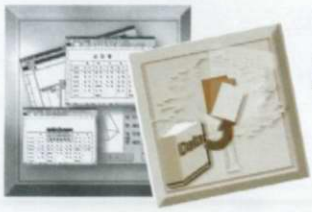


분산화와 클라이언트 서버

최용락
프로소프트웨어 대표

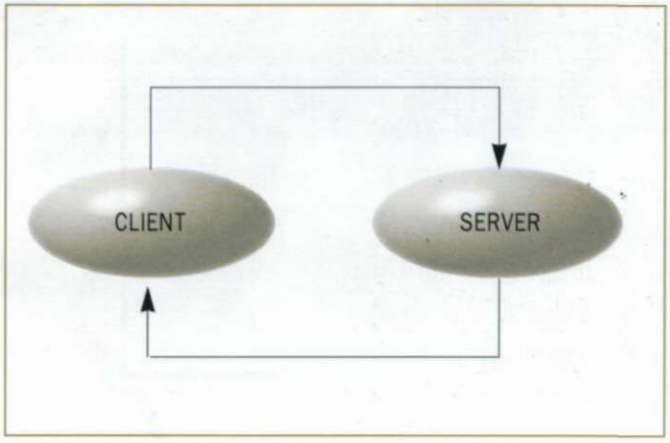
2. 클라이언트/서버

최근 수년간 정보산업 관련 업계 및 업체에서 가장 관심을 갖고 추구한 것이 바로 클라이언트/서버 형태의 정보 시스템 구축이다. 물론 이러한 방법의 구성은 하드웨어의 발전뿐 아니라 통신 즉 네트워크의 발전과 병행하여 생각해 볼 수 있다.

또한 각 어플리케이션을 구축하기 위한 제반 환경 및 도구(Tool)들의 분산환경 지원도 빼놓을 수 없을것이다. 그리고 하드웨어/오퍼레이팅 시스템/네트워크/데이터베이스/사용자 인터페이스 환경의 개방화(Open System) 등을 들 수 있다.

클라이언트/서버의 정의를 내려보면 서로 별개의 시스템으로써 단독으로 운영 가능하며 각각의 프로세서가 서로 다른 프로세서들에게 서비스를 요청하고 그 요청에 대해 응답(처리)을 행하여 상호 연결 운용을 하는것을 말한다. 간략하게 말하여 하나의 프로세서가 다른 프로세서들에게 서비스를 요구하고 이러한 서비스 요구에 대해 응답처리하는 시스템 구조를 말한다.

〈그림 1〉 Client/Server 란?



이러한 클라이언트/서버 구조에서는 정보의 요구자(클라이언트)와 공급자(서버)사이의 통신은 "명쾌한" 접근이 필수적이다.

약 2년전에 극장가에서 상영된 영화중에서 "의뢰인"이라는 영화가 상영되어 많은 관객을 동원했었다. 그 영화의 영어 원제는 "클라이언트"였다.

다시말해 클라이언트란 사건을 변호 의뢰하는 사람이라는 법정용어이다. 여기에 대해 변호사는 의뢰된 사건에 서비스를 제공하는 것이다. 이러한 관계가 클라이언트/서버의 관계인 것이다.

클라이언트/서버구조에서 생각해 보면 모든 프로세서로 부터 서로 다른 프로세서로 처리를 의뢰하여 처리가 되는 구조이므로 현재까지 우

데 이 터 베 이 스 리 포 트②

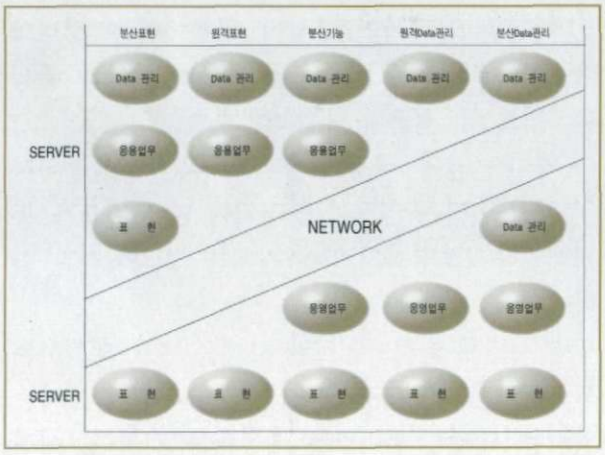
클라이언트 서버(Ⅰ)

리가 사용하던 정보시스템에서 생각하면 터미날 서버, 화일 서버, 데이터베이스서버, 디스크서버, 메일 서버, 통신 서버, 디스플레이 서버, X-윈도우서버 등 다양한 서버들을 생각해 볼 수 있다. 최근에는 프린터 서버, 팩시밀리 서버, 컴퓨테이셔널 서버, 등을 비롯하여 데이터베이스의 데이터와 표현(Presentation)사이의 연결을 통해 3티어(3-Tier)가 가능하게 하는 어플리케이션 서버의 발전까지에 이르게 되었다.

●클라이언트/서버 발전 5단계

클라이언트/서버의 발전 5단계를 살펴보면 〈그림 2〉의 단계로 발전 되어간다.

〈그림 2〉



물론 표현은 GUI환경을 포함한 End-User 컴퓨팅을 지향하며 사용자가 보다 편리하고 쉽게 요구하는 데이터에 접근이 가능하게 해야한다.

또한 응용업무(Application)는 최근의 3티어(3-Tier)방법을 이용한 보다 신속한 데이터와 사용자의 요구와의 연결을 책임진다. 그리고 궁극적인 클라이언트/서버의 목표는 공용 데이터(Common Data)의 일부를 제외하고 모든 데이터까지를 클라이언트에 상주시키는 것이다.

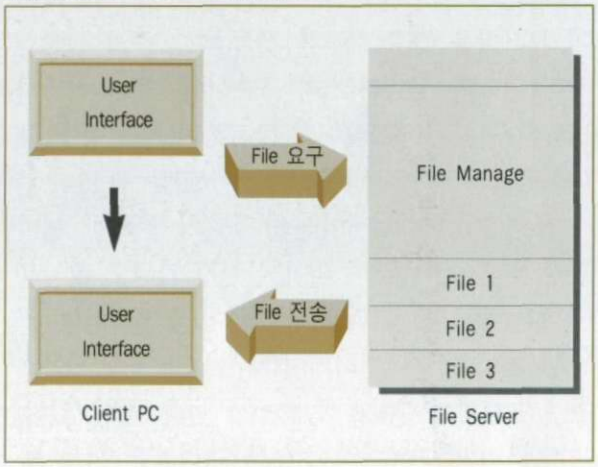
이는 물론 통신(네트워크)의 발달 및 초고속화가 가능해져서 통신에 대한 과부하(Over-Load) 또는 병목현상(Bottle Neck)이 발생되지 말아야 하고 워크스테이션(Work Station)급 하드웨어의 기능향상 및 가격하락 등이 지속적으로 이루어져야 한다.

또한, 개발도구(Development Tool)의 기능 및 성능 향상 그리고 케이스툴의 발전 등을 생각해야 한다.

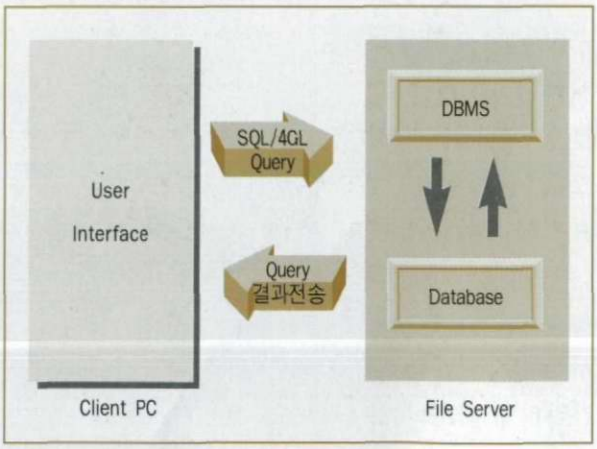
●화일 서버 모델 과 클라이언트 서버 모델

다음의 〈그림 3〉과 〈그림 4〉는 화일 서버 모델과 크라이언트/서버 모델의 구조적인 비교이다. 이 그림들을 〈그림 2〉와 연결하여 생각해 보면

〈그림 3〉 File Server Model



〈그림 4〉 Client/Server Model



기술의 발전 방향과 각각의 단위들이 분산 데이터 관리에서는 클라이언트 PC로 이양되어 갈 것이라는 것을 생각해 볼 수 있다.

●클라이언트 /서버 요구사항

클라이언트/서버 구조에서의 요구사항을 생각해 보면

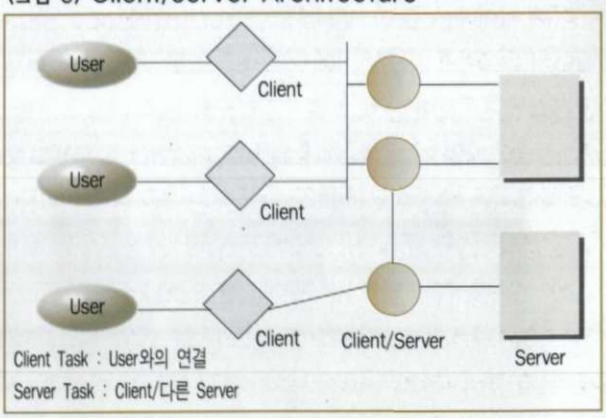
① 클라이언트 최종 사용자에게 친숙한 GUI(Graphic User Interface)환경
② 클라이언트 최종 사용자에게 중요한 어플리케이션 로직(Application Logic) 일부분 또는 전체가 존재
③ 네트워킹 능력을 갖춘 클라이언트/서버 시스템
④ 클라이언트 앤드(End)와 서버 앤드는 구분되어야 하며 상호 연결 운용가능
⑤ 분리된 플랫폼(Platform)에서 운용되며 하나의 플랫폼에서 작업 가능하도록 투명성과 독립성 보장
⑥ 서버는 동시에 다중 클라이언트 운용가능
⑦ 작동은 항상 클라이언트 앤드에서 시작하며 서버쪽에서는 콘트롤(Control) 가능
⑧ 서버는 데이터 보호, 보안, 백업, 복구, SQL 기능 제공
⑨ 하드웨어 플랫폼 업그레이드(Upgrade)시 양쪽 (클라이언트/서버) 모두 업그레이드 불필요 등이다.

●클리이언트 / 서버의 구조

클라이언트/서버의 구조는 〈그림 5〉와 같이 구성된다. 중간의 클라이언트/서버를 미들 서버(Middle Server)라고 한다.

여기에서 클라이언트가 해야 할 일은 사용자와의 연결이며 서버가 해야 할 일은 클라이언트와 다른 서버와의 연결이다. 이를 다시 작업 영역으로 생각해 보면 클라이언트 어플리케이션은 사

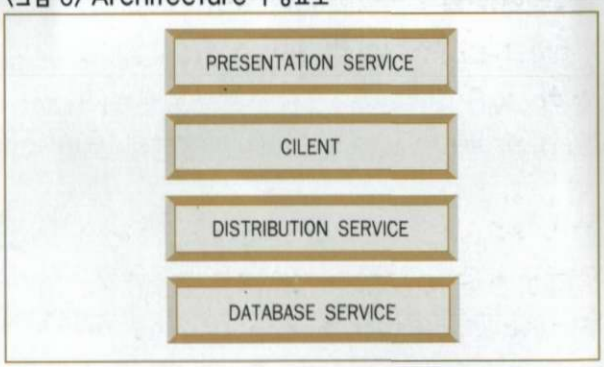
〈그림 5〉 Client/Server Architecture

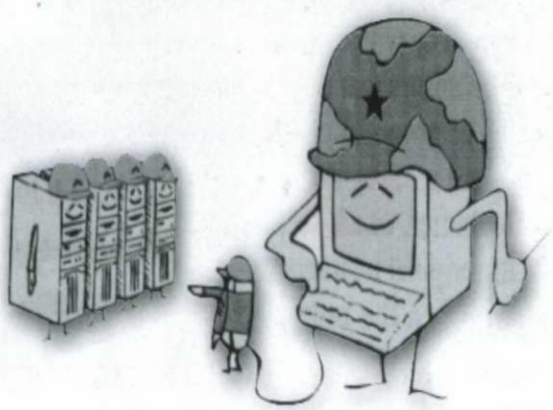


용자와의 연결 및 서버 어플리케이션으로의 전송, 파견이며 서버 어플리케이션은 업무의 로직을 관리하며 데이터베이스 접근 및 처리를 관리한다.

클라이언트/서버 구조의 구성요소를 생각해 보면 〈그림 6〉과 같다.

〈그림 6〉 Architecture 구성요소





이러한 구조가 3-Tier의 대표적 구조이다.

여기에서 표현 서비스(Presentation Service)는 사용자에게 어떤 방법으로 잘 전달하는가를 하는 것이다.

또한 사용자에게 확장된 기능 및 연결, 대화형 통제를 들 수 있고, 클라이언트는 데이터를 받는 일, 받아들인 데이터가 올바른 것인가를 검증하며 보안 등을 담당한다. 분배 서비스(Distribution Service)는 프로그램과 프로그램의 연결 및 운영, 프로토콜의 관리, 메세지의 전송 등을 담당한다. 서버는 코드 및 테이블의 접근, 업무 처리 기능 등을 담당한다.

그리고, 데이터베이스 서비스(Database Service)는 데이터의 불일치 제거 및 무결성 유지, 데이터 Locking의 관리, 효율성 극대화 등을 담당한다.

●클라이언트 / 서버의 작업 분할

클라이언트/서버의 작업 분할을 고려해 보면 흔히 요즘 자주 사용되는 프런트앤드(Front-End)와 백앤드(Back-End)로 설명해 보면 〈그림 7〉과 같다.

〈그림 7〉 Client/Server 작업 분할

Application Program (Front-End)	Database Server (Back-End)
• Forms Design	• Stotage
• Presentation	• Security
• Application Logic	• Data Administration
• Data Manipulation	• Selections/Aggregation
• Query Tools	• Indexing/Sorting
• Menu/Utilities	• Batch Updates

●클라이언트 / 서버의 필요성

클라이언트/서버 모델이 필요한 경우는 먼저 성능(Performance)의 향상을 기대할 때이다.

①CPU(Central Processing Unit)의 성능 향상 즉 보다 강력한 CPU가 필요하거나 다중 CPU가 필요한 경우,

②메모리(Memory) 증설이 필요한 경우 또는 어플리케이션 프로그램이나 처리(Processing)의 분산이 필요한 경우

③디스크 I/O가 많이 발생하여 다중 디스크를 사용해야 하거나 서버에서 과다한 디스크 I/O가 발생하여 클라이언트에서 어플리케이션을 수행해야 하는 경우

④락킹(Locking)처리의 어려움이 있을때 데이터별, 어플리케이션별, 사용자별 락킹을 고려할때 그리고, 처리비용의 절감을 위하여 고려할 수 있다.

⑤적절한 지역(장소)에 적당한 성능의 하드웨어를 두어 중앙의 하드웨어의 과부하를 줄이며 단위당 증설 비용을 절감하고자 할때이다.

⑥클라이언트/서버 구조에서 어드레스(Address)처리를 물리적으로 다른 지역(Site)에서 처리할 수 있도록 할 때이다.

여기에서 클라이언트/서버 구축의 문제점을 간략히 생각해 보자.

이제 클라이언트/서버 구축을 할 수 있는가라는 문제보다는 어떻게 구축하는가가 문제이다.

구축시 문제점의 가장 대표적인 것은

①클라이언트/서버에 대한 막연한 기대이다. 다시 말해 자신의 정보 시스템 상황이나 필요한 정보 구조 등을 고려하지 않은 접근이다.

②클라이언트/서버를 구축하는 동안에도 기존 어플리케이션의 지원이다. 다시 말해 클라이언트/서버를 구축하는 동안에도 기업의 정보시스템은 지속적으로 운영되고 있다는 것이다. 여기에서 여러분은 구축 기간동안 업무에 과부하가 부여된다는 점을 유념해야 한다.

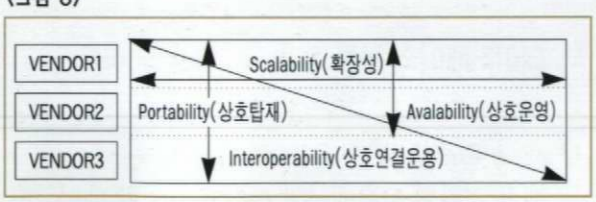
③기존의 정보시스템 즉 하드웨어, 어플리케이션, 오퍼레이팅 시스템과의 공존하는 시스템 구축을 할 수 있는가 이다. 다시 말해 기존 투자된 정보시스템을 그대로 이용 가능한가 이다.

④기존 어플리케이션과 향후 구축될 새로운 환경과 통합 운영될 수 있는가 하는 문제이다.

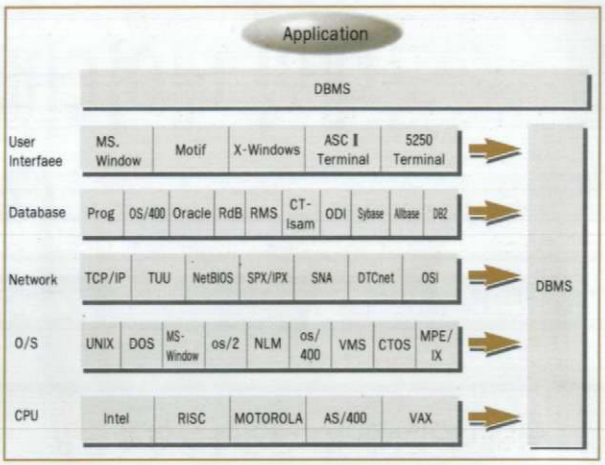
여기에서 우리는 개방 시스템(Open System)을 이해해야 할 것이다. 1980년 초반 이후 지속적으로 발전해온 개방 시스템은 사용자가 시스템 구축시 하드웨어, 오퍼레이팅 시스템, 네트워크, 데이터베이스, 소프트웨어의 선택을 업체 및 환경에 구애 받음 없이 자유롭게 결정할 수 있다는 개념이다.

이는 또한 최소의 비용으로 최고 효율의 시스템 구축 실현을 목표로 한다.

〈그림 8〉



그리고, 개방시스템을 통한 고려사항을 생각해 보면 〈그림 9〉와 같을 것이다.



●클라이언트 / 서버 처리의 정의

끝으로 클라이언트/서버 처리의 정의를 생각해 보자

여기에서 우리는 응용프로그램의 3가지 업무를 고려해 보면 어플리케이션 작업의 3단계를 이해하면 될 것이다.

①사용자 연결 업무(User Interface Processing)

②응용 프로그램 로직(Application Program Logic)

③데이터 베이스 업무(Database Processing)

User Interface Processing	Application Logic		DataBase Work	
	Main Logic	Stored Procedures	Join & Query Optimization	DataBase Accesses

위의 그림을 이해하고 자신의 어플리케이션, 데이터를 어떻게 클라이언트/서버화 할것인가를 고려하고 특히 자신의 데이터베이스가 어떠한 구조에서 가장 좋은 클라이언트/서버 구조인가를 이해해야 할 것이다. DpC