클라이언트/서버 환경에서 분산 어플리케이션의 기본유형과 적용기준*

김갑중
인하대학교 경영학부 부교수
(garpkim@dragon.inha.ac.kr)
신재훈
삼성SDS 책임
(seattle@samsung.co.kr)

오늘날 클라이언트/서버 기술은 과거의 정보시스템이 안고 있었던 여러 가지 문제들을 해결해 줄 수 있는 일반적인 해결책으로 인식되고 있다. 따라서 클라이언트/서버 환경으로 전환하였거나 계획 중에 있는 기업의 수가 빠른 속도로 늘어가고 있다. 그러나 클라이언트/서버를 둘러싼 다양한 기술적 발전에도 불구하고 다음과 같은 문제들에 대하여는 별다른 관심이 주어지지 않고 있다. 즉, 클라이언트/서버 시스템에는 어떠한 유형들이 가능한가? 어떠한 업무가 클라이언트/서버 환경에 가장 적합한 것인가? 클라이언트/서버 환경의 유형과 적용업무의 특성간에는 어떤 관계가 있는가? 클라이언트/서버 환경으로 시스템을 구축하고자 할 때 가장 먼저 직면하게 되는 이와 같은 질문에 대하여 아이러니칼하게도 이렇다 할 연구와 대안의 제시가 없었다. 본 논문의 목적은 기업의 정보시스템 관리자 또는 시스템 개발자로 하여금 적용하고자 하는 업무의 특성에 맞는 최적의 클라이언트/서버 유형으로 구축할 수 있도록 지침을 제공하는 것이다. 이를 위해 본 논문에서는 분산 어플리케이션의 가능한 기본유형들을 체계적으로 도출하였고 각 업무의 특성을 61개의 항목으로 분류하여 기본유형과 각 업무특성 간의 적합도를 전문가들의 의견을 종합하여 계량적으로 산출하였다. 본 연구의 결과에 의하면 클라이언트/서버 환경 하에서 분산 어플리케이션의 기본유형은 총 14개의 종류가 있으며, 적용기준을 위해서는 4가지의 측면을 고려하여야 한다. 클라이언트/서버 시스템 구축시, 업무의 특성을 상세하게 분석하고 여기로부터 도출되는 요구사항을 적용기준에 있는 각 항목과 비교 검토함으로써 해당 업무의 특성을 가장 잘 반영하는 분산 어플리케이션의 유형이 무엇인지를 찾아낼 수 있을 것이며 이를 통해 최적의 클라이언트/서버 시스템을 구축할 수 있을 것이다.

I. 서 론

오늘날 클라이언트/서버(Client/Server) 기술은 과거 메인프레임(Mainframe)을 중심으로 한 Legacy System의 고질적 문제점인 개발기간의 장기화, 생산성의 저하, 유지보수의 어려움, 비용의 지속적 증가, 이기종 환경통합의 어려움 등을 해결해 줄 수 있는 새로운 기업 정보시스템으로 널리 확산되고 있다. 1980년대 말, 처음 등장하였을 때만 하더라도 일시적 유행으로 여겨지던 클라이언트/서버 환경은, 네트워크 기술, TCP/IP 프로토콜, 최종사용자컴퓨팅(EUC)을 위한 GUI 기술, 객체지향기술, 미들웨어 및 각종 개발도구의 발달에 힘입어 이제는 기업의 새로운 컴퓨팅 환경으로 자리매김을 하였다.

논리적 모형으로서의 클라이언트/서버 기술은 클라이언트와 서버 프로세서 사이에 응용처리를 적절히 나누어 비즈니스 요구를 만족시키는 분산컴퓨팅이라고 간단히 정의할 수 있다. 하지만 이를 실제로 구현하는 방법은 사용하는 요소기술에 따라 다양할 수 있다. 따라서 어떤 업무를 클라이언트/서

논문 접수일 : 98. 1 게재확정일 : 98. 5
* 이 논문은 1997년 한국학술진흥재단의 공모과제 연구비와 1997년 인하대학교 교내연구비의 공동 지원에 의하여 연구되었음.

버로 구현하려면, 그와 관련된 업무환경과 정보처리특성 등을 면밀히 검토한 후 이를 다양한 클라이언트/서버 구현방식과 비교분석해야만 그 업무의 특성에 가장 적합한 시스템을 구축할 수 있을 것이다 (김영걸, 박영민, 1998).

그러나 클라이언트/서버를 둘러싼 다양한 기술적 발전에도 불구하고 아이러니컬 하게도 클라이언트/서버 환경으로 시스템을 구축하고자 할 때 가장 먼저 직면하게 되는 클라이언트/서버 환경과 업무 특성과의 관계에 대하여는 이렇다 할 연구가 진행되어 오지 못하였다. 즉, 클라이언트/서버를 구축하기 위한 기술적인 방법은 다각도로 연구되어 왔으나, 각각의 구현 방법에 가장 적합한 업무의 특성이 무엇인가에 대한 연구는 상대적으로 취약했었다. 이러한 인식을 바탕으로 본 논문에서는 업무특성별 최적의 클라이언트/서버 시스템을 구축하기 위한 방법론을 제시하고자 한다. 이를 위해 먼저 클라이언트/서버 환경에서 구현이 가능한 분산 어플리케이션의 기본유형을 분류·정립하고 도출된 기본유형과 업무특성과의 관계를 분석할 수 있는 적용기준을 제시할 것이다.

II. 분산 어플리케이션에 대한 기존의 논의

분산 어플리케이션이란 어플리케이션의 구성요소가 둘 이상의 물리적 단위로 나뉘어진 것을 의미한다. 전통적인 비분산컴퓨팅 환경에서는 어플리케이션의 구성요소에 대하여 크게 관심을 두지 않아도 문제가 없었다. 어플리케이션의 운영을 위해 필요한 모든 요소가 하나의 시스템 안에 존재하므로 각 구성요소의 분할이란 큰 의미를 가질 수 없었기 때문이다. 그러나 클라이언트/서버와 같은 분산컴퓨팅 환경에서는 흩어져 있는 컴퓨팅 자원에 어플리케이션을 어떻게 나누어 배치할 것인가라는 문제가 중요한 이슈로 등장하고 이를 해결하기 위하여는 어플리케이션이 어떠한 요소로 구성되어 있는가를 이해하는 것이 필수적이다. 우리가 일반적으로 어플리케이션이라고 부르는 것은 표현(Presentation), 기능(Function), 데이타 관리(Data Management)의 세 가지 구성요소를 합친 것으로 기업 및 조직의 특정 정보요구사항을 만족시키기 위해 개발·운영하는 프로그램을 말한다.

(1) 표현(Presentation): 사용자 인터페이스(User Interface)라고도 불리는 표현층(Presentation Layer)은 최종사용자와 직접적으로 상호작용을 하는 부분이다. 사용자는 표현층을 통하여 자신이 원하는 정보를 입력하거나 처리결과를 조회해 볼 수 있다.

(2) 기능(Function): 업무논리(Business Logic)로도 불리는 기능층(Function Layer)은 어플리케이션의 핵심에 해당하는 부분으로서 표현층을 통해서 입력된 사용자 요구사항을 미리 정해진 업무규칙(Business Rule)에 따라 처리하는 요소이다.

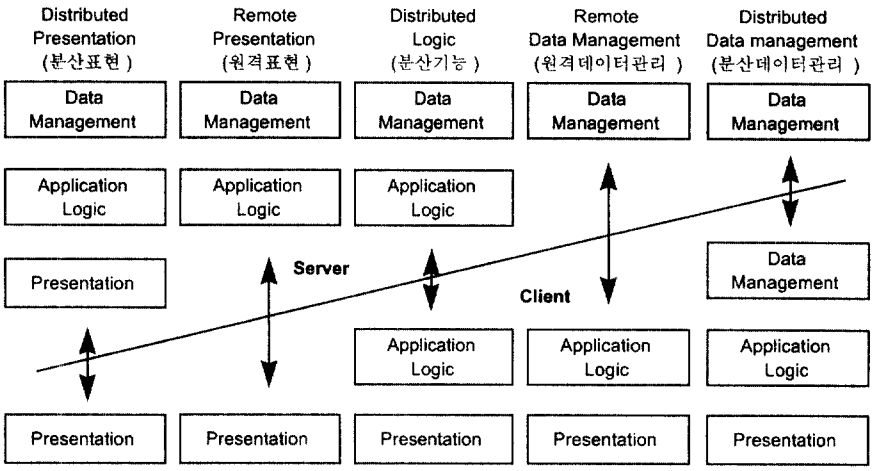
(3) 데이타관리(Data Management): DB 자원(Database Resources)으로도 불리는 데이타관리층(Data Management Layer)은 기능층의 처리대상인 데이타를 관리하는 부분이다. 이때, 데이타관리에 필요한 로직을 프로그램 내에 포함시킬 수도 있지만 요즈음에는 별도의 DB 관리시스템을 통하여 데이타를 관리하고 프로그램에는 이것과의 연결로직(Link Logic) 만을 남기는 것이 보다 일반적이다.

분산 어플리케이션을 구현하기 위하여 클라이언트와 서버 사이에 구성요소를 분할하는 가장 단순

한 방식은 해당 어플리케이션을 각 구성요소별로 나누는 것이다. 즉, 표현층을 클라이언트 쪽에 두고 기능층과 데이타관리층을 서버에 두거나, 표현층과 기능층을 클라이언트 쪽에 두고 데이타관리층을 서버에 두는 것이다. 이러한 형태는 어플리케이션을 원래의 구성요소에 따라 분할하는 것으로 이해가 쉽고 단순하다는 장점이 있는 반면, 기능층이 클라이언트나 서버의 어느 한 편에 몰려 있기 때문에 클라이언트와 서버간의 협동적 처리(Collaborative Processing)라는 개념을 충분히 구현하기에는 미흡하다는 단점이 있다. 다음으로 생각할 수 있는 분할형태로는 기능 자체를 나누어 일부는 클라이언트에, 또 다른 일부는 서버에 두고 이들 사이의 협력을 통하여 정보를 처리하는 보다 복잡한 것이 있다. 이 방식은 분산컴퓨팅의 원래 아이디어에 충실하다는 장점이 있지만 하나로 수행하던 기능을 클라이언트와 서버로 나누어 처리함으로써 추가적 인터페이스 로직(Interface Logic)을 개발하여야 하고 네트워크에 부하를 많이 주어 실제 운영시 성능이 저하될 수 있다는 단점이 있다.

분산 어플리케이션의 유형과 관련하여 가장 널리 알려진 분할모델은 가트너그룹(Gartner Group)이 발표한 모델이다. 가트너그룹은 앞에서 살펴 본 어플리케이션의 기본적인 분할 유형에다 표현층과 데이타관리층을 분할하는 모델을 추가함으로써 [그림 1]과 같은 5가지의 모델을 제시하였다. 이 중에서 원격표현(Remote Presentation), 원격데이타관리(Remote Data Management), 분산기능(Distributed Logic)은 산업계와 학계에서 클라이언트/서버 환경의 기본모델로 널리 받아 들여져 현재 분산 어플리케이션 구현의 규범적인 형태로 인식되고 있다.

그러나, 가트너모델로 대표되는 분산 어플리케이션에 대한 기존의 논의는 다음과 같이 실질적인 적용 및 구현 측면에서 한계점을 드러내고 있다. 첫째, 가트너모델은 어플리케이션의 구성요소의 물리적 배치만을 고려하였을 뿐, 각 구성요소의 논리적 관계에 대하여는 언급이 없다. 둘째, 가트너모델은



〈그림 1〉 분산 어플리케이션의 가트너 모델

2층구조(Two-Tier Architecture) 만을 염두에 두었을 뿐, 전통적인 1층구조(One-Tier Architecture)나 최근에 활용이 늘고 있는 3층구조(Three-Tier Architecture)에 대하여는 간과하고 있다. 셋째, 가트너모델은 어플리케이션 구성요소를 나눈 결과만을 보여줄 뿐, 왜 그렇게 나누어야 하는지에 대한 기준 제시가 불분명하다. 넷째, 가트너모델은 각 유형이 어떤 업무에 적합한가에 대한 논리적 설명과 기준이 없다. 이와 같이 클라이언트/서버 환경에 기반한 분산 어플리케이션에 대한 기존 논의는 단순한 유형의 나열일 뿐, 왜 어떤 업무를 특정 유형으로 구현하여야 하는가를 명확하게 설명하고 있지 않기 때문에 업무분석을 바탕으로 클라이언트/서버 환경을 구축하려는 개발자에게 실질적 도움을 주는데 한계가 있다.

III. 분산 어플리케이션의 기본유형

1. 기본유형의 분류 기준

분산 어플리케이션의 기본유형을 도출하기 위해서는 일정한 기준이 필요한 바, 본 연구에서는 다음 3가지 기준을 적용하고자 한다 (Gartner Group, 1994).

① 하드웨어 층(Hardware Layer: HL): 하드웨어 층이란 메인프레임, 중형서버, PC 등의 물리적 하드웨어를 의미한다.

② 물리적 소프트웨어 층(Physical Software Layer: PSL): 물리적 소프트웨어 층은 어플리케이션의 3대 요소를 물리적으로 몇 군데로 나눌 것인지와 관련된 기준으로서 HL과 내용 면에서 동일한 구조를 가지고 있다. 즉, 세 가지 구성요소가 모두 하나의 메인프레임에 있으면 1층구조이고, PC와 중형 혹은 대형서버에 나뉘어져 있으면 2층구조이며, 엔터프라이즈 서버와 중형서버 그리고 PC에 나뉘어져 있으면 3층구조를 이룬다. 그러므로 본 연구에서는 HL을 PSL과 동일한 것으로 취급한다. 이러한 관점에서 볼 때 앞에서 언급한 가트너모델은 PSL을 기준으로 분류한 것이며 그 중에서도 2층구조만을 다룬 것이라고 할 수 있다.

③ 논리적 소프트웨어 층(Logical Software Layer: LSL): 논리적 소프트웨어 층의 구분은 어플리케이션 구성요소 간의 연결부분(Interface)을 명확하게 정의함으로써 가능하다. 어플리케이션의 각 구성요소를 모듈화하여 연결부분을 명확히 함에 따라, 설계자는 어플리케이션의 각 요소를 상호독립적으로 설계할 수 있으며 융통성, 확장성, 유지보수의 용이성 등의 측면에서 이점을 얻을 수 있다. LSL의 관점에서 볼 때 전통적 파일시스템처럼 표현, 기능, 데이타관리를 논리적으로 명확히 구분하지 않고 혼용하여 구현하는 것은 1층구조이고, 표현과 기능을 하나의 층으로 하고 데이타관리를 또 다른 층으로 분리하거나, 표현을 하나의 층으로 하고 기능과 데이타관리를 다른 층으로 분리하는 것이 2층구조이며, 표현과 기능, 데이타관리를 모두 분리하여 각 요소별 논리적 독립성을 확보하는 것은 3층구조라 할 수 있다.

본 논문에서 적용할 기준은 PSL과 LSL이다. PSL은 어플리케이션 구성요소의 실질적인 위치와 관련된 것으로 표현과 기능, 데이타관리를 몇 군데로 나눌 것인가에 따라 1층, 2층, 3층으로 구분된다. 한편, LSL은 표현과 기능, 데이타관리를 각기 독립적으로 설계할 수 있는가와 관련된 것으로 이 역시 1층, 2층, 3층으로 구분될 수 있음은 이미

살펴본 바와 같다.

2. 기본유형의 분류

PSL과 LSL을 기준으로 적용할 때 고려해야 할 4 가지 전제조건은 다음과 같다.

어플리케이션 구성요소간의 인터페이스: 세 가지 구성요소는 각기 독립적으로 설계가 가능하다. 그러나 실제 구현에 있어서 표현과 데이타 관리는 기능을 통해서만 연결이 되며 인터페이스가 가능하다.

어플리케이션 별 분할: 기본유형의 설정 시, 분할은 단위 어플리케이션을 기준으로 한다. 즉 어플리케이션의 집합으로 이루어진 시스템은 기본유형 설정의 고려대상에서 배제한다.

수평분할: 분할은 수평으로만 이루어진다. 이는 "어플리케이션 별 분할"이라는 전제조건에서 자연스럽게 도출되는 것으로서 수직분할은 단위 어플리케이션 선정에서 이미 이루어진 것으로 간주하고 기본유형 선정 시에는 수평으로 분할하는 경우만을 고려한다.

순서불변: 분할 방법에 관계없이 어플리케이션 구성요소의 인터페이스 순서는 변하지 않는다. 표현, 기능, 데이타 관리를 나누는 방법에 관계없이 어플리케이션 구성요소 상호간의 인터페이스 규칙은 반드시 지켜져야 한다. 즉, 표현 다음에는 반드시 기능이 와야 하고 기능 다음에는 반드시 데이타 관리가 와야 한다. 이 때 후위 구성요소(예: 기능)가 오기 전에 전위 구성요소(예: 표현)가 몇 개까지 올 수 있는지는 고려하지 않는다. 따라서 "표현-표현-기능-데이타 관리"나 "표현-기능-기능-기능-데이타 관리"는 가능한 분할의 예이지만 "표현-기능-표현-데이타 관리"나 "표현-기능-데이타 관리-기능-데이타 관리"는 기본유형으로 가능한 분할이 아니다.

이러한 전제조건 하에서 PSL을 주 결정자(Main Determinant)로 하여 기본유형의 전체집합을 구하면 다음과 같다.

(1) 1층 구조(1-Tier Architecture)

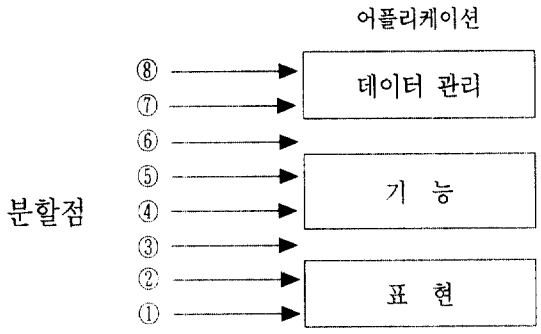
물리적으로 1층구조를 갖는 경우, LSL의 구분은 무의미해진다. 논리적으로 어떻게 나누는가와 상관없이 표현, 기능, 데이타 관리가 결과적으로 한 장소에 위치하기 때문이다. 또한 어플리케이션이 PC와 그룹서버, 엔터프라이즈 서버 중 중 어디에 있는가에 따른 기능적, 논리적 차이를 찾을 수 없으므로, 이 경우 가능한 기본유형은 단 한가지이다.

(2) 2층 구조(2-Tier Architecture)

분산 어플리케이션이 물리적으로 2층 구조를 갖는다는 말은 표현, 기능, 데이타 관리로 구성된 어플리케이션을 한 번만 분할한다는 말과 같은 의미를 같는다. 이 경우 LSL의 분할방법은 가트너모델과 같다. 즉, 구성요소의 인터페이스 부분을 분할점으로 하는 방법 두 가지와, 각 구성요소 자체를 분할점으로 하는 방법 세 가지를 합쳐 다섯 가지의 기본유형이 있다.

(3) 3층 구조(3-Tier Architecture)

분산 어플리케이션이 PSL관점에서 3층 구조를 갖기 위해서는 어플리케이션을 나누는 분할점이 2개 있어야 한다. 이 때 구성요소를 연결하는 인터페이스 부분을 두 번 나눈다는 것은 의미가 없으므로 논리적으로 가능한 분할점은 구성요소의 인터페이스 부분이 2개, 표현, 기능, 데이타 관리 각각을 나누는 분할점이 6개 (3x2=6)로 합계 8개가 된다.(〈그림 2〉 참조) 이 8개의 분할점을 기준으로 LSL을 3개로 나누는 방법은 〈표1〉과 같이 총 28



〈그림 2〉 3층 구조에서의 분할점

개(8C2=28)의 경우의 수가 성립되는데, 여기서 의미상 동일한 유형(표1의 비고란 참조)을 제외시키면 〈표 2〉에서 보는 바와 같이 총 13가지의 기본유형을 얻을 수 있다.

〈표 2〉에 보인 기본유형 가운데 표현층이 클라이언트와 그룹서버로 나누어진 형태를 취하고 있는 5개의 유형(가, 나, 다, 라, 마)은 이론상으로는 가능하지만, 현실에서는 실례를 찾기 어려운 것이다. 왜냐하면 3층구조로 분산 어플리케이션을 만들면

〈표 1〉 3층구조의 기본유형

순번	1	2	3	4	5	6	7	8	9	10	11	12	13	14
제1분할점	①	①	①	①	①	①	①	②	②	②	②	②	②	③
제2분할점	②	③	④	⑤	⑥	⑦	⑧	③	④	⑤	⑥	⑦	⑧	④
ES	D F P3	D F	D F3 F2	D F3	D	D3 D2	D3	D F	D F3 F2	D F3	D	D3 D2	D3	D F3 F2
GS	P2	P3 P2	F1 P3 P2	F2 F1 P3 P2	F P3 P2	D1 F P3 P2	D2 D1 F P3 P2	P3	F1 P3	F2 F1 P3	F P3	D1 F P3	D2 D1 F P3	F1
C	P1	P1	P1	P1	P1	P1	P1	P2 P1	P2 P1	P2 P1	P2 P1	P2 P1	P2 P1	P
비고				3과 동일			6과 동일	2와 동일	3과 동일	3과 동일	5와 동일	6과 동일	6과 동일	

순번	15	16	17	18	19	20	21	22	23	24	25	26	27	28
제1분할점	③	③	③	③	④	④	④	④	⑤	⑤	⑤	⑥	⑥	⑦
제2분할점	⑤	⑥	⑦	⑧	⑤	⑥	⑦	⑧	⑥	⑦	⑧	⑦	⑧	⑧
ES	D F3	D	D3 D2	D3	D F3	D	D3 D2	D3	D	D3 D2	D3	D3 D2	D3	D3
GS	F2 F1	F	D1 F	D2 D1 F	F2	F3 F2	D1 F3 F2	D2 D1 F3 F2	F3	D1 F3	D2 D1 F3	D1	D2 D1	D2
C	P	P	P	P	F1 P	F1 P	F1 P	F1 P	F2 F1 P	F2 F1 P	F2 F1 P	F P	F P	D1 F P
비고	14와 동일			17과 동일				21과 동일	20과 동일	21과 동일	21과 동일		26과 동일	

〔범례〕 D: 데이타 관리(Data Management), F: 기능(Function), P: 표현(Presentation)
ES: Enterprise Server, GS: Group Server, C: Client

〈표 2〉 중복을 제외한 3층구조의 기본유형

구분	가	나	다	라	마	①	②	③	④	⑤	⑥	⑦	⑧
ES	D F P	D F	D F	D	D	D F	D	D	D F	D	D	D	D
GS	P	P	F P	F P	D F P	F	F	D F	F	F	F	D	D
C	P	P	P	P	P	P	P	P	F P	F P	F P	F P	D F P

서 서버에 별도의 표현층을 두면 효용은 적으면서 구현의 복잡성만 늘어나기 때문이다. 따라서 이들을 제외시키면 3층 구조에서 고려해야 할 분산 어플리케이션의 실질적 유형은 〈표2〉 1-8번에 보인 것과 같이 8개가 된다. 이상의 논의를 종합하면 클라이언트/서버 환경에서 개념적, 논리적으로 구분가능한 분산 어플리케이션의 기본유형은 총 14개(1층 구조 1개, 2층 구조 5개, 3층 구조 8개)가 된다.

기본유형에 대한 논의를 마치기에 앞서 적용기준 서술을 위한 예비작업으로 3층구조에서 가능한 8가지 기본유형을 데이타의 집중/분산 여부와 클라이언트에 기능이 존재하는지의 여부를 기준으로 재분류하면 〈표3〉과 같다.

위 표에서 P-F-*D는 클라이언트에 기능이 없으면서 데이타가 집중되어 있는 P-F-D와 P-F-FD를 단일그룹으로 표시한 것이며 PF-F-*D는 클라이언트에 기능이 있으면서 동시에 데이타가 집중되어 있는 PF-F-D와 PF-F-FD를 묶어서 표시한 것이다. 한편 PF*-D-D는 클라이언트에 기능이 있으면서 데이타가 분산되어 있는 PF-D-D와 PFD-D-D를 함께 표시한 대표유형이다.

〈표 3〉 3층 구조 기본유형의 재분류

	데이터의 집중	데이터의 분산	
		지역서버를 공용서버로 사용	지역서버를 전용DB서버로 사용
Client에 기능이 없는 경우	P-F-*D	P-FD-D	
Client에 기능이 있는 경우	PF-F-*D	PF-FD-D	PF*-D-D

Ⅳ. 기본유형의 적용기준

지금까지 클라이언트/서버 환경에서 가능한 분산 어플리케이션의 기본유형에는 어떠한 것들이 있는지 살펴보았고 이제는 특정 업무에 적합한 기본유형이 무엇인가를 결정하는 기준에 대해 알아보고자 한다. 이를 위해 본 논문에서는 3-5년간 클라이언트/서버 환경에서 시스템을 설계, 구축해 본 경험이 있는 7-10명의 실무전문가들의[1] 의견을 수렴하여 다음과 같이 문제에 접근하였다.

1) 참여자들의 구성을 살펴보면 개발자 3명, 분석/설계자 3명, 전반관리자 1명이 모든 토의과정에 참여하였으며 데이타베이스, 네트웍/통신, 소프트웨어 공학분야의 전문가들이 부분적으로 참여하였다. 이들은 각각 자신의 경험과 지식을 바탕으로 의견을 개진하였으며 항목의 선정과 기술방법에 있어서는 모든 참여자의 동의를 기본전제로 하였다.

① 우선, 어플리케이션이 사용 가능하기 위해 갖춰야 할 특성을 세가지 구성요소인 표현층, 기능층, 데이타 관리층 외에 운영관리(보안관리, 장애대책, 유지보수 등) 측면을 더하여 4개의 대항목으로 구분하고, ② 이를 다시 28개의 소항목과 61개의 세부항목으로 나누어 정리하였다. ③ 그런 다음 각각의 기본유형들이 세부항목의 기술적 요구사항을 얼마만큼 잘 만족시켜 줄 수 있는가를 5등급으로 계량화시켰다.

이러한 방법으로 정립된 적용기준은 어플리케이션이 갖춰야 할 특성별로 가장 적합한 클라이언트/서버의 기본유형이 무엇인지를 보여준다. 따라서 실제 적용에 있어서는 클라이언트/서버로 구축하려는 업무가 적용기준에서 밝힌 어플리케이션의 특성 중에서 어떠한 세부항목들을 필요로 하는지를 검토한 후, 각 세부항목에 따른 기본유형별 배점을 합산하여 합계치가 가장 큰, 또는 선택한 세부항목을 가장 많이 만족시키는 기본유형을 선택하면 될 것이다. 이하에서는 이를 보다 상세히 기술하고자 한다.

1. 어플리케이션의 특성분류

(1) 표현층(Presentation)측면

윈도우즈의 사용이 보편화 되고 그래픽이나 이미지 처리가 강력한 PC의 보급에 따라 사용자 인터페이스가 더욱 중요한 의미를 갖게 되었다. 기존의 텍스트 중심의 프로그램이 GUI 위주로 이전되고 있으며 또한 사용자의 다양한 정보욕구를 충족시키기 위한 멀티미디어의 지원처리가 보편화되고 있다. 본 논문에서는 표현층의 처리특성을 다음과 같이 4가지 항목으로 나누었으며 각 항목에 대한 기본유형의 지원정도를 파악하면 다음과 같다.

1) 윈도우즈(Windows)환경을 통한 다중창, 다중작업처리의 지원 정도: 그래픽 중심의 사용자 인터페이스와 멀티태스킹(Multi-tasking)이 가능하여야 하며, 이를 위하여는 사용자 측에서 GUI기능과 멀티태스킹을 지원할 수 있어야 한다.

2) 음성처리지원: 음성 데이타 처리를 위하여는 클라이언트 또는 서버 중 어느 하나가 단독으로 처리하는 것보다 양자간의 협동작업으로 처리 운용을 하는 경우가 효과적이다.

3) 화상처리지원: 정지화상 또는 동화상의 처리는 클라이언트 쪽보다는 성능이 우수한 서버 쪽에서 관리하여 클라이언트를 화상 데이타 처리에서 분리시키는 것이 보다 효과적이다.

4) 화상회의지원: 화상회의는 화상처리와 음성처리에 기초하며 다자간 통신이 가능해야 한다. 이에 가장 적합한 것은 클라이언트와 서버간의 적절한 기능분화를 통하여 여러 종류의 데이타 처리가 이루어지는 경우로서 분산기능형과 원격데이타관리형이 여기에 해당된다.

(2) 기능층(Function)측면

1) 투자보호

프로그램 투자보호 측면에서 가장 중요하게 고려해야 할 사항은 기존의 메인프레임 환경에서 개발된 막대한 프로그램의 재활용성과 신규개발 프로그램의 이식성이다.

① 기존 프로그램의 재활용도: 1층구조의 경우 기존 프로그램의 수정이 불필요하므로 투자보호 측면에서 가장 효과적이다. 2층구조의 분산표현형은 기존 프로그램의 일부만 수정하면 되므로(대략 20-30%) 비교적 효과적이라 할 수 있다. 2층구조의 분산기능형과 대부분의 3층구조는 기존의 프로

그램 코드를 활용할 수 있으나 해당 코드가 재활용 가능한 형태로 모듈화 되어 있어야 한다. 그 외의 기본유형에서는 프로그램의 재개발과 통합작업이 선행되어야 한다.

② 신규개발 시 이식성: 개방형 환경에서 업계 표준에 따라 프로그램을 개발한다면 양호한 이식성을 보유하게 되어 투자보호 측면에서 유리하다. 메인프레임의 경우에도 최근에는 개방화 추세로 나아가고 있으나 현재로서는 UNIX나 윈도우즈(Windows) NT환경에서 프로그램을 개발하는 것이 투자 보호 측면에서 보다 유리할 것으로 보여진다. 따라서, 2층 및 3층구조 환경에서 프로그램을 개발하는 것이 1층구조에서 개발하는 경우보다 투자보호 측면에서 양호하다.

2) 개발의 용이성

급변하는 경영환경에 탄력적으로 대처하고, 사용자의 다양한 시스템 욕구를 충족시키려면 시스템을 단기간에 쉽게 구축할 수 있어야 한다. 개발의 용이성은 신기술 수용을 통해 개발생산성을 높이고 정보시스템 전반에 걸친 경쟁력을 확보하기 위해 필수적인 요소로서 다음과 같은 항목들을 고려하여야 한다.

① 개발인력의 경험 및 기술보유 정도: 각 유형별로 관련 기술력을 보유하고 있는 인력의 확보 용이성이 기준이 된다. 1층구조를 전통적인 메인프레임으로 보면, 가장 오래된 구조이기 때문에 기술인력의 확보가 쉽다. 2층구조와 3층구조는 새로운 기술이고 또 계속 변화, 발전하고 있는 기술이므로 인력의 확보가 어렵고, 기술 습득 또한 쉽지 않다.

② 4세대 언어(4GL)지원 능력: 4GL을 지원할 수 있는 유형은 2층구조의 분산데이타관리형으로, 이는 2층구조 중 일반적으로 가장 많이 사용되고 있는 구조이다.

③ 개발환경 구축의 용이성: 각 유형 중에서 개발환경의 구축이 가장 쉬운 것은 1층구조이다. 3층구조의 경우는 통합하여야 할 설비들이 많기 때문에 가장 복잡하다.

④ 개발 및 적용의 신속성과 용이성: 개발시점에서 시간이 짧고 용이한가를 따진다.

⑤ 변경 및 보완의 신속성과 용이성: 유지보수가 쉽고 빠르게 수행될 수 있는 정도이다.

⑥ 신기술 적용의 용이성: 신기술의 적용은 폐쇄형보다는 개방형 시스템이 훨씬 유리하다. 따라서 2층구조나 3층구조가 보다 유리하다.

3) 활용범위

활용범위란 해당 프로그램의 지역적 사용범위를 의미하는 것으로 분산/집중 여부의 결정에 필요한 항목이다.

① 전사적 사용: 지역에 관계없이 전사적으로 사용하기에 적당한 유형은 1층구조와 3층구조이다. 2층구조는 LAN 정도의 규모로 사용자수가 많지 않을 경우에 효과적이므로 전사적 사용에는 부적합하다.

② 지역 또는 본사에서만 사용: 일부지역에서만 사용하는 데는 2층구조의 정도가 알맞다. 1층구조를 지역 내에서만 사용하기에는 시스템규모가 너무 크다.

③ 부문간 사용: 지역간, 또는 지역과 본사간의 사용에는 2층구조나 3층구조가 적당하다. 1층구조는 역시 규모가 너무 커서 부적당하다.

4) 사용자 수

프로그램 사용자의 수는 해당 프로그램의 분산 또는 집중여부에 영향을 주며 사용자 수에 따라 적

용 가능한 기본유형도 달라진다. 또한 업무효율을 높이고 사용자 만족을 극대화하려면 사용계층에 대한 특성을 검토하여 이에 적합한 유형을 선정하는 작업이 필요하다.

① 다수 사용: 사용자가 많을 때 적합한 구조는 메인프레임 규모의 1층구조와 3층구조이다. 2층구조는 다수가 사용할 경우 응답속도와 처리속도가 급격히 느려질 수 있으므로 적합하지 않다.

② 소수 사용: 다수가 사용하는 시스템을 소수가 사용하여도 지장은 없으나, 소수 사용자에게 적합한 유형은 2층 구조, 특히 원격데이타관리형과 분산데이타관리형이라 할 수 있다.

③ 경영자 사용: 경영자에게는 의사결정 지원을 위한 어플리케이션을 제공하여야 한다. 이러한 어플리케이션은 GUI를 지원하여야 하고, 정확한 데이타를 제공하여야 한다. 또한 소규모의 네트워크로도 지원 가능하므로 2층구조 중 원격데이타관리형이 적합할 것이다.

5) 처리 형태

프로그램이 어떠한 처리특성을 가지는가에 따라 이를 적용하는 하드웨어 플랫폼이나 관련되는 툴들이 결정된다. 고려할 항목은 다음과 같다.

① 정형적 업무: 업무처리 방식이 고정되어 변하지 않는 경우에는 분산기능형이 적합하다.

② 비정형적 업무: 업무가 변할 때마다 어플리케이션을 수정하는 것은 좋지 않다. 따라서 EUC환경을 구축하여 필요한 데이타를 사용자가 직접 가져오거나, 그래프 등 다양한 형태의 보고서를 작성할 수 있어야 한다. 따라서 적합한 유형은 분산데이타관리형이다.

③ 온라인 처리: 메인프레임 형태의 1층구조는 온라인처리에 적합하다. 3층구조에서는 OLTP 모니터를 사용하여 온라인처리를 효과적으로 구현할 수 있다.

④ 일괄 처리: 일괄처리의 경우도 실시간 처리와 마찬가지로 특정 유형을 선택하기는 어렵지만, 데이타와 기능이 분리되어 있는 유형의 적합도가 다소 떨어진다고 할 수 있다.

6) 트랜잭션(Transaction)

트랜잭션이란 하나의 기능을 처리하기 위하여 수행되는 프로세스들의 집합으로 트랜잭션의 성격에 따라 이를 적절히 지원할 수 있는 기본유형이 달라진다.

① 대량의 트랜잭션: 대량의 트랜잭션 처리는 대용량 처리능력의 1층구조가 적합하다. 3층구조의 경우에는 TP 모니터를 사용하여 대량 트랜잭션을 처리할 수 있다.

② 소량의 트랜잭션: 트랜잭션 수가 적은 경우에는 클라이언트와 서버가 기능층을 나누어 협동처리로 해결할 수 있으므로 분산기능형이 적합하다.

③ 복잡한 트랜잭션: 트랜잭션 처리절차가 복잡한 경우에는 1층구조로 해결할 수 있다.

④ 단순한 트랜잭션: 트랜잭션 처리절차가 단순한 경우에는 1층구조가 너무 무거우므로 기능층을 클라이언트로 분리시킨 유형이 적당하다. 2층구조에서는 원격데이타관리형이나 분산데이타형이 적합하고, 3층구조에서도 클라이언트에 기능층을 위치한 유형이 적당하다.

7) 연관성

다른 프로그램과의 연관성이 깊은 프로그램들은 물리적으로 같은 컴퓨터 내에 있는 것이 유리하므로 1층구조가 가장 적합하다. 가장 부적합한 유형은 2층구조 중 어플리케이션 로직이 클라이언트와

서버에 분산된 분산기능형이라고 할 수 있다.

8) 업무형태

해당 프로그램이 수행하는 작업의 성격에 따라 이를 효과적으로 처리하기 위한 기본유형의 선택이 달라지게 된다.

① 부하가 큰 연산(분석, 시뮬레이션 등): 분석업무나 시뮬레이션 등에는 고속도의 CPU가 필요하므로 클라이언트 보다는 서버의 CPU가 적합할 것이다. 따라서 서버측에 기능층을 두는 유형, 즉 1층구조와, 2층구조의 원격데이타관리형이 적합하다.

② 통계처리: 통계처리 업무는 대부분의 컴퓨터 시스템에서 다룰 수 있는 작업이다. 따라서 모든 유형이 적절한데, 특히 1층구조와 3층구조, 그리고 2층구조 중에서는 분산기능형과 원격데이타관리형이 적합하다.

③ 대량의 데이타 입력: 데이타 입력이 많은 경우에는 GUI가 CUI 보다 오히려 효율이 많이 떨어진다. 따라서 CUI기반의 1층 구조가 뛰어나다 할 것이다.

④ 단순 조회: 데이타 조회 측면에서 보면, HTML 등을 구현하는 2층구조의 원격표현형이 가장 알맞다고 할 수 있다.

⑤ 대량 출력: 대량 데이타의 출력 속도는 텍스트 기반의 1층구조가 가장 빠르다. 2층구조나 3층구조는 DB에서 데이타를 가져오는데 많은 시간이 걸리며, 그래픽 데이타의 출력은 텍스트보다 더욱 늦어진다.

⑥ 최종 사용자 컴퓨팅(EUC): EUC는 멀티태스킹을 지원하는 GUI 환경에서 쉽게 구현할 수 있으므로 2층과 3층구조가 적당하다 할 것이다.

9) 버전 관리

버전 관리란 프로그램이 변경되어 이를 반영한 수정본을 적용 또는 배포시켜야 할 경우에 대한 대비책으로 프로그램 관리 측면에서 중요한 고려사항 중 하나이다.

① 빈번한 프로그램 변경 요구: 사용자의 사소한 요구사항 변화에 따라 수시로 어플리케이션을 수정하여야 한다는 것은 최종 사용자 컴퓨팅(EUC)이 되지 않고 있다는 의미와 같다. 따라서, 1층구조에이 같은 경우가 많고, 2층구조와 3층구조는 상대적으로 적다.

② 프로그램 배포의 용이성: 서버측에 프로그램이 존재하고 있는 경우에는 서버 프로그램만을 수정하면 되므로 어플리케이션 수정시 배포가 가장 쉽다. 즉 기능층이 서버에 있는 유형이 가장 적합하다.

10) 융통성

프로그램 수정이 지속적으로 요구되는 경우, 이에 대한 탄력적 대응은 업무생산성 측면에서 매우 중요하며 이를 위한 효과적 지원책이 필요하다.

① 절차가 자주 바뀌는 업무: 사용자가 데이타를 직접 조작하기에 가장 유리한 유형을 선택하는 것이 EUC 측면에서 어플리케이션의 융통성을 높이는 방법이다.

② 다양한 패키지 활용: 업무처리를 위해 사용가능한 패키지 프로그램의 종류와 수가 어느 정도 되는지가 평가의 기준이다. 적용 가능한 패키지 프로그램의 종류와 수는 현업에서 가장 많이 구현된 기본유형이 가장 다양하고 많을 것이다.

11) 응답시간

프로그램에 대한 응답시간은 업무생산성을 결정

하는 중요한 요소로서 특별히 빠른 응답시간이 요구되는 시스템에 대해서는 이를 충실히 지원할 수 있는 환경과 자원을 할당해야 하며 또한 적합한 기본유형의 선정을 통하여 요구되는 응답시간을 보장해 주어야 한다. 일반적으로 고속의 CPU를 갖추고, 네트워크 등의 설비에 의한 영향을 가장 적게 받는 유형이 가장 신속하고 안정된 응답시간을 보장한다고 할 수 있다.

(3) 데이터(Data)측면

1) 기존의 DB 활용

현재 기존의 메인프레임 환경에서 새로이 클라이언트/서버 시스템을 구현하고자 할 때 가장 손쉬운 방법은 표현층만을 GUI 형태로 개선하는 것이다. 이 경우 기존 DBMS 체계에 변경을 가하지 않고 구현 가능한 사용자 인터페이스 개선만으로도 어느 정도 효과를 기대할 수 있다. 이러한 어플리케이션의 경우에는 기존의 DBMS를 이용할 수 있다. 그러나 변경이 빈번한 데이타의 경우는 새로운 DB형태로 전환하는 것이 바람직하다. 현재의 DB를 24시간 계속하여 사용하는 어플리케이션의 경우는 DB변경이 어려우므로 기존 DB를 그대로 이용하는 것이 바람직하다.

2) 입출력

어플리케이션에서 입력, 수정, 조회, 출력이 이루어지는 빈도에 따라 분산유형을 결정한다. 월초, 월말의 보고용 자료, 평상시 조회수, 그리고 출력 데이타의 양도 함께 고려되어야 한다.

① 수정빈도: DB가 분산되어 있으면 데이타의 수정빈도는 떨어진다.

② 데이타 량: 데이타 량이 많은 경우에 적합한 유형은 네트워크 트래픽에 의한 영향이 가장 적고, 또 한번에 많은 데이타를 가져올 수 있는 유형이다.

3) 데이타베이스 크기

단위 레코드의 길이가 길거나 건수가 많아 데이타베이스의 크기가 커지면 어플리케이션 특성에 따라 통합 또는 분산의 방향을 결정해야 한다.

4) 데이타의 처리량

어플리케이션에서 일정주기내에 처리하는 데이타 수는 시스템의 성능에 영향을 미친다.

① 일정주기내에 대량의 데이타 처리: 데이타가 분산되어 있는 것보다는 집중되어 있는 경우와 대용량의 컴퓨터인 경우가 보다 적합하다.

② 일정주기내에 소량의 데이타 처리: 대량 데이타를 처리할 수 있는 유형도 가능하나, 효율성을 고려한다면 가급적 데이타를 분산시켜 적은 양을 유지할 수 있는 유형이 더 적합할 것이다.

5) 사용 데이타의 범위

어플리케이션 운영시 사용되는 데이타가 DB 내의 전체 데이타인가 또는 부분적 데이타인가에 따른 기준이다. DB 전체의 데이타를 사용하기에 보다 효과적인 것은 모든 DB가 물리적으로 근접한 위치에 있는 경우이다.

6) 데이타의 활용주기

DB내의 데이타를 얼마나 자주 업데이트(Update)하는가와 관련된 기준이다.

7) 데이타의 이질성

어플리케이션에서 처리하는 데이타의 성격에 따른 기준이다. 전사적으로 통용되지 않고 지역별로

이질적인 데이타를 사용하여 별도관리를 하는 어플리케이션의 경우는 분산처리가 바람직하다. 단, 계획수립업무와 같이 각 지역별로 이루어지지만 하나의 어플리케이션으로 처리가 가능한 경우는 분산하지 않아도 좋다.

8) 어플리케이션이 사용하는 DB의 수

어떤 어플리케이션은 작업처리를 위하여 많은 수의 DB를 동시에 사용하는 경우가 있다. 이와 같이 여러 개의 DB에 동시로 접근하여 사용하는 경우, 사용 DB 수에 별로 영향을 받지 않는 유형은 1층구조이다. 왜냐하면, 여러 DB가 물리적으로 한 컴퓨터 내에 있기 때문에 접근에 문제가 없기 때문이다.

9) 데이타 가용성(Data Availibility)

특정기간 동안에 데이타를 사용할 수 있는 시간의 비율을 말한다. 예를 들어 하루동안 24시간 사용할 수 있는 데이타라면 가용성은 1이다. 데이타 가용성에 영향을 미치는 요소로는 데이타의 종류와 사용자의 위치 및 분포 등이 있다.

① 사용자가 한 지역에 집중되어 있는 경우: WAN보다는 LAN이 적합하므로 LAN구축시 가장 효율이 높은 유형을 선택한다.

② 사용자가 여러 지역에 분산되어 있는 경우: 2층구조 중 원격데이타관리형과 분산데이타형은 LAN정도의 소규모 그룹에 적당한 유형이므로 이 경우에는 적합하지 않은 유형이다.

10) 데이타 무결성(Data Integrity)

데이타 무결성을 위하여는 데이타의 중복을 배제 또는 최소화해야 한다. 단일 DB를 사용한다면 중복이 최소화될 수 있을 것이고 여러 개의 DB가 다수의 컴퓨터에 존재한다면 중복성 배제는 어려워질 것이다. 따라서, 데이타 무결성을 위하여는 집중구조를 가진 데이타 유형이 보다 적합하다. 또한 다수의 DB가 여러 컴퓨터에서 운용되는 경우, 하나의 DB에서 갱신되는 내용이 다른 컴퓨터에 있는 DB에 전달되는 시간을 최소화할 수 있는 유형이 보다 적합하다.

11) 중복성(Redundancy)

데이타의 중복은 무결성유지를 위하여는 바람직하지 않다. 그러나, 시스템의 성능을 높이기 위해, 또는 컴퓨터 간의 호환성 등, 구현상의 문제로 실무에서는 중복을 허용하는 경우가 있다. 이처럼 통제된 중복의 허용은 응답시간을 향상시킨다. 2층구조 중, 분산데이타형은 클라이언트와 서버에 데이타를 분산시켜 사용자가 서버의 상태와 무관하게 클라이언트에 있는 데이타에 접근할 수가 있다.

(4) 관리(Management)측면

1) 보안관리

보안관리 측면에서는 일반적으로 폐쇄형인 메인프레임의 경우가 개방형 시스템에 비하여 다음과 같은 이유로 우수하다.

- 대부분의 H/W가 통제된 전산실 또는 기계실에 위치하여 물리적 보안이 용이하다.
- 주요 보안기능이 마이크로코드(Microcode) 및 O/S차원에서 제공되므로 시스템 S/W에서 어플리케이션까지 통합적인 보안기능을 제공한다.
- 다단계 시스템 보안등급을 적용하여 사용자별로 보다 융통성 있는 보안체계를 구축할 수 있고 통제가 용이하다.
- H/W는 물론 시스템 자원 전반에 대한 보안기능을 제공한다.

· 보안기능의 구현이 용이하므로 적은 인력으로도 효율적인 구축이 가능하다.
· 정보자원이 집중되어 있고 시스템구성이 단순하여 보안체제의 운영이 용이하다.

따라서 ①구축의 용이성과 ②운영의 신뢰성 측면에서, 메인프레임을 이용하여 정보자원을 집중 관리하는 1층구조가 우수하다. 그러나 물리적 보안의 경우 보안 대상이 통제구역에 위치하여 관리된다면 유형별로 별다른 차이점을 갖지는 않는다. 3층구조의 경우 데이타가 메인프레임 또는 DB서버에 위치하므로 신뢰성있는 데이타보안이 가능하다고 할 수 있다.

2) 백업 및 복구(Back Up & Recovery)

일반적으로 데이타가 하나의 시스템에 집중되어 있는 경우, 장애발생시 전체데이타가 소실될 우려가 있다. 그러나 1층구조의 경우 데이타가 집중되어 있으나, 메인프레임이 다양한 복구기능을 제공하므로 데이타 복구 기능은 우수하다고 할 수 있다.

3) 유지보수의 용이성: 유지보수가 용이한가의 여부를 말한다.

4) 지원(Support)

① 구성관리의 용이성: Configuration Management가 용이한가의 여부

② Vendor 활용의 용이성: 공급자로부터 지원을 받을 수 있는가의 여부

5) 설비

관련되는 설비를 가장 적은 비용과 인력으로 관리할 수 있는 방안을 도출하고 이를 적용하여 관리비용을 최소화하는 방향으로 기본유형을 선정한다.

① 가격대 성능비 향상: 일반적으로 메인프레임보다는 UNIX나 NT 환경의 서버급 컴퓨터가 가격대비 성능면에서 뛰어나다.

② PC성능 활용: PC의 성능을 활용한다는 것은 EUC환경을 구축하는 것과 관계가 깊다. 따라서 데이타서버로부터 직접 데이타를 가져와서 PC의 기능을 이용하여 업무처리에 적용하는 것이 PC성능 활용면에서 좋다.

③ 시스템 확장성: 추가적인 확장이 필요할 때 어떤 것이 좋은가의 여부.

2. 적용기준표의 작성과 해석

지금까지의 논의를 토대로 적용기준표를 작성하면 첨부된 표와 같다. 표에서 부여된 항목별 숫자의 의미는 다음과 같다.

· -2 : 해당항목을 그 유형으로 구현하는 것이 매우 나쁜 경우
· -1 : 해당항목을 구 유형으로 구현하는 것이 나쁜 경우
· 0 : 해당항목이 그 유형과 관련이 없거나 그 유형으로 구현하는 것이 좋고 나쁨을 판별하기 어려운 경우
· 1 : 해당항목을 그 유형으로 구현하는 것이 좋은 경우
· 2 : 해당항목을 그 유형으로 구현하는 것이 매우 좋은 경우

적용기준표에 따라 각 유형에 적합한 업무의 특성을 기술하면 아래와 같다.

(1) 1층구조의 적용기준

1층구조는 오직 한 장소에서만 사용되는 어플리케이션이나, 보안관리상 분산 컴퓨팅이 적절하지 않은 경우에 적합한 방식이다. 또한 대형시스템의 컴퓨팅 파워를 이용할 필요가 있거나 네트웍의 이용이 많지 않은 경우에도 1층구조는 적절한 해결책이라고 할 수 있다. 그러나 기업의 정보처리 요구사항의 지속적 증가와 개방 시스템의 등장 및 확산에 따라 1층구조에 대한 수요는 점차 줄어들고 있는 상황이다.

(2) 2층구조의 적용기준

1) 분산표현형: 호스트와 같은 시스템에서 프로그램 로직을 변경하지 않고 최종사용자에게 그래프를 보여주는 수준의 그래픽 환경을 제공하고자 하는 업무에 적합하다.

2) 원격표현형: 서버나 호스트의 기능과 데이타는 유지하면서 클라이언트 부분만 GUI를 사용하려는 업무에 적합하다. Thin-Client 개념으로 보면, 기능과 데이타 관리를 서버에 두고 클라이언트의 부담을 줄여 소프트웨어 버전 관리 등을 쉽게 하고자 할 경우가 여기에 해당된다. 또한 웹 브라우저를 사용한 인트라네트 환경을 갖추는 경우에도 좋은 구현방법이다.

3) 분산 기능형: 분산기능형은 원격데이타 관리의 확장된 형태인 Stored Procedure/ Trigger를 이용할 경우와 Middleware를 이용하여 구축하는 경우로 크게 나눌 수 있다. Stored Procedure/ Trigger를 이용하는 형태는 원격데이타관리와 같은 적용 요건을 가지고 있다. 즉,

〈표 5〉 분산기능형의 적용요건

요건	Stored Procedure/ Trigger	Middleware
클라이언트의 위치	· LAN 환경에 적합	· LAN/WAN 환경에서 모두 가능
개발기간	· 비교적 짧은 개발기간	· 설계시 많은 시간이 요구됨
적용대상 응용업무	· 일반업무	· 일반업무 · 빠른 응답을 요구하며 사용자 수가 많은 OLTP에 가까운업무 · 사용자가 분산/집중 데이터 관리에 적용하기 좋음
데이터의 보안등급	· 중간정도	· 보안등급이 높은 업무구현 가능
처리속도	· 중간정도	· 빠른 응답요구 충족가능

- LAN 환경에서 사용하기 적합하다.
- 비교적 짧은 개발기간을 요하는 업무에 적합하다.
- 데이타의 보안등급이 중간정도인 업무에 적합하다. (보안등급이 높은 업무를 적용하기 위해서는 좀 더 많은 노력을 기울여야 한다)
- 보통의 처리속도를 요구하는 업무에 적용하기에 적합하다.

Middleware를 이용하여 업무를 구축하는 경우 클라이언트는 WAN 구간에서도 이용 가능하다. 일반업무 중 지역적으로 여러곳에 분산되어 있으나 데이타는 한 곳에 집중관리해야 하는 경우 Middleware를 이용하여 구축하는 것이 바람직하다. 대량의 데이타 처리와 다수의 사용자가 동시에 작업하는 환경은 OLTP환경과 가깝다. OLTP환경은 3층구조로 구현하기에 적합하나, 사용자의 수나 데이타의 양 등이 비교적 적은 경우에는 Middleware를 이용한 분산기

능형으로 구현하는 것이 바람직하다. Middleware를 이용하는 방식은 비교적 빠른 응답시간을 충족시켜 줄 수 있는 구조를 가지고 있다. 데이타의 관리와 접근을 중앙에서 집중적으로 할 수 있기 때문에 비교적 높은 보안성을 갖는다.

4) 원격데이타관리형: 원격데이타관리형은 LAN 환경에 적합한 방식이다. RDBMS의 경우 SQL을 이용하여 데이타를 가져온 후 클라이언트에서 데이타를 가공하므로 네트워크 부하가 증가한다. 따라서 원격지 사용자가 네트워크를 통해 서버에 접속하여 사용하는 데이타 량이 많을 경우 처리 속도가 늦어진다. 통상적으로 64Kbps 선로를 이용할 경우 10명 이상의 사용자가 사용하기에는 부적합하다.

.모든 업무절차와 표현층이 클라이언트에 있으므로 업무부분과 DBMS 설계만으로도 구축이 가능하다. 따라서 단순한 업무나 개발기간이 짧은 업무 등을 적용하기에는 좋은 모형이다. 보고서 작성도구를 이용하여 업무를 구축할 경우, 혹은 데이타 웨어하우스를 구축하는 업무는 초기 데이타를 어떻게 구축할 것인가가 가장 중요한 부분이다. 원격데이타관리형은 대부분의 일반 업무에 적용할 수 있는데 특히 데이타 웨어하우스는 중역정보시스템(EIS)이나 의사결정지원시스템(DSS) 또는 통계/분석업무에 적용하기도 좋다. 또한 보안 측면에서는 그 중요성이 그다지 심각하지 않은 경우에 적합한 형태라고 할 수 있다.

5) 분산데이타관리형

클라이언트를 독립적으로 운영할 수 있으며, 클라이언트가 다루어야 할 데이타를 단순하게 관리할 수 없는 데이타일 경우 사용 가능하다. LAN환경에서는 사용하기 편리하나, WAN에서는 서버의 데이타를 자주 이용하지 않는 경우에 적합하다.

〈표 6〉 원격데이타관리형의 적용요건

요 건	1st Generation Tool	Reporting Tool(Data Warehouse)
클라이언트의 위치	· LAN 환경에 적합 · Remote 환경에서 64K line의 경우, 동시 사용자가 10명까지 가능	· LAN 환경에 적합 · Remote 환경의 경우 사용자수가 적고 데이터가 소량일 경우, DBMS를 원격에서 구축하는 것이 바람직
개발기간	· 개발기간이 짧은 경우에 적용하기가 용이	· DB를 구축하는 것이 주요 사항임 · DB가 구축된 단계에서는 개발기간이 짧아짐
적용 응용업무	· 단순한 application (단일부서 업무이거나 동시 사용자 수가 100명을 넘지 않는 업무에 적합)	· DSS(Decision Support System) · 통계, 분석
데이터 및 응용업무의 보안	· 응용업무에서 데이터의 보안이 특별히 중요하지 않는 경우	· 응용업무의 데이터가 특별히 중요하지 않는 경우 · 특정 부서에서만 보안이 요구되는 데이터를 다룰 수 있는 경우

〈표 7〉 분산데이타관리형의 적용요건

요 건	분산데이터관리형
클라이언트의 위치	· LAN환경 · WAN환경은 서버의 데이터 접근이 적을 경우 사용 가능
개발기간	· 보통 또는 짧음
Application 형태	· 비교적 독립적으로 운영가능한 업무
데이터 및 Application의 보안등급	· 보안등급이 낮은 업무에 적용 가능
처리속도	· 보통 (클라이언트의 성능에 의해 결정됨)

(3) 3층구조의 적용기준

일반적으로 3층구조는 전사적으로 처리해야 할 기능이 있는 경우, 이를 지역 서버와 엔터프라이즈 서버에 나누어 공통 적용하거나(P-F-FD형), 업무 내용 자체는 변동이 없지만 트랜잭션 처리 속도의 신속성, 응답시간, 가용성, 보안 요구사항 등의 측면에서 트랜잭션 하나 하나가 업무에 지장을 초래할 수 있는 경우, 기존시스템과 새로 도입할 시스템이 상이한 경우, 트랜잭션 중심의 온라인 업무, 사이트 수가 많은 경우, 데이타의 폭주와 동시 사용자 수가 많은 경우, 그리고 전사적으로 적용할 업무 등에 적합하다.

1) P-F-*D: 전사적으로 처리되어야 할 기능이 있는 경우 지역 서버와 엔터프라이즈 서버에 기능을 두어서 공통 적용한다.(P-F-FD형)

2) PF-F-*D: 클라이언트와 지역 서버, 엔터프라이즈 서버, 모두에서 처리할 기능이 있는 경우에 적용한다. 클라이언트의 기능층은 입출력되는 데이타의 정확성(Integrity)검사를 위한 것이며 지역 서버의 기능층은 공통업무 처리 기능이고, 엔터프라이즈 서버는 데이타 서버 역할을 하는 동시에 데이타의 추출 및 관리 업무를 처리한다. 정형적 업무뿐만 아니라 비정형적이고 수시로 변하는 사용자의 요구를 만족시키기 위한 EUC 환경도 지원한다.

3) P-FD-D: 본 유형은 두 가지 방식으로 적용가능하다. 첫번째 방식은, P-FD-D형을 3층구조인 P-F-D 형의 변형으로 보아 지역서버와 엔터프라이즈 서버에 데이타를 두는 방식이다. 지역서버에 지역 특유의 데이타를 두고 엔터프라이즈 서버에는 전사 공통 데이타를 두는 방법이다. 두번째 방식은 2층구조인 원격표현형의 변형으로 엔터프라이즈 서버의 데이타를 실시간, 또는 일괄처리 방식으로 지역 서버에 복제받아 2층구조와 같은 방법으로 사용하는 것이다. 전사 공통 자료는 엔터프라이즈 서버로, 지역 자체적으로 관리해야 데이타는 지역 서버로 관리하는 경우에 적용한다.

4) PF*-D-D: 이 유형은 단순 업무나 개발기간이 짧은 업무, 또는 데이타나 업무에 대해 보안요구가 그다지 높지 않은 경우에 적용하기 좋다. 업무 형태가 복잡하거나 많은 사용자를 지원해야 하는 업무는 설계와 개발에 시간과 노력을 들더라도 확장성이 좋고 또 유지보수가 유연한 PF-FD-D형

또는 P-FD-D형으로 구현하는 것이 바람직하다.

5) PFD-D-D: 표현, 기능 및 데이타관리가 모두 클라이언트에서 이루어지고, 데이타가 클라이언트, 지역서버, 그리고 엔터프라이즈 서버에 분산된 형태이다. 따라서 클라이언트를 독립적으로 운영하면서 필요시 지역서버와 엔터프라이즈서버의 데이타와 연계하여 처리해야 하는 업무에 적용한다. 지역적으로 떨어져 있는 여러 곳의 자료를 참조하는 업무나 워크플로우(workflow)적 특성을 갖는 업무에 적용가능하다. 예를 들어, 결재, 문서회람 등과 같이 연속처리업무를 자동화시킬 때는 서로 공유하는 공통 문서를 여러 타 업무시스템에서 공유하여 가공할 수 있어야 한다.

V. 결 론

클라이언트/서버 시스템이 모든 문제를 해결해 주는 만병통치약이 될 수는 없다. 그러므로 새로운 시스템을 구축하려는 조직과 개발전문가는 왜 클라이언트/서버 시스템을 구축하려고 하는지에 대한 이유를 찾아야 한다. 업무의 특성에 따라서는 클라이언트/서버 환경이 맞지 않을 수도 있으며, 또 설사 잘 맞는다 하더라도 가능한 여러 가지 대안 중에서 실제 적용할 유형을 결정하려면, 적용업무의 특성으로부터 나오는 시스템구축의 이유가 명확해야 한다. 기존의 클라이언트/서버 시스템에 대한 연구는 기술적인 측면에만 치우쳐, 이같이 기본적인 문제에 대한 해답을 제공하는데 미흡하였다. 따라서 본 논문에서는 이 같은 문제를 해결하기 위하여 물리적 소프트웨어 층과 논리적 소프트웨어 층의 개념을 이용하여 1, 2, 3 층 구조에서 가능한 분산 어플리케이션의 기본유형을 논리적으로 도출하고 이를 응용업무에 적용하기 위한 지침으로서 적용기준표를 작성, 제시하였다.

본 연구의 결과를 요약하면 다음과 같다. 첫째, 표현, 기능, 데이타관리로 구성된 어플리케이션을 클라이언트/서버 환경에서 분할하는 기본유형은 1, 2, 3층 구조를 모두 합하여 총 14 개의 종류가 가능하다. 또한, 동일한 기본유형이라 할지라도 이를 물리적인 시스템으로 구현하는 경우에 사용기술에 따라 가능한 방법은 여러 가지가 있을 수 있다. 둘째, 응용업무의 적용기준을 표현, 기능, 데이타, 운영 관리의 4가지 측면에서 고찰하였다. 따라서 시스템 개발자들은 응용업무의 특성을 본 논문에서 제시한 적용기준표에 맞추어 비교, 평가함으로써 그 업무에 가장 적합한 환경의 분산 어플리케이션 유형을 선택할 수 있다. 셋째, 응용업무에 적합하지 않은 유형으로 구현된 시스템은 향후의 유지, 보수와 신규 시스템의 개발, 접목 시에 많은 어려움에 직면하게 될 것이다. 따라서 시스템 개발자들은 어떤 기술을 사용할 것인지를 결정하기에 앞서 개발 대상 업무의 특성과 그로부터 도출되는 요구사항을 명확히 파악해야 하며, 이를 바탕으로 최적의 클라이언트/서버 유형을 결정해야 할 것이다.

본 연구의 최대 공헌은 클라이언트/서버 환경으로 분산 컴퓨팅시스템을 구축하려는 조직 및 시스템개발 전문가들로 하여금 적용업무의 특성에 따라 최적의 분산유형을 결정할 수 있도록 실질적인 가이드라인을 제시하여 준다는 점에 있다. 본 연구에서 지침으로 제시한 적용기준은 모든 종류의 조직 업무에 일반적으로 적용할 수 있을 것이다. 클라이언트/서버 환경으로 정보시스템을 구축하려는 기업 및 조직체들은 본 연구결과를 활용하여 더 이상 몸에 맞지 않는 옷을 입는 것과 같이 불편하고 어색

한 모습을 보이지 않고 업무특성에 따른 최적 시스템을 구현할 수 있으며 유지보수나 신규시스템 추가 접목 시에 겪는 여러 어려움 들을 감소시킬 수 있을 것으로 기대된다.

본 연구의 한계점으로는 적용기준 항목 선정에 있어서, 일반적으로 고려할 수 있는 대부분의 항목들을 포함시켰지만 전체가 망라되어 있는 완전한(Complete) 상태라고 보기 어려우므로, 앞으로 컴퓨팅환경의 변화 및 신기술의 등장과 더불어 적용기준의 항목들을 지속적으로 보완하여야 할 것이라는 점을 지적할 수 있다.

본 연구의 후속작업으로 이루어져야 할 것은 본 연구에서 이론적으로 제시한 분산 어플리케이션의 기본유형과 이를 적용하기 위한 기준을 실제의 시스템 구축사례에 적용하여 실증적 타당성을 검증함으로써 본 연구 내용의 객관성과 활용가능성을 한층 높이는 작업이다. 실제 사례에 적용하는 방식에는, 시간적 차원으로는 이미 구축 완료된 시스템개발 사례를 응용하는 사후적 방식과 현재 진행 중이거나 아직 착수하지 아니한 시스템개발 사례를 이용하는 사전적 방식이 있을 수 있고, 또 내용적 차원으로는 적용가능성 만을 검증해보는 소극적 방법과 적용가능성 만이 아니라 기준표의 정확성과 타당성까지 검증하는 적극적 방법이 있을 수 있다.

향후 연구에서는 주로 사후적 방식과 적극적 방식을 채택하여 작업을 진행하고자 한다. 사전적 방식인 경우 주로 민간 개발업체에서 수행하는 시스템 개발 프로젝트에 연구자들이 공동으로 참여하기가 쉽지 않으며 또한 프로젝트의 업무분석 및 설계단계를 본 연구의 수행일정과 맞추기가 어렵다는 현실적 어려움 때문이다. 본 연구결과에 대해 소극적으로 실제사례의 적용가능성만을 입증하는 데에는 큰 어려움이 없을 것으로 보인다. 그러나 연구결과에 대한 보편성을 확보하고 도출된 기준표에 대한 내용면에서의 타당성을 입증하여 본 연구의 학술적 공헌도를 높이기 위하여는 적극적 방식이 보다 바람직하다고 생각한다. 이러한 작업을 위한 사례의 선정은 접근가능성 등의 현실적 여건을 고려하여 연구자들이 과거에 수행하였거나 참여한 바 있는 클라이언트/서버 시스템 개발사례를 대상으로 하려고 한다.

참 고 문 헌

김영걸, 박영민 (1997), "기업의 정보처리 특성과 클라이언트-서버 아키텍춰 구현전략에 관한 연구", **한국정보처리학회논문지**, 제4권 제5호.

Patrick Smith & Steven Guengerich 저 정성권, 이재원 역, 클라이언트/서버 컴퓨팅 2판, **동일 출판사**, 1995

Berson, A. (1994), *Client/Server Architecture*, McGraw-Hill, 1994.

Byron, D. (1995), "Issues in On-Line Transaction Processing", *DATAPRO UNIX & Open Systems, Technology Concepts*, November 1995.

Byron, D. (1995), "Open Software Foundation's Distributed Computing Environment (DCE)", *DATAPRO, UNIX & Open Systems Service, Standards*, IU07-700, August 1995.

Byron, D. (1995), "Understanding Distributed Computing", *DATAPRO UNIX & Open System Service, Concepts*, IU05-700, May 1995.

DATAPRO (1992), "The Best in Client/Server Computing", supplement to Datamation October 1, 1991, DATAPRO, *Management Information Technology*, Technology Issues & Trends,

1223, June, 1992.

Dewire, D. (1993), *Client/Server Computing*, McGraw Hill, 1993.

Gartner Group (1994), "Advantages of Multiple Tier in Client/Server Computing", *Gartner Group Research Note*, October 12, 1994.

Gartner Group (1994), "More Client/Server Tiers, But Five-Styles Model Remains Valid", *Gartner Group Research Note*, October 12, 1994.

Guttman, M. K. and Matthews, J. R. (1993), "Client/Server Computing: Emerging Trends, Solutions, and Strategies", *DATAPRO, Managing Information Technology, Technology Issues & Trends* 1222, January 1993.

IBM (1994), "A Guide to Open Client/Server", *Open Enterprise Group of IBM Europe*, April, 1994.

Lauletta, P. A. (1994), "The Hidden Cost of Client/Server Computing", *DATAPRO UNIX & Open System Services, Concepts* IU05-725, July 1994.

Long, C. (1995), "Middleware: Overview", *DATAPRO UNIX & Open System Technology Concepts*, 9200, November 1995.

Lyons, F. W. (1994), "An Introduction to Client/Server Security", *DATAPRO Managing Information Technology, Security* 6038, November 1994.

Mackenzie, J. (1995), Document Repository, *Special Report, Byte*, April, 1995.

Nutt, G. (1994), *Open Systems*, Prentice Hall, 1994.

Orfali, R. (1995), Intergalatic Client/Server Computing, *Special Report, Byte*, April, 1995.

Orfali, R., Harkey, D., and Edwards, J. (1996), *Essential Client/Server Guide*, Wiley, 2nd ed., 1996.

Paul, P. M. (1994), "Client/Server Middleware: Solutions for Integration", *DATAPRO UNIX & Open Systems Services, Concepts* IU05- 725, July 1994.

Renaud, P. E. (1996), *Introduction to Client/Server Systems*, New York, Wiley, 2nd ed., 1996.

Schulte, (1995), "Two-Tier vs. Three-Tier Trade-Offs", *Gartner Group Research Note*, January 18, 1995.

Sinha, A. (1992), "Client/Server Computing: Current Technology Review", pp. 77-98, CACM, July 1992.

Stead, J. (1995), *Open, Cooperative Computing Architecture (OCCA)*, AT&T GIS, 1995.

Tiedrich, A. H. (1994), "Client/Server Client Development Tools", *DATAPRO Technology Issues & Trends 1224*, April 1994.

Umar, A. (1993), *Distributed Computing and Client-Server Systems*, Prentice Hall, 1993.

부표: 분산 어플리케이션 기본유형의 적용기준표

대항목	소항목	세부항목	1층	DP	RP	DF	RD	DD	P-F-D	PF-F-D	P-F-FD	PF-F-FD	P-FD-D	PF-FD-D	PF-D-D	PFD-D-D
표현층	종류	다중창, 다중처리 지원정도	-2	-1	0	1	1	1	2	2	2	2	2	2	2	2
		음성처리 지원	-2	-1	1	1	1	1	1	1	1	1	1	1	1	1
		화상처리 지원	-2	1	1	1	1	1	1	1	1	1	1	1	1	1
		화상회의 지원	-2	-1	-2	2	2	1	-1	2	-1	2	1	2	2	2
기능층	투자보호	기존 프로그램의 재활용도	2	1	0	-1	-1	-1	-2	-2	-2	-2	-2	-2	0	-1
		신규개발시 이식성	-2	-1	1	1	1	1	2	2	2	2	2	2	1	1
	개발의 용이성	개발 인력의 경험 및 기술보유 정도	2	1	-1	0	1	-1	-2	-2	-2	-2	-1	-1	1	0
		4세대 언어(4GL)지원능력	-2	-1	-1	1	2	1	-1	-1	1	1	-1	-1	2	1
		개발환경구축의 용이성	2	1	1	-1	0	-2	-2	-2	-2	-2	-2	-2	0	-1
		개발 및 적용의 신속성과 용이성	-2	-1	0	-1	2	-1	1	1	-1	-1	-1	-1	1	-1
		변경 및 보완의 신속성과 용이성	-2	-1	0	1	1	1	2	2	2	2	2	2	1	1
		신기술 적용의 용이성	-2	-1	1	1	1	1	2	2	2	2	2	2	1	1
	활용범위	전사적 사용	2	1	0	1	-1	-2	2	2	2	2	2	2	2	2
		지역 또는 본사에서만 사용	-2	-1	1	1	2	2	-1	-1	-1	-1	1	1	1	2
		부문간(지역간, 지역과 본사) 사용	-2	-1	1	1	0	1	2	1	1	1	1	1	1	1
	사용자	다수 사용	2	1	1	1	-1	-1	2	2	2	2	2	2	-1	-2
		소수 사용	0	0	0	0	2	2	0	0	0	0	0	0	2	2
		경영자 사용	-2	-1	1	0	2	2	0	0	0	0	0	0	1	0
	처리형태	정형적 업무	0	0	0	2	0	0	2	2	2	2	2	2	1	0
		비정형적 업무	-1	-1	-1	-2	2	1	-2	-1	-2	-1	-2	-1	2	1
		온라인 처리	2	2	0	1	0	0	1	1	1	1	0	1	0	0
		일괄 처리	2	1	1	0	-1	-1	-1	-1	0	0	-1	-1	-2	-2
	트랜잭션	대량의 트랜잭션 처리	2	2	-2	0	-1	-1	2	2	2	2	2	2	-2	-2
		소량의 트랜잭션 처리	-1	-1	0	2	1	1	0	0	0	0	0	0	1	1
		복잡한 트랜잭션 처리	2	2	1	1	0	-1	1	1	1	1	1	1	-2	-2
		단순한 트랜잭션 처리	-1	-1	1	2	2	2	0	0	0	0	0	0	1	1
	연관성	타 시스템과의 연관성이 깊다	2	2	1	0	0	0	1	1	1	1	1	1	0	0
	업무형태	부하가 큰 연산(분석, 시뮬레이션 등)	2	2	1	0	2	-1	1	1	1	1	1	1	2	-1
		통계처리	2	2	1	2	2	1	2	2	2	2	2	2	2	1

대항목	소항목	세부항목	1층	DP	RP	DF	RD	DD	P-F-D	PF-F-D	P-F-FD	PF-F-FD	P-FD-D	PF-FD-D	PF-D-D	PFD-D-D
기능층		대량의 데이타 입력	2	1	-1	1	-1	-2	1	1	1	1	-2	-2	-2	-2
		단순 조회	1	1	2	1	1	2	1	1	1	1	1	1	1	2
		대량 출력	2	2	1	1	-1	-2	1	1	1	1	1	1	0	-2
		최종사용자컴퓨팅(EUC)	-2	-1	-1	1	2	2	-1	2	-1	2	-1	2	2	2
	버전관리	빈번한 프로그램 변경요구	2	1	1	-1	-2	-2	2	-1	2	-1	2	-1	0	-2
		프로그램 배포의 용이성	2	1	1	-1	-2	-2	1	-1	1	-2	2	-1	-2	-2
	융통성	절차가 자주 바뀌는 업무	-2	-1	1	1	1	2	1	-1	0	-2	1	-1	0	0
		다양한 패키지 활용	1	0	1	1	2	1	1	1	1	1	1	1	2	1
	응답시간	응답시간	2	1	1	1	0	1	2	2	2	2	2	2	0	1
데이타	기존DB 활용측면	기존 DB의 활용가능도	2	2	1	-1	-2	-2	1	1	1	1	1	1	2	2
	입출력	수정빈도	2	2	1	0	1	-1	-1	-1	-1	-1	-2	-2	-2	-2
		많은 데이타 량	2	1	1	0	-1	-2	0	0	0	0	-1	-2	-1	-2
	데이타베이스 크기	대형 데이타베이스	2	2	1	0	-1	-2	0	0	0	0	-1	-1	-1	-2
	데이타의 처리량	일정주기내에 대량의 데이타 처리	2	2	1	-1	-1	-2	0	0	0	0	-1	-1	-2	-2
		일정주기내에 소량의 데이타 처리	0	0	0	1	1	2	0	0	0	0	-1	-1	1	2
	사용하는 데이타	DB내 모든 레코드 사용	2	2	2	0	0	-2	1	1	1	1	-2	-2	-2	-2
	데이타의 활용주기	수시 Update	2	2	2	-1	-2	-2	1	1	1	1	-2	-2	-2	-2
	데이타의 이질성	여러 DBMS를 사용(Oracle, Sybase등)	-2	-2	-2	2	1	1	2	2	2	2	2	2	2	2
	사용하는 DB의 수	여러 DB를 동시접근(수직분할)	2	2	2	2	0	1	2	2	2	2	2	2	2	2
	데이타의 가용성	사용자가 한 지역에 집중된 경우	2	2	1	1	2	1	-2	-2	-2	-2	-2	-2	1	2

대항목	소항목	세부항목	1층	DP	RP	DF	RD	DD	P-F-D	PF-F-D	P-F-FD	PF-F-FD	P-FD-D	PF-FD-D	PF-D-D	PFD-D-D
관리		사용자가 여러지역에 분산된 경우	1	1	1	-1	-2	-2	2	2	2	2	2	2	2	-2
	데이타의 무결성	DB의 내용과 실세계와의 일치여부	2	2	1	1	1	-1	1	1	1	1	-1	-1	-1	-1
	중복성	성능향상 위한 중복성 필요 여부	0	0	0	0	1	2	0	0	0	0	1	1	1	2
	보안관리	구축의 용이성	2	1	1	0	-1	-2	1	1	1	1	1	1	-1	-1
		운영의 신뢰성	2	1	1	1	1	1	0	0	0	0	0	0	0	0
	백업 및 복구	백업과 복구의 용이성	2	2	1	0	0	-1	1	1	1	1	0	0	-1	-2
	유지보수	유지보수의 용이성	2	1	1	0	0	0	0	0	0	0	0	0	-1	-1
	지원	구성관리의 용이성	2	0	0	0	0	0	1	1	1	1	0	0	-1	-1
		Vendor 활용의 용이성	2	1	1	0	0	0	1	-1	-1	-1	-1	-1	-1	-1
	설비	가격대 성능비	-2	-1	1	1	1	1	1	1	1	1	1	1	1	1
		PC성능 활용	-2	-1	0	1	2	2	1	2	1	2	1	2	2	2
		시스템 확장성	-2	-1	-1	1	0	0	2	2	2	2	2	2	1	1
합계			30	31	31	30	24	0	37	35	34	34	22	20	22	2

Classifications of and Guidelines for Optimal Distributed Applications under Client/Server Environment

Garp Choong Kim* · Jae Hoon Shin**

Abstract

Client/Server technology is now regarded as a general solution to many problems inherent in legacy systems in the past. New concepts, technologies, and hardware solutions are pouring into the market almost everyday. However, several key questions, ironically, are still left answered and does not get much attention: What are the possible types of the client/server systems? What kind of business applications best fits to the client/server? How about the relationship between business characteristics and types of the client/server? Characteristics of business application determine the optimal type of client/server system. This study aims at providing guidelines for selecting the optimal client/server systems based on characteristics of business application. In doing so, this paper found out that there are fourteen different kinds of client/server systems, categorized the application characteristics into four major and sixty-one detailed items, and generated a five-scale fitness table based upon the experts' opinions between business characteristics and types of client/server systems.

* School of Business Administration, Inha University
** Strategic Planning Group, Samsung SDS