL을 이용하여 어플리케이션 서버를 통하지 않고 데이터베이스 서버가 작업이 없는 시간대에 직접 처리하도록 함으로써 시스템을 이용을 극대화 하였다.

그러나 이 시스템의 성공적인 오픈은 순탄치만은 않았다. 약 3개월의 튜닝 기간을 거치고서야 4-5초의 응답 시간을 가질 수 있었다. 여기서 얻을 수 있는 교훈은 비록 3-계층 구조를 가지고 있다 할지라도 클라이언트와 서버간의 통신이 되는 주체는 SQL이고 이에 대한 튜닝을 소홀이 한다면 아무리 좋은 클라이언트/서버 구조를 갖고 있다 할지라도 이는 현실을 외면한 이상이라 할 수 있다. 이와 같이 많은 사용자가 동시에 사용한다는 것은 자주 사용되는 SQL문이 존재하게 되고 이와 같은 SQL문은 정밀한 튜닝이 필요하다는 지극히 당연한 사실이다. 더불어 효과적인 인덱스 설정 및 기타 오라클 파라메터 튜닝, 데드락 감지 및 해결 등도 함께 병행되어야 할 중요한 작업으로 인식 되었다. 이 그룹웨어 시스템의 성공적인 구축은 발생 가능한 트랜잭션의 특징 및 빈도를 파악하여 이에 대비한 적합한 구조를 가져 간 것이 밑거름이 되었다 할 수 있다.

3.3 사례 3 : 인사/급여/교육

사례 3에서 소개하는 회사는 국내 굴지의 대기업 계열사인 인원 4000명 회사로서 기존 COBOL 기반의 문자 형태의 시스템 환경을 GUI 환경으로 전환함으로써 업무 효율을 높이고, 기존 호스트 중심의 시스템에 비효율적 과다 투자를 개방형 시스템인 클라이언트/서버 환경으로 옮김으로 미래 기술에 대한 대비 및 투자 효과를 높이기 위해 빠른 시간 내에 새로운 시스템을 구축하고자 하였다. 특히 이 사례 시스템이 구축을 시작한 시기는 이 그룹 내의 전 계열사간의 클라이언트/서버 신기술 도입이 경쟁으로 이루어진 시기로서 구축 하고자 하는 시스템의 특성을 파악 자신의 업무 환경에 맞는 시스템을 구축하기 보다는 이론적으로 선진적이고 타 계열사와 차별화 할 될 수 있는 전산 업무 환경 구축이 우선시 되었다.

여기 사례는 업무 환경에 맞지 않는 3-계층 클라이언트/서버 환경 도입에 따른 실패 사례로서 그 이유를 검증해 보고자 한다.



(그림 3-3) 사례 3 시스템 구성도

특성상 인사 업무는 많은 트랜젝션이 발생하기 보다는 시각적인 분석 및 통계가 필요한 업무라 하겠다. 예를 들면 올해 진급 대상자, 한 사원의 입사 이래 담당업무, 퇴직자 분석 등을 들 수 있겠다. 시스템을 구축 하고자 하는 대상 인원은 비교적 많은 4000명이라 할지 라도 실질적으로 이 시스템을 활용할 대상은 인사 부서 직원과 간부 관리자 등 50명을 넘지 않으며 신규 입사자도 년에 200명을 넘지 않으므로 추가적으로 발생할 수 있는 데이터의 양도 미루어 짐작 할 수 있다. 급여 업무는 인사정보를 기반으로 급여를 지급하기 위해 월에 한번 주로 긴 배치 작업을 통해 급여 명세서를 발행하는 전형적인 배치 업무이고, 교육 업무는 해당 사원이 수료한 교육 과정에 대해 평가하고 조회하는 업무이다. 이들 3가지 업무의 특징은 매일 반복적으로 많은 사용자에 대해 빠른 응답하는데 주안점을 두는 것 보다는 업무 성격상 여러 가지 다양한 분석이 가능하도록 데이터 양에 비해 많은 어플리케이션이 존재하게 된다. 그러나 이 회사는 클라이언트/서버의 단점만 부각하여 3-계층 구조를 가져 감으로써 많은 모듈을 어려운 C 언어를 이용 작성하여 어플리케이션 서버에 등록하였다. 특히 급여 부분의 작업은 심각한 응답 속도 저하로 인하여 어플리케이션을 다시 작성하여 데이터베이스 서버에서 처리 하도록 재 구축 하였다. 이는 제대로 업무 특성 분석이 이루어 지지않아 많은 비용과 시간을 들이고도 클라이언트/서버의 특징을 충분히 활용하지 못하고 오히려 시스템만 복잡해지는 결과를 초래 하였다.

4. 결 론

지금까지 클라이언트/서버의 효과적인 구축 방법을 실무 사례 중심으로 알아 보았다. 한 시스템을 성공적으로 구축한다는 것은 적은 투자 비용으로 빠른 응답 속도을 얻으면서 기존 시스템의 투자를 최대로 보장하는 것이다. 따라서 클라이언트/서버를 어떤 구조로 구성 하였는가 어떤 구조가 이론적 우위에 있는가 라는 이분론적인 접근 보다는 현재 운영중인 전산 시스템의 현황과 새롭게 도입하고자 하는 시스템의 작업 특성

에 따라 자신에 맞는 클라이언트/서버 구조를 가져가야 겠다. 기업 전산 환경이란 기업의 성패를 좌우하는 지극히 현실적인 문제로서 클라이언트/서버 구축 방법론 접근도 현실적인 접근이 필요하다 하겠다.

돌이켜 보면 80년대부터 컴퓨터 업계는 그랙픽 사용자 인터페이스을 지원하는 개인용 컴퓨터인 클라이언트, 서비스를 위한 정보를 저장 관리하는 데이터베이스를 보유한 고성능 서버, 이 둘을 연결하는 이더넷 기반의 LAN등의 기술로 인해 클라이언트/서버 환경이 출현하여 이미 업계의 표준으로 자리 잡았다.

기업들은 자신의 전산 환경의 기반 구조인 메인프레임 기반의 컴퓨팅 구조를 다운 사이징, 라이트 사이징을 위해 재구축 하였고, 개인용 컴퓨터가 소개된 지 15년이 지난 지금 클라이언트/서버는 주도적인 아키텍쳐가 되었으며 대부분의 현대 기업 시스템의 근간을 형성하고 있다. 물론 아직도 일부 레가시(legacy) 어플리케이션이 있다 할지라도 이들 레가시(legacy) 어플리케이션에 대한 데이터를 연동하는 역할까지도 클라이언트/서버 시스템이 중요한 역할을 수행하고 있다.

비록 개인용 컴퓨터가 호스트 중심의 케릭터 모드 터미널에 대한 대안이 되어 전산 환경 구축에 클라이언트/서버가 많은 역할을 하였지만 몇가지 과제를 안고 있다.

첫째, 클라이언트가 더 많은 하드웨어와 소프트웨어를 요구하면서 점점 거대해져가지만 실지로 활용도 측면은 사용자에 따라 천차만별이었다. 단순 사용자에게도 똑 같은 하드웨어 사양을 요구하게 됨으로써 결국 클라이언트 자원에 대한 낭비를 가져 왔다.

둘째, 수백 개의 클라이언트에 있는 여러 가지 소프트웨어 패키지(package)의 버전을 관리하기 위해서는 상당한 자원이 필요하게 되었고 이는 궁극적으로 클라이언트 관리가 점점 힘들어져 가고 있다.

반면 지난 수년간 웹을 기반으로 하는 인트라넷, 인터넷 기술은 비약적인 발전을 거듭하여 왔고, 이에 대한 구체적인 표현으로 네트워크 컴퓨팅이 급부상하고 있다. 앞으로 2-3년 내에 이와 같은 네트워크 컴퓨팅은 컴퓨터 기술 분야에 확고한 위치를 점하게 되고 이는 위의 클라이언트/서버에 대한 문제점을 해결할 수 있는 새로운 기술 분야가 되리라고 생각된다. 앞으로 네트워크 컴퓨팅에 대한 많은 연구가 필요하리라 생각되어진다.

홍 정 화

1990년 인하대학교 전산학과 (학사)
1992년 인하대학교 전산학과 (석사)
1992년 삼성종합기술원 연구원
1997년-현재 한국오라클(주) 기술본부 제조전문팀 차장

관심분야 : Data Warehousing, Application tuning