분산 컴퓨팅 환경하에서의 데이타 자원 관리

Data Resource Management under Distributed Computing Environment

서울대학교 경영대학 교수 안 중 호

박사과정 조 희 경

Abstract

The information system of corporations are facing a new environment expressed by miniaturization, decentralization and Open System. It is therefore of utmost importance for corporations to adapt flexibly to such new environment by providing for corresponding changes to their existing information systems. The objectives of this study are to identify this new environment faced by today's information system and develop

effective methods for data resource management under this new environment.

In this study, it is assumed that the new environment faced by information systems can be specified as Distributed Computing Environment, and in order to achieve such system, presents Client/Server architecture as its representative computing structure. This study defines Client/Server architecture as a computing architecture which specialize the fuctionality of the client system and the server system in order to have an application distribute and perform cooperative processing at the best platform. Furthermore, from among the five structures utilized in client/server architecture for distribution and cooperative processing of application between server and client, this study presents two different data management methods under the client/server environment: one is "Remote Data Management Method" which uses file server or database server and the other is "Distributed Data Management Method" using distributed database management system.

The result of this study leads to the conclusion that in the client/server environmemt, although distributed application is

assumed, the data could become centralized (in the case of file server or database server) or decentralized (in the case of distributed database system) and the data management method through a distributed database system where complete responsibility and powers with respect to control of data used by the user are given not only is it more adaptable to modern flexible corporate environment, but in terms of system operation, it presents a more efficient data management alternative compared to existing data management methods in terms of cutting costs.

1. 서 론

급격하게 변화하는 기업 경영의 환경과 다양화해가는 고객의 요구는 현대의 기업으로 하여금 변화하는 환경에 보다 유연하게 적응할 수 있도록 정보화를 기반으로 하는 조직 구조로 변화해 나아갈 것을 요구하고 있다. 또한 정보기술의 측면에서는 퍼스널 컴퓨터와 워크스테이션의 성능 발전에 따른 소형-고성능화, 통신 기술의 발달에 따른 분산화, 운영체제(operating system)의 개방 체제(open system)화에 따른 개방화 추세가 주류를 이루고 있다.

이러한 조직 측면의 원인과 기술 측면의 원인은 기업 조직내에서 경영관리

에 필요한 어플리케이션(application)을 다양한 종류의 컴퓨터에 분산하여 처리하는 경향을 가져오게 되었는데, 이러한 분산 어플리케이션을 지원하기 위한 새로운 정보기술 기반구조로서 분산 컴퓨팅(distributed computing) 환경이 등장하게 되었다.

특히 클라이언트/서버 아키텍쳐(client/server architecture)는 분산 컴퓨팅 환경에서 실행가능한 대표적인 컴퓨팅 아키텍처로, 이는 단순히 하나의 컴퓨팅 기술을 의미한다기 보다는 최종사용자에게 실질적인 권한과 책임을 부여하는 동시에 정보 시스템 부문을 재구성하려는 정보시스템에 있어서의 새로운 패러다임, 더 나아가 하나의 철학이라고 까지 말할 수 있는 개념이다.

클라이언트/서버 아키텍쳐가 기업 조직에서 어플리케이션의 처리를 여러 개의 다양한 컴퓨터에 분산시키는 도구가 되는 반면, 분산 데이터베이스 시스템(distributed database system)에서는 그러한 어플리케이션의 처리에 사용되어지는 데이터를 분산하여 관리하는 방법론을 채택하고 있다. 즉 소형화, 분권화, 국제화 등의 기업 환경 변화는 세분화된 기업 조직으로 하여금 그들에게 필요한 데이타의 관리에 대한 자율권을 주어 결과적으로 중앙 집중적인 데이타 관리 방식으로부터 점차로 분산 데이타 관리 방식으로 나아가게 하고 있는데 현재 우리나라의 여러 기업 및 조직에서도 분산 데이타베이스 시스템의 구현을 시도하고 있다.

최근에 정보 산업계와 기업의 일선 사용자들간에 분산 컴퓨팅, 클라이언트/서버, 다운사이징(downsizing), 협동 처리(cooperative processing), 분산 데이타베이스와 같은 용어들이 많이 사용되어지고 있음에도 불구하고 아직까지 그 정확한 개념 정의조차 이루어지지 못하고 있으며 따라서 시스템 구

현과 실행에 많은 혼동을 가져오는 것이 사실이다.

한편 기업에 있어서 정보 시스템의 가장 큰 목적중 하나가 기업에서 필요한 데이타를 효율적으로 관리하여 의사결정의 질을 향상시키려는데 있는바 이는 클라이언트/서버 아키텍쳐로 구현된 시스템의 경우라도 예외가 될 수는 없는 것이다. 따라서 효율적 데이타의 관리라는 측면에서 클라이언트/서버 아키텍쳐를 재조명해보고 클라이언트/서버 환경하에서의 바람직한 데이타 관리 방법론을 개발해 나가는 것은 매우 중요하다고 하겠다.

이에 본 연구에서는 클라이언트/서버 아키텍쳐에 대한 정확한 정의와 이론체계를 확립하고 이러한 클라이언트/서버 아키텍쳐의 정의와 이론체계를 기반으로 클라이언트/서버 환경하에서의 데이타 관리 방안을 제시하고자 한다.

2. 분산 컴퓨팅 환경

본 장에서는 여러 개의 다양한 컴퓨팅 시스템에 어플리케이션을 분산하여 처리하는 클라이언트/서버 환경과 이러한 어플리케이션의 처리에 필요한 데이타를 분산하여 관리하는 분산 데이타베이스 시스템 환경을 포함하는 정보기술 기반구조로서 분산 컴퓨팅 환경에 대해서 살펴 보기로 한다.

2.1 분산 처리의 개념

분산 컴퓨팅(distributed computing) 환경을 이해하기 위해서는 그 기본

이 되는 개념으로서 분산 처리(distributed processing)를 먼저 이해해야만 하는데, Martin[1981]은 분산 처리 시스템(distributed processing system)를 다음의 조건을 만족하는, 하나 이상의 처리기(processor)를 갖는 컴퓨터 시스템의 운영 방법이라고 정의하였다.

첫째, 처리기들이 서로 다른 위치에 자리잡고 있고,

둘째, 그들이 원거리통신(telecommunication)에 의해 상호연결되어 있으며,

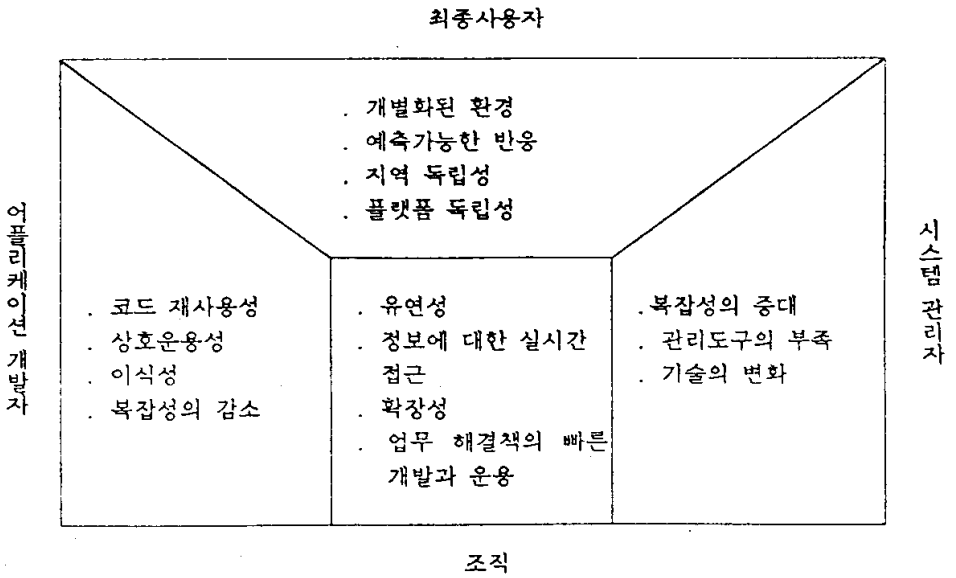
셋째, 시스템이 하나의 조직을 위해 서비스를 제공하고 있고,

넷째, 시스템이 통합된 방법에 의해 계획되고 설계되어진다.

이로부터 분산 데이타 처리 시스템(distributed data-processing system)의 개념이 도출되는데 이는 "지리적으로 분산된 하나 이상의 개별적 처리기를 운용하는 조직내에서 데이타를 조작하는 시스템"을 가리키며 대개의 경우 각각의 처리기들은 원거리통신에 의해 연결된다.

2.2 분산 컴퓨팅 환경의 개념

분산 컴퓨팅(distributed computing) 환경은 분산 어플리케이션 처리를 위한 정보기술 기반구조(information technology infastructure)를 가리키는 말로 〈그림 1〉에서 보듯이 최종사용자에게 있어서 분산 컴퓨팅 환경은 조직의 요구에 맞추어진, 그래픽 사용자 인터페이스(Graphic User Interface : GUI)를 통해 접근이 가능한 정보와 컴퓨터 환경을 의미하는 반면, 시스템 관리자에게 있어서 분산 컴퓨팅 환경은 기술의 변화, 복잡성의 증대, 관리도구의 부족을 의미한다.



〈그림 1〉 분산 컴퓨팅에 대한 다양한 시각

자료원: Khanna(editor)[1994]

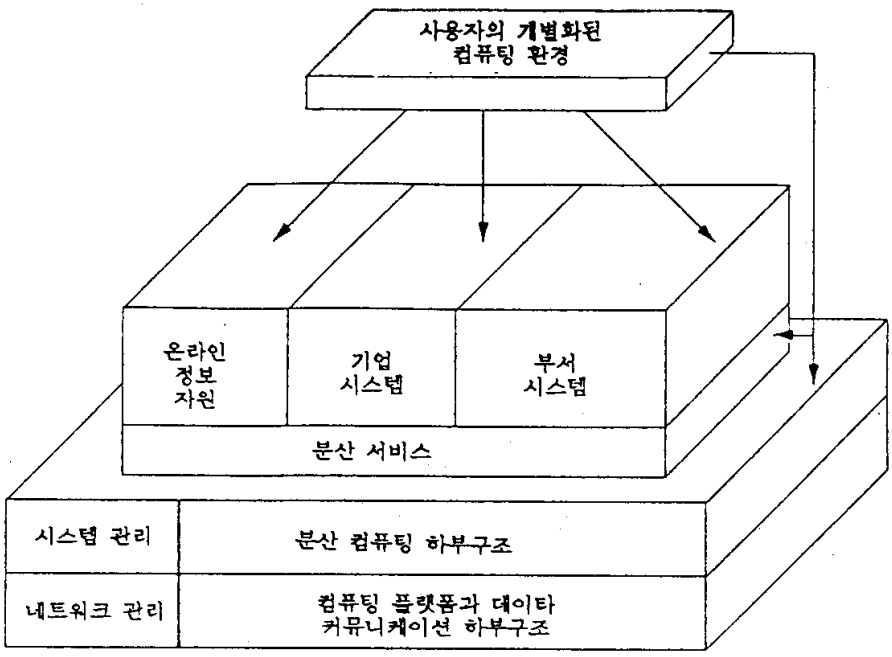
또한 어플리케이션 개발자에게 있어서 분산 컴퓨팅 환경은 복잡성을 감소시키고, 상호운용성(interoperability)과 이식성(portability), 어플리케이션 코드를 재사용할 수 있는 능력(reusability)을 제공해 주는 것이다.
이러한 분산 컴퓨팅 환경의 구축은 조직으로 하여금 유연성(flexibility), 확장성(scalability), 정보에 대한 실시간(real-time) 접근, 그리고 업무 해결책(business solution)의 빠른 개발을 가능케 한다. [Khanna(editor), 1994]

분산 컴퓨팅 환경에서 구축된 분산 컴퓨팅 시스템은 "업무기능을 성취하기

위하여 커뮤니케이션 네트워크(communition network)를 통해 상호 연결된 자율적(autonomous) 컴퓨터들의 집합" 이라고 정의할 수 있는데 기술적인 측면에서 보면 분산 컴퓨팅 시스템을 구성하는 컴퓨터들은 전역변수(global variable)을 통해 주기억 장치(main memory)를 공유하지 않으며 결과적으로 컴퓨터간의 정보(지식)는 단지 네트워크 상에서 메세지(massage)를 통해서만 교환하게 된다.

이러한 분산 컴퓨팅 시스템에 비해서 분산 컴퓨팅 환경은 어플리케이션, 기업시스템, 관리 설비 등을 총 망라한, 보다 광의의 개념인 것이다.

지난 10년간 정보 산업이 네트워킹에 중점을 두어 왔다면 다가오는 10년간은 그 중점이 분산 컴퓨팅으로 옮겨질 것이라고 예상 되는데, 분산 컴퓨팅 환경이 기존의 네트워크 컴퓨팅 환경(networked computing) 환경과 다른 점은 네트워크 컴퓨팅 환경은 사용자로 하여금 주변기기, 데이타, 프로그램을 공유하고, 정보에 접근하고, 전자우편과 전자회의(electronic conference)를 통해 서로간에 의사소통할 수 있도록 하는 반면 시스템 관리와 이질적인 플랫폼에서 수행되어지는 다양한 어플리케이션간의 통합(integration)을 제공하기 위한 하부구조(infrastructure)가 부족하다는 점이다. 즉 네트워크 컴퓨팅 환경은 〈그림 2-2〉에서 분산 컴퓨팅 하부구조와 분산 서비스 블럭(distributed service block)이 빠진 형태를 갖는다. 이에 비해 분산 컴퓨팅 환경은 분산 컴퓨팅 하부구조, 시스템 관리, 분산 어플리케이션의 세가지 요소로 구성되어 있으며, 다양한 시스템간의 통합(integration) 측면이 강조된다.



〈그림 2〉 분산 컴퓨팅 환경

자료원: Khanna(editor)[1994]

〈그림 2-2〉는 사용자의 입장에서 본 분산 컴퓨팅 환경을 나타내는데 사용자의 개인별 컴퓨팅 환경(user's personalized computing environment)로부터 사용자는 부서 시스템(departmental system), 기업 시스템(enterprise system), 온라인 정보 자원(online information resource)에 접근할 수 있으며 이들 업무 시스템은 분산 컴퓨팅 기반구조(distributed computing infrastructure) 위에서 수행되는 분산서비스(distributed

service)의 토대위에 구축되어지는 한편, 이러한 분산 컴퓨팅 하부구조는 컴퓨팅 플랫폼[1]과 데이타 커뮤니케이션 하부구조의 토대 위에서 구축되어진다. 또한 사용자는 때때로 분산 컴퓨팅 서비스에 직접적으로 접근하기도 한다. [Khanna(editor), 1994]

3. 클라이언트/서버 아키텍쳐

본 장에서는 앞장에서 제시한 분산 컴퓨팅 환경을 구현하기 위한 하나의 컴퓨팅 구조로서 클라이언트/서버 아키텍쳐의 일반적인 개념을 살펴보고 협동처리의 측면에서 이를 새롭게 정의하는 한편, 5가지 협동 처리 방식에 대해 살펴보기로 하겠다.

3.1 클라이언트/서버 아키텍쳐의 정의

클라이언트/서버 아키텍쳐(client/server architectue)는 "미래의 물결"이라고 불리우는 1990년대의 컴퓨팅 패러다임으로서 통제와 대고객 서비스를 증진시키는 동시에 비용 또한 최소화 하려고 노력하는 기업에게 있어서 매우 각광받는 전략 중 하나가 되었다.

1) 플랫폼(platform)은 하드웨어, 시스템 소프트웨어(Operating System), 어플리케이션 소프트웨어, 그리고 이들을 지원하는 유틸리티 소프트웨어(utility software)로 구성된다.[Atre, 1992]

클라이언트/서버 아키텍쳐는 2장에서 살펴 본 분산 컴퓨팅 환경이라는 정보 기술 기반 구조 위에서 실행 가능한 가장 대표적인 컴퓨팅 아키텍쳐로서 네트워크를 통하여 클라이언트와 서버간에 어플리케이션을 분담하여 처리하는 컴퓨팅 구조를 지칭하는 말이다.

Dewire[1993]는 클라이언트/서버 아키텍쳐를 "대부분의 어플리케이션이 프로그래밍이 가능한 데스크탑 컴퓨터에서 처리되어지며, 그러한 데스크탑 컴퓨터 또한 주인/종속 구조(master/slave configuration)내에서 또다른 컴퓨터 -데이타베이스 서버와 같은- 로부터 어플리케이션 서비스를 받는 컴퓨팅 구조"라고 정의하였다.

즉 클라이언트/서버 아키텍쳐는 하나의 어플리케이션을 다수의 과업(task)들로 분할하여 클라이언트와 서버가 자신의 처리 능력과 처리 비용을 감안하여 가장 효율적인 방법으로 이 과업들을 분담하여 통신 네트워크를 통하여 협동 처리(cooperative processing)하는 컴퓨팅 구조를 가리키는데 이때 각각의 과업(task)들은 서로 다른 운영 체제(operating system), 서로 다른 네트워크 프로토콜(network protocol)을 가지는 이질적 플랫폼들 -즉 클라이언트와 서버들- 에서 수행되어진다. 또한 어플리케이션을 구성하고 있는 개별적인 과업들이 여러 플랫폼하에서 따로 따로 개발, 유지보수되어 질 수 있으므로 어플리케이션 개발 기간을 단축할 수 있다는 장점을 갖는다.

위와 같은 정의에 의하면 클라이언트/서버 아키텍쳐는 어플리케이션을 수행하는 전방 시스템 -클라이언트- 와 데이타 베이스 접근과 처리를 담당하는 후방 시스템 -서버- 로 이루어지는 분산 처리에 기초한 개념으로, 클라이언트와 서버 그리고 이 둘 간의 커뮤니케이션을 제공하기 위한 수단인 네트워크로 구성되어 있음을 알 수 있다.

3.2 클라이언트/서버 협동처리의 개념

이 절에서는 본 논문이 연구의 기반으로 삼고있는 클라이언트/서버 아키텍처의 정의를 내리기 위해 클라이언트/서버 협동 처리에 대해서 살펴보고자 한다.

3.2.1 복수 계층 환경(Multitiered Environment)

클라이언트/서버 협동 처리는 복수 계층 환경(multitiered environment)을 그이론적 기반으로 삼고 있다. 즉 분산 컴퓨팅 아키텍쳐는 오늘날 컴퓨팅 환경을 지배하는 두 가지 상반된 원인에 의해 영향을 받아왔는데 그 하나는 워크스테이션과 퍼스널 컴퓨터의 가격대 성능비상의 잇점을 이용하고 최종사용자 컴퓨팅을 실현시키기 위해 어플리케이션을 세분화하여 최종사용자에게 개발과 운용의 자율권을 부여하는 것이다. 또한 이와는 상반된 다른 하나의 원인은 기업의 전사적인 데이타에 접근하려는 사용자의 요구인데, 이는 시스템을 통합하려는 사용자의 요구를 가져오게 되어 결과적으로 대형의 강력한 메인프레임하에 어플리케이션을 집중시키게 된다.
이러한 상반된 두가지 요구를 모두 만족시키기 위해서는 복수 계층 환경(multitiered environment)이 필요하다. 특히 대부분의 대규모 조직은 복수 계층 환경에서도 특히 3계층 구조(3-tired architecture)로 나아가고 있는 추세인데 이는 전통적인 계층적 컴퓨팅 모델인 주인-종속구조(master-slave architeture)로 부터 나온 개념으로 여기에 협동 처리의 개

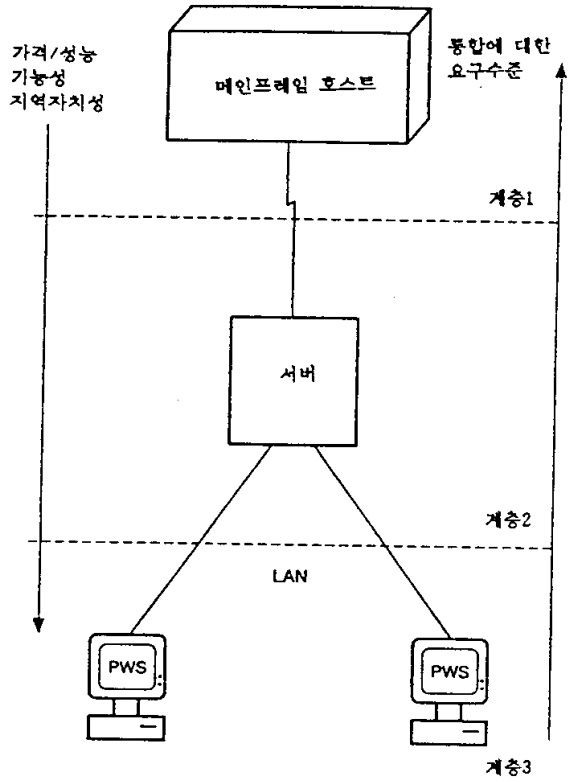
념을 첨가한 것이다. [Berson, 1992]

3계층 구조의 가장 위층은 전사적인 데이타의 원천으로서 강력한 메인프레임이 자리잡고 있고 가장 아래층은 최종사용자 인터페이스를 제공하는 클라이언트로서 워크스테이션과 퍼스널 컴퓨터가 자리잡고 있다. 중간층은 가장 위층의 시스템 -즉 메인프레임- 의 클라이언트인 동시에 가장 아래층 -즉 워크스테이 션 또는 퍼스널 컴퓨터- 의 서버로서의 기능을 수행하는 LAN서버가 자리잡고 있는데 이러한 아키텍처는 가장 위층에 메인프레임을, 그리고 중간층과 가장 아래층에 LAN서버와 클라이언트 시스템을 추가함으로써 수평적 확장이 쉽게 이루어질 수 있다.

실제로 어플리케이션을 이러한 구조로 구현하기 위해서는 우선적으로 어플리케이션이 어떻게 구성되어 있는가 하는 것을 살펴보아야 하는데, 일반적으로 어플리케이션은 최종사용자의 터미날이나 워크스테이션과 상호작용하는 어플리케이션 코드의 일부분으로 스크린 포매팅(screen formatting), 윈도우 관리(window management), 키보드 및 마우스의 관리등과 같은 과업을 수행하는 표현 처리 논리(Presentation Processing Logic), 최종사용자나 데이타베이스 입출력에 직접 관련이 없는 어플리케이션 코드의 일부분으로 화면(screen)이나 데이타베이스로부터 입력 데이타를 받아서 특정 업무의 요구사항, 규칙, 알고리듬에 따라서 그것을 처리하는 기능을 수행하는 업무 처리 논리(Business Processing Logic), 그리고 데이타베이스 입출력에 직접 관련한 어플리케이션 코드의 일부분인 데이타 관리 논리(Data Management Logic)[2]로 구성되어 있다. 따라서 3계층 구조로 어플리케이

2) 데이타 관리 논리는 SQL과 같은 데이타 처리어(Data Manipulation Language:DML)에 의해 어플리케이션 내에서 데이타를 처리하는 데이타베이

션을 구현하기 위해서는 이와같은 3가지 구성요소를 각각 어느 계층에 위치시킬 것인가 하는 것이 중요한 문제가 된다.



〈그림 3〉 3-계층 환경

자료원: Berson[1992]

스 처리 논리(Database Processing Logic)와 데이타베이스 관리 시스템이 실제로 데이타를 처리하는 데이타베이스 처리(Database Processing)로 구성되어 있다.

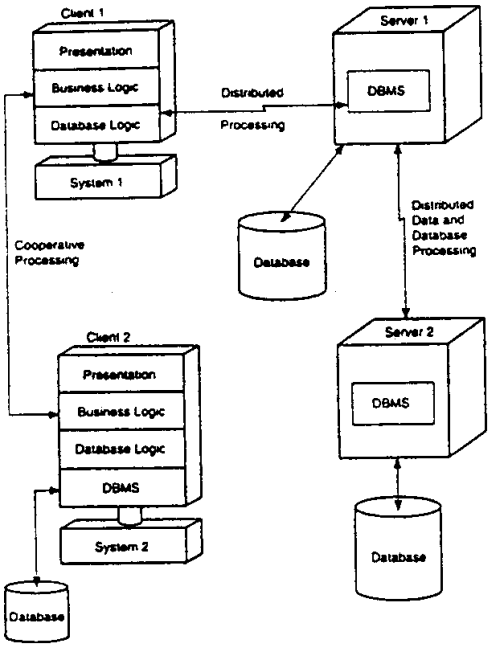
3.2.2 클라이언트/서버 협동 처리의 개념

위에서 살펴본 바와 같이 클라이언트/서버 협동 처리는 어플리케이션의 각 구성요소들을 3계층 구조로 구현하는 과정으로부터 유도되는 개념인데 이러한 클라이언트/서버 협동 처리를 정의하기에 앞서 일반적인 협동 처리의 개념을 살펴보는 것이 필요하다. 협동 처리(cooperative processing)란 일반적으로 "두개 또는 그 이상의 컴퓨터들이 하나의 프로그램 처리를 공유하는 구조(architecture)"라고 정의할 수 있다. 협동 처리 환경하에서는 어플리케이션의 부분들이 서로 다른 컴퓨터에서 수행되어지고 데이터 역시 다양한 컴퓨팅 시스템내에 자리잡을 수 있으므로 협동 처리를 한마디로 하면 프로그램, 파일, 데이타베이스와 같은 컴퓨팅 자원들을 네트워크를 통해 분산시킬 수 있는 능력이라고 할 수 있다.

그러면 클라이언트/서버 협동 처리란 무엇일까? 컴퓨팅 기술의 발전과 더불어 컴퓨팅 환경은 일반 목적의 집중 시스템으로 부터 네트워크를 통해 상호연결된, 여러개의 전문화된 시스템들의 집합적인 시스템 아키텍쳐로 발전해 나가는 추세인데, 이때 전문적인 시스템들의 집합을 하나의 컴퓨팅 실체로 결합하기 위해서는 네트워크내의 모든 시스템 노드를 상호연결하여 각각의 노드들이 주어진 상호관계를 유지하도록 해야 한다.

이러한 분산 컴퓨팅 환경하에서 "분산한다"는 것은 가능한 컴퓨터 자원을 여러 부분으로 분할하여 네트워크를 통해 그들을 전파시키는 것을 의미하는데 여기서 문제가 되는 것은 어떠한 자원을 분산시킬 것이며 그러한 분산으로 인한 결과는 무엇인가 하는 것이다. 단지 데이타만 분산시키는 경우 데이타의

분산으로 인한 여러가지 혜택을 얻을 수 있는 반면 어플리케이션의 집중으로 인한 병목현상(bottleneck)이 발생할 수 있고 성능, 이식성(portability), 어플리케이션 확장성(scalability)의 잇점이 제한된다는 단점 또한 갖게 된다. 그러나 데이타이외에 어플리케이션의 처리 자체가 분산되는 경우에 있어서는 일단 어플리케이션의 구성요소가 분산되면 하나의 업무 어플리케이션을 수행하는데 있어서 그 구성요소들이 서로 협조해야만 한다.[Berson, 1992]



〈그림 4〉 클라이언트/서버 분산 협동 처리

자료원: Berson[1992]

이에 클라이언트/서버 아키텍쳐를 협동 처리의 측면에서 보면 첫째, 클라이언트(주로 표현 논리 그리고 일부 업무 논리)와 서버(일부 업무논리, 그리고 데이타베이스 논리와 DBMS)간에 어플리케이션 처리 요소를 분산시키고 둘째, 클라이언트와 서버간의 긴밀한 상호작용을 통해 분산 협동 처리(distributed cooperative processing)를 수행하는 컴퓨팅 구조(Computing architecture)라고 정의할 수 있다. [Berson, 1992][Atre, 1992]

따라서 클라이언트/서버 컴퓨팅은 분산 협동 처리(distributed cooperative processing)의 하나의 특수한 형태라고 할 수 있으며 클라이언트/서버 아키텍쳐에서는 주어진 어플리케이션 기능을 수행하기 위하여 서로 협동하는 클라이언트와 서버 간에 여러 어플리케이션 구성요소들이 분산되어지며, 클라이언트가 상호작용을 시작해서 서버에게 특정 서비스를 요청하면 서버는 이러한 서비스를 수행하여 그에 대한 반응을 클라이언트에게 되돌려 보내는 방식으로 그들간의 상호작용이 이루어지는 것이다. 특히 클라이언트와 서버가 서로 물리적으로 분리된 원격지 노드에 위치하는 경우에는 클라이언트와 서버간의 상호작용을 위한 의사소통은 적합한 네트워크를 통해서 이루어져야만 한다.

이처럼 분산 협동 처리의 한 형태로서 클라이언트/서버 아키텍쳐를 정의한다면 클라이언트/서버 상호작용은 분산 환경하에서 여러가지 컴퓨팅 요소들의 협동적 상호작용이라고 정의할 수 있으며 이는 클라이언트와 서버 노드의 전문화를 증진시키기 위한 것이다. 즉 클라이언트/서버 시스템의 최적 설계는 클라이언트와 서버간의 협동 처리로 부터 얻는 이익을 최대화할 수 있도록 각

노드들을 전문화함으로써 이루어질 수 있는 것이며, 최적의 플랫폼에서 개발되어진 어플리케이션을 최적의 플랫폼에서 실행하는 것이 그 기본 사상이다. [Atre, 1992]

그런데 협동 처리는 클라이언트/서버 아키텍쳐를 구성하는데 있어서 필요조건이기는 하지만 이것이 클라이언트/서버 아키텍쳐를 특징짓는 유일한 요구조건이라고 하기에는 충분치 않다. 이는 만일 두개의 소프트웨어 구성요소들이 협동처리를 하고 있다 하더라도 이것이 반드시 클라이언트/서버 컴퓨팅 모델을 나타내고 있는 것은 아니라는 것을 의미한다.3)

한편 이러한 협동 처리로 얻을 수 있는 혜택은 다음과 같다. [Atre, 1992] 첫째, 사용자의 입장에서 보았을 때에는 사용자가 화면(screen)에서 보고 있는 것이 바로 시스템 자체이다.

둘째, 윈도우(Window), 오픈룩(Open Look), 뉴웨이브(New Wave), 모티프 (Motif) 또는 프리젠테이션매니저(Presentation Manager)와 같은 사용 자에게 좀더 친숙한 인터페이스를 사용함으로써 마이크로 컴퓨터 또는 워크스테이션 사용자가 어플리케이션을 쉽게 사용하도록 하며 오류 (error)가 발생할 가능성을 줄일 수 있다.

셋째, 협동 처리는 호스트 또는 서버 컴퓨터에서 수행되어지는 처리의 양을 감소시킬 수 있다.

넷째, 업무가 클라이언트와 서버로 분산되어질 수 있다.

다섯째, 서버에 의해 수행되어지는 단위 업무당 비용은 대개의 경우 클라이언 트에 비해 높다. 따라서 업무를 클라이언트로 분산시켜 협동 처리 하는 것

3) 클라이언트/서버 아키텍쳐의 요건이 성립하려면 소프트웨어간의 협동 처리라는 특성이외에도 하드웨어의 기능 전문화라는 특성이 뒷받침되어야 한다.

은 비용의 절감을 가져올 수 있다.

위와같은 협동 처리로 인한 장점은 소프트웨어 구성요소의 협동처리 뿐만 아니라 하드웨어 구성요소의 전문화를 통해서도 실현될 수 있다. 따라서 클라이언트/서버 아키텍처는 소프트웨어 구성요소간의 협동처리 뿐만 아니라 하드웨어 구성 요소간의 협동적 상호작용도 포함하는 보다 확장된 개념으로 정의할 수 있게 된다.

3.2.3 단일 시스템 이미지(Single System Image)

위와같은 클라이언트/서버 협동 처리를 가능하게 하기 위해서는 시스템이 단일 시스템 이미지(Single System Image : SSI)를 가지고 있어야 한다. 즉 분산 협동 처리를 하는 클라이언트/서버 환경하에서는 클라이언트와 서버간에 어플리케이션 요소들을 분산시켜서 서로 다른 여러개의 시스템들이 사용자가 요청한 하나의 어플리케이션 실행에 참여하게 되지만, 최종 사용자에게는 이러한 분산된 전체의 "복수 클라이언트/복수 서버" 환경이 단일 사용자의 강력한 시스템처럼 보여져야 한다는 것이다.

적절하게 구현된 클라이언트/서버 분산 환경은 최종사용자에게 단일 시스템 이미지를 제공해야 하는데 이러한 단일 시스템 이미지 환경내에서 업무 시스템의 사용자는 데이타가 어디에 저장되었는지, 클라이언트와 서버 처리기들이 어떻게 일하고 있는지, 시스템의 네트워킹이 어떻게 이루어지고 있는지에 대해서 전혀 알 필요가 없으며 이러한 단일 시스템 이미지를 제공받기 위해서 클라이언트/서버 구성요소들은 어떠한 형태로든 협동 처리에 참여해야

만 하는 것이다.

이와같은 현상을 투명성(transparency)이라고 하는데 이는 다음과 같은 요인들에 의해서 얻어질 수 있다. [Smith, 1992]

첫째, 사용자가 접근하는 모든 어플리케이션이 공통의 "시각과 느낌(look and feel)"을 주도록 오류(error)는 모든 어플리케이션에서 같은 방법 으로 표현되고 해결되며, 접근(access)은 모든 어플리케이션에서 표준 보안 절차를 통해 제공되어져야 하며, 모든 사용자는 접근의 필요와 권리를 가지는 모든 서비스에 대해 접근할 수 있어야 한다.

둘째, 권한이 있는(authorized) 사용자에게는 보안(security) 절차가 가시화되 지 않으나 권한이 없는(unauthorized) 사용자에게는 시스템에 대한 통 제가 철저하게 이루어져야 한다.

셋째, 기능간 또는 어플리케이션간의 상호작용은 모든 어플리케이션에서 같은 방법으로 행하여져야 하며 새로운 어플리케이션은 쉽게 추가되어질 수 있어야 한다.

3.2.4 협동 처리 방식

위에서 협동 처리는 분산 처리의 특수한 경우로 전형적인 어플리케이션의 구성요소 -표현 논리, 업무 논리, 데이타 관리 논리- 들이 둘 혹은 그 이상의 컴퓨팅 시스템에 분산되어 그들간에 고도의 상호작용을 하는 것이라고 정의하였는데 특정 어플리케이션 구조를 위해 선택되어지는 분산점(distribution point)[4]에 따라 다음과 같은 협동 처리 방식이 정의될 수 있으며 이러한 방

식들을 서로 복합적으로 사용하는 것이 가능하다.[5] [Berson, 1992]

3.2.4.1 분산 표현 방식

분산 표현 방식은 표현 논리(presentation logic)기능만을 클라이언트와 서버에 분산시키고 나머지 업무 논리(business logic)기능과 데이타 관리 논리(data management logic) 기능은 서버에 두는 방식이다.[신상훈, 1992]

전형적인 분산 표현 모델은 전위(front-end)요소와 후위(baek-end)요소로 구성되는데, 전위요소는 사용자 인터페이스의 물리적인 부분 -즉 화면 디스플레이, 그래픽 사용자 인터페이스(GUI), 윈도우(window)관리, 색깔, 폰트(font), 마우스(mouse), 키보드(keyboard)- 등을 취급하며 터미날, 퍼스널 컴퓨터, 워크스테이션등의 사용자 인터페이스 장치-즉 클라이언트-에 위치하는 반면, 후위요소는 전위요소와는 다른 노드 -즉 서버- 에 위치하여 여러 시스템들에 의해 공유되어지는 공통의 표현 기능을 수행한다.[Berson, 1992]

분산 표현 방식은 기존의 데이타 체계와 기술 구조에 변화를 주지 않고 단순히 하나의 표현층 -예를 들어 그래픽 사용자 인터페이스(GUI)와 같은- 만을 추가시키는 방식으로 협동 처리 방법중 가장 기본적인 방법에 속하는데

4) 분산점이란 어플리케이션 요소가 나머지 요소들로부터 분산되어 나가는 위치를 말한다.

5) 이는 세계적인 전문 조사기관인 가트너 그룹에서 제시한 분산 모형의 분류 체계에 의한 것으로서 표현(presentation), 업무 논리(business logic), 데이타 관리(data management)로 이루어진 정보시스템 기반 구조의 관점에서 클라이언트/서버 협동 처리 구조를 분류한 것이다.

[신상훈, 1992] 이러한 협동 처리 방식은 비교적 간단하게 구현할 수 있다는 장점을 가지는 반면, 분산 협동 처리가 가지는 장점을 충분히 이용하지 못한다는 단점을 가진다.

3.2.4.2 원거리 표현 방식

표현 기능이 어플리케이션의 나머지 부분으로 부터 분리되는 방식 중 하나인 원거리 표현(remote presentation) 방식은 어플리케이션 코드의 표현 기능 전체를 클라이언트에 위치시키고 어플리케이션의 나머지 부분 -업무 논리와 데이타 관리 논리- 을 서버에 위치시키는 방식이다.[신상훈, 1992] 따라서 원거리 표현 처리는 표현 기능과 여타 다른 기능간의 협동 처리 방식이라고 할 수 있으며 이러한 방식은 최종 사용자 상호작용이 사전에 결정되어지는 비대화 방식의(nonconversational) 어플리케이션 유형에 가장 적합하다.[Berson, 1992]

이러한 협동 처리 방식은 표현 논리가 사용자의 컴퓨터로 완전히 분리(off-load)됨으로써 처리비용이 높은 중앙의 대형 컴퓨터를 표현 논리의 처리에 사용하지 않아도 되기 때문에 경제적이며, 더욱 빠른 사용자 인터페이스 응답시간을 가지게 된다는 장점을 가지는 반면, 표현 논리가 사용자의 컴퓨터로 분리되어 전산실에 의해 통제되지 않기 때문에 사용자 인터페이스의 체계적인 관리가 어렵다는 단점을 가진다.

3.2.4.3 분산 업무 논리 방식[6)]

단순한 분산 환경에서는 어플리케이션 기능이 단일 시스템 -즉 네트워크내의 단일 노드- 에 위치하는데, 이 경우 표현 기능과 데이타 관리 기능은 어플리케이션과 분리되지만 어플리케이션 업무 논리는 더 작은 구성 요소로 분리되지 않는다. 이러한 설계는 비교적 간단한 반면 어플리케이션이 단일 노드에 위치함으로써 대량의 트랜잭션을 처리해야 하는 환경하에서는 처리상의 병목현상(throughput bottleneck)을 초래할 수 있고, 어플리케이션이 위치하는 단일의 노드가 단일의 고장점(single point of failure)으로 작용할 수 있으므로 전체 시스템의 신뢰도가 떨어지며, 분산된 컴퓨팅 자원을 충분히 활용하지 못하고, 트랜잭션의 증가로 인해 어플리케이션이 위치하는 노드(application-hosting node)를 계속적으로 증설해야만 하는 문제점을 가진다.

이러한 문제점을 완화시키기 위한 자연스러운 방법은 어플리케이션 기능을 여러 노드에 걸쳐 분산시키는 것인데, 분산 업무 논리(distributed business logic) 방식은 업무 논리 기능을 서로 다른 시스템 노드 -즉 클라이언트와 서버- 에 나누어 분산시키는 방식이다. 즉 클라이언트가 표현 기능의 전부와 업무 논리 기능의 일부분을 수행하고 서버가 그 나머지 업무논리 기능과 데이타 관리 논리 기능을 수행하는 것인데, 이 경우 클라이언트와 서버 모두가 어플리케이션 논리를 처리할 수 있도록 각각 컴파일된 프로그램 모듈들을 소유하고 있으며 서로는 메세지(message)를 사용하여 대화한다.

전형적인 업무 논리 기능의 분리는 전위/후위(front-ent/back-end) 방식으로 이루어지는데 이는 전위요소가 상호작용을 시작하고 이에 대해 후위요

6) 이러한 방식을 분산 기능(distributed function) 방식이라고도 한다.[신상훈, 1993] [박주석, 1994]

소가 반응하는 형태로 이루어지며, 전위요소는 논리적으로 클라이언트 노드에, 후위요소는 서버 노드에 위치하게 된다.

이러한 분산 업무 논리 방식은 특히 복잡하고 고도의 상호작용이 이루어지며, 데이타베이스 입출력이 빈번한 클라이언트/서버 어플리케이션에 잘 적용되는데 이때 표현 기능에 관련한 업무 논리 부분은 최종사용자 노드 -즉 클라이언트 워크스테이션- 에, 데이타베이스에 관련한 업무 처리는 데이타베이스 관리 시스템(DBMS)을 포함하고 있는 노드-즉 데이타베이스 서버-에 각각 위치하게 된다.

잘 구현된 분산 업무 논리 방식은 결과적으로 어플리케이션 프레그멘트(fragment)가 협동적 트랜잭션을 처리하는 중에 교환해야 하는 메세지의 양을 줄어들게 하고, 응답시간(response time)을 증진시키고, 가용 컴퓨팅 자원을 보다 잘 활용할 수 있도록 해주는 반면, 설계와 개발이 가장 어려운 협동 처리 방식이다.[7][Berson, 1992]

3.2.4.4 원거리 데이타 관리 방식

원거리 데이타 관리 방식은 클라이언트가 표현 기능과 업무 논리 기능의 전부를 처리하고 서버는 데이타 관리 기능을 담당하는 방식이다.[신상훈, 1992]

데이타 관리 논리는 데이타베이스 처리 논리(data processing logic)와 데이타베이스 처리(database processing)로 구성되어 있는데 데이타베이스

7) 분산 업무 논리 방식은 트랜잭션(transaction)과 분산 트랜잭션 처리(Distributed Transaction Processing)와 밀접한 관련이 있으나 분산 트랜잭션 처리의 개념은 연구의 목적을 벗어나므로 본 논문에서 제외시켰다.

처리 기능은 반드시 데이타베이스 자체와 같은 시스템 내에 의치해야 하는 반면, 데이타 처리 논리-데이타 처리어(Data Manipulation Language: DML)-의 기능은 데이타베이스 관리 시스템(DBMS)와 가깝게 두거나 또는 어플리케이션 업무 논리와 가깝게 둘 수 있다. 이때 특정 어플리케이션을 위한 모든 데이타, 데이타 처리 논리, 데이타베이스 관리 시스템(DBMS)을 어플리케이션 노드 -즉 클라이언트- 와 따로 분리시켜서 별도의 시스템 -즉 서버- 에 위치시키는 협동 처리 방식을 원거리 데이타 관리 방식이라고 하는데 이 경우 어플리케이션 트랜잭션은 한번에 단일 위치의 데이타만을 취급할 수 있다.

아키텍처의 측면에서 원거리 데이타 관리 방식은 전위/후위(front-end / back-end) 처리 모델로서 구현될 수 있는데, 이때 전위 시스템은 어플리케이션의 업무 논리 부분을, 후위 시스템은 데이타베이스와 데이타베이스 관리 시스템(DBMS)을 포함하게 되며, 이는 사용빈도가 낮고 단순하고 적은 양의 결과를 산출해 내는 소규모의 특별한 조회(ad hoc query) 처리 -즉 의사결정 지원 시스템(Decision Support System: DSS) 어플리케이션- 에 가장 적합한 협동 처리 방식이다.[Berson, 1992]

이러한 협동 처리 방식은 사용자에게 프로그램과 통제에 대한 최대의 유연성을 주면서도 조직의 체계적인 정보 환경을 해치지 않기 위하여 전산실에서 조직의 모든 데이타를 통합의 관점에서 엄격하게 관리할 수 있다는 장점을 가지는 반면 많은 기업에서 중요한 정보 처리 방식인 트랜잭션 관리의 개념이 매우 약하다는 단점을 갖는다.[박주석, 1994]

3.2.4.5 분산 데이타 관리 방식

분산 데이타 관리 방식은 분산 표현 방식과는 반대로 표현 논리와 업무 논리의 기능을 클라이언트가 수행하며, 데이타 관리를 클라이언트와 서버가 분담하여 수행하는 방식이다. 이러한 분산 데이타 관리 방식은 여러 개의 노드에 분산되어 있는 데이타(데이타베이스)를 취급하는데 이는 데이타를 그 발생 원천에 보다 가깝게 위치시키고, 중요한 데이타의 경우 여러개의 복사본을 서로 다른 위치에 둠으로써 데이타의 가용성을 높이고 단일의 고장점을 제거함으로써 신뢰성을 증진시킬 수 있다.

이렇게 데이타가 여러 노드에 분산되어지는 경우에는 데이타 관리 논리의 전부 혹은 일부가 반드시 데이타와 함께 분산되어져야 하며 적어도 데이타베이스 처리 논리(DBMS)의 일부분은 데이타베이스 자체와 같은 시스템에 우치시켜야만 한다.

분산 데이타 관리 환경은 분산이 두가지 방향을 갖는다는(two-way distribution) 특징을 갖는데 이는 첫째, 데이타와 데이타베이스 관리 시스템(DBMS)이 어플리케이션 논리가 위치한 노드를 포함한 여러 개의 노드에 걸쳐 분산되어지고 둘째, 데이타 관리 기능이 전방의 클라이언트(데이타 처리 논리 또는 데이타 처리어)과 후방의 서버(데이타베이스 기능 또는 DBMS)간에 분산되어 지는 것을 의미한다.

이러한 구조는 데이타 접근 요청 -즉 데이타 처리어 요청((DML request)- 이 같은 노드상에서 어플리케이션 업무 논리로 부터 데이타 처리 논리로 보내어지기 때문에 네트워크 소통량을 줄일 수 있으며,[Berson, 1992] 조직이 보유한 데이타가 사용자가 쉽게 접근할 수 있도록 여러 지역으

로 분산되며 다른 지역에 있는 데이타가 마치 자기 지역에 있는 데이타처럼 검색되거나 수정될 수 있다는 장점을 가지는 반면, 분산된 조직의 데이타를 하나의 논리적 체계로 통합하기 위하여 데이타베이스 관리 시스템(DBMS)의 복잡한 여러 기능이 필요하므로 기술적인 어려움이 따른다는 단점을 갖는다.

4. 클라이언트/서버 환경에서의 데이타 관리

본 장에서는 앞 장에서 살펴본 클라이언트/서버 협동 처리 구조로부터 클라이언트/서버 환경하에서 사용가능한 데이타의 관리 방안을 제시하고 더 나아가 보다 바람직한 방안을 모색하고자 한다.

4.1 분산 데이타베이스 시스템

먼저 최근에 새롭게 대두되고 있는 데이타 관리 방안의 하나인 분산 데이타베이스 시스템에 대해서 살펴 보기로 한다.

4.1.1 분산 데이타베이스 시스템의 정의

C.J. Date[1985]는 집중 데이타베이스들이 결합된 형태로서 분산 데이타베이스를 정의하였는데 그는 분산 데이타베이스란 "커뮤니케이션 네트워크

(communication network)로 상호 연결되어 있는 사이트(site)[8] 또는 노드(node)[9]들의 집합으로 구성된 시스템으로 각 사이트들이 그 자신의 고유한 데이타베이스 시스템 -즉 그 자신의 고유한 데이타베이스, 터미날, 중앙처리기, 지역 DBMS- 을 갖는 시스템"이라고 하였다. 따라서 분산 데이타베이스 시스템은 어떠한 개별적 객체라기보다는 독립적이지만 상호 협력하는 중앙 집중적 시스템(centralized system)들이 협조관계(partnership)을 이루는 것이라고 보아, 분산 데이타베이스 시스템은 개별적 집중 데이타 베이스 시스템들이 논리적으로 결합된 형태라고 정의하였다.[10]

Bell[1992]은 분산 데이타베이스란 데이타베이스의 요소를 적용한 통합(integration)과 컴퓨터 네트워크의 요소를 적용한 분산(distribution)이라는 양극단의 두 개념을 합쳐 놓은 것이라고 하였는데 여기서 분산의 측면은 네트워크 내에서 여러 사이트(site)에 걸쳐 데이타를 물리적으로 분산시킨다는 것을 의미하는 반면, 통합의 측면은 이와같이 분산된 데이타가 이용자에게는 단일의 동질적인 데이타베이스로 보이도록 이를 논리적으로 통합하는 것

8) 사이트(site)는 분산 시스템 네트워크에 참여하는 하나의 관계 데이타베이스 관리 시스템(relational DBMS)이라고 정의한다.[Atre, 1992]

9) 노드(node)는 단일의 운영체제(OS)하에서 주기억장치를 공유하고 있는 하나 이상의 중앙연산장치(CPU)라고 정의할 수 있는데 이는 데이타를 주고받는 네트워크내의 하나의 위치(point)가 된다.[Atre, 1992]

10) 분산 데이타베이스 관리 시스템(Distributed DBMS: DDBMS)을 논의할 때 단일 지역 시스템(site)의 측면을 다룰 때와 전체적인 측면에서의 시스템을 다룰 때를 구분하기 위하여 "지역(local)"이라는 용어와 "전역(global)"이라는 용어를 구분하여 사용하는데 지역 데이타베이스가 네트워크내에서 하나의 지역 시스템(site)에 저장한 데이타베이스를 의미하는 반면 전역 데이타베이스는 모든 지역 데이타베이스들의 논리적 통합을 의미한다. 즉 전역 데이타베이스는 물리적으로는 존재하지 않는 개념적 실체이다.

을 의미한다. 이에 반하여 집중 DBMS는 논리적, 물리적 통합을 요구한다. 위의 개념에 근거하여 분산 데이타베이스를 정의하면 "컴퓨터 네트워크의 여러 노드에 걸쳐서 물리적으로 분산되어 있는, 공유되어지는 데이타를 논리적으로 통합한 그들 데이타의 집합체"라고 정의할 수 있으며 이러한 분산 데이타베이스 시스템을 관리하는 데이타베이스 관리 시스템(DBMS)을 분산 데이타베이스 관리 시스템(DDBMS)라고 한다.

4.1.2 분산 데이타 베이스 시스템의 목적[11)]

분산 데이타 베이스 시스템의 주된 목적은 지리적으로 멀리 떨어져 있는 원거리자원과 데이타를 사용함으로써 일반 사용자의 생산성을 향상시키고 최대의 시스템 가용성을 얻도록 하며, 각 노드는 각기 어느 정도의 자치성을 획득함으로써 지역적인 정보 처리의 효율을 증진시키는 것이다. 또 시스템의 확장이나 변경을 용이하게 하며 일부 노드가 고장나더라도 전체 시스템은 계속 가동될 수 있도록 가용성과 신뢰도를 증진시키는데 있다.

이러한 목적을 달성하기 위해서 분산 데이타베이스 시스템은 사용자에게는 마치 중앙 집중식 시스템인 것처럼 보여야만 한다. 즉 사용자는 논리적 데이타의 견지에서 생각할 수 있어야만 하며 물리적으로 그러한 객체가 어디에 그리고 몇번 이나 중복되어 저장되고 있는지에 대해서는 알 필요가 없어야 한다. 일반적으로 말해서 분산 데이타베이스 시스템의 문제들은 개념적

11) C.J.Date는 1987년에 성공적인 분산 데이타베이스 시스템이 갖추어야 할 12가지 요건을 발표하였는데 본 논문에서 다루고 있는 분산 데이타베이스 시스템의 4 가지 목적은 이러한 12가지 요건중 가장 중요한 것들이다.

(conceptual) 또는 외부적(external) 수준의 문제가 아닌 내부적 수준의 문제들인 것이다. 따라서 성공적인 분산 데이타베이스 시스템은 사용자에게 위치(location) 투명성, 중복(replication) 투명성, 고장(failure) 투명성, 병렬처리(concurrency) 투명성 등의 네가지 투명성(transparency)을 제공해야 한다.

4.1.3 분산 데이타베이스 시스템의 장단점

분산 데이타베이스 시스템은 집중 데이타베이스 시스템에 비하여 다음과 같은 잇점이 있다.[Date, 1985]

① 지역자치성(local autonomy)

데이타베이스 시스템을 사용하는 기업은 종종 논리적으로 -사업부, 부서, 프로젝트 등으로-분산되어져 있으며 물리적으로도 -공장, 연구소등- 분산되어 있는 경우가 많다. 시스템의 분산은 기업내의 개별적 그룹으로 하여금 그들의 책임하에 그들 자신의 데이타에 대한 지역적 통제권을 행사할 수 있도록 해주며, 그들이 다른 원격지 데이타 처리 센터에 의존하는 정도를 줄이는 한편 반대로 순수한 지역적 이슈에 대해 다른 데이타 처리 센터가 깊숙히 개입하지 못하게 한다.

② 점진적인 시스템 확장

데이타 처리의 크기가 점차로 증가함에 따라 시스템 확장에 대한 요구가 생

기게 되는데 현재의 분산 시스템에 새로운 지역 시스템(site)을 하나 더 추가하는 것이 현재의 집중 시스템을 더 큰 것으로 바꾸는 것보다 훨씬 더 쉽고 비용면에서 경제적이다. 더구나 시스템 확장으로 인한 사용자의 시스템 이용 중단을 줄일 수 있다.

③ 신뢰성과 가용성

분산 시스템은 그것이 "전부가 아니면 아무것도 아니라는 전제(all-or-nothing proposition)"하에 있지 않으므로 집중 시스템보다 더 큰 신뢰성을 제공한다. 왜냐하면 분산 시스템은 개별적인 지역 시스템(site) 또는 개별적인 커뮤니케이션 연결(link)이 고장나더라도 제한된 수준으로나마 기능을 계속 수행할 수 있기 때문이다. 또한 만일 데이타가 중복되어 여러 지역에 위치한다면 가용성 또한 증대되는데, 이는 중복된 데이타의 복사본들이 고장시의 백업 역할을 할 수 있기 때문이다.

④ 효율성과 유연성

분산 시스템에서의 데이타는 정상적인 사용 위치에 보다 가깝게 저장될 수 있으므로 반응시간(response time)과 커뮤니케이션 비용을 둘 다 줄일 수 있다. 그리고 사용 패턴이 바뀌는 경우 데이타는 동적으로 이동 또는 중복되거나 또는 현재의 복사본이 제거될 수 있다.

반면에 분산 데이타베이스 시스템에서는 처리기와 메모리, 데이타베이스들이 분산되어 있기 때문에 질의 처리, 트랜잭션 관리, 데이타 갱신, 장애 발생시 회복 절차등이 복잡해 진다는 문제점이 있다.

4.2 클라이언트/서버 환경에서의 데이타의 관리 방식

클라이언트/서버 협동 처리의 5가지 방식으로부터 클라이언트/서버 환경하에서의 2가지 주요한 데이타 관리 방식이 나오게 되는데 이것이 바로 원거리 데이타 관리 방식과 분산 데이타 관리 방식이다.

4.2.1 원거리 데이타 관리 방식

원거리 데이타 관리 방식은 여러 클라이언트 시스템이 단일의 서버에 위치한 데이타에 접근하는 종래의 화일 서버 또는 데이타베이스 서버 방식으로 구현될 수 있다. [Berson, 1992]

4.2.1.1 화일 서버

화일 서버는 조직내에서 사용되는 화일을 관리하면서 이 화일들을 다수의 클라이언트 시스템들이 공유할 수 있도록 해주는 역할을 하는데 이 때 클라이언트가 요청한 데이타의 전송이 전체 화일 단위로 이루어지기 때문에 사용자의 수나 화일의 양이 조금만 증가해도 네트워크상의 데이타 소통량이 기하급수적으로 증가하는 문제점을 가진다. 또한 다수의 사용자가 하나의 화일에 동시적으로 접근하는 경우 발생할 수 있는 여러가지 문제점들을 해결할 수 있는 메카니즘을 가지고 있지 못하며 따라서 데이타의 무결성(integrity)과 보안(security)을 유지하는데 많은 어려움이 따른다. [Berson, 1992][박주석,

1994]

4.2.1.2 데이타베이스 서버

위와같은 화일 서버의 문제점을 근본적으로 해결하기 위하여 나오게 된 개념이 데이타베이스 서버인데 이는 기존의 데이타베이스 관리 시스템(DBMS)을 서버로 사용하는 것이다.[박주석, 1994] 데이타베이스 서버는 클라이언트가 요청한 데이타의 전송이 전체 화일 단위가 아닌 원하는 레코드별로 이루어 질 수 있도록 하며, 데이타베이스 관리 시스템(DBMS)의 향상된 기능으로 인하여 화일 서버에서 발생할 수 있는 여러가지 문제점들을 해결할 수 있는데, [Berson, 1992] 화일 서버 환경의 경우 사용자 개개인이 각자 자신의 화일을 생성하고 관리하므로 데이타의 중복성과 불일치성이 발생할 가능성이 큰데 비하여 데이타베이스 서버 환경하에서 프로그램이 개발될 경우에는 데이타베이스 관리 시스템(DBMS)의 향상된 기능으로 인하여 이러한 문제점을 해결할 수 있다.

이러한 원거리 데이타 관리방식은 비교적 구현하기 쉽다는 장점을 가지는 반면 다음과 같은 단점을 가진다. [Berson, 1992]

첫째, 원거리 데이타 관리 구조는 단일시스템 -즉 데이타베이스 서버- 에데 이타베이스를 집중시킴으로써 성능 병목현상(performance bottleneck)을 초래할 수 있다.

둘째, 원거리 데이타 관리 방식은 데이타의 요청과 그에 대한 반응이 서버와 어플리케이션 간에 네트워크를 통하여 전달되어야 하므로 심각한 커뮤니케이션 부담을 초래한다.

셋째, 데이타베이스 서버는 전방 시스템으로부터 보내어지는 데이타 처리어 요청의 처리 기능이 제한되어 있다.

넷째, 데이타의 원천과 데이타베이스 관리 시스템(DBMS)의 위치가 일원화되 어 있으므로 단일의 고장점(sigle point of failure)을 제공한다.

다섯째, 데이티베이스 서버의 일원화는 분산 컴퓨터 환경내에서 컴퓨팅 가용 자원을 충분히 활용하지 못하게 한다.

4.2.2 분산 데이타 관리 방식

화일 서버나 데이타베이스 서버를 이용한 원거리 데이타 관리 방식이 갖는 문제점들을 해결하기 위해서 등장하게 된 것이 분산 데이타 관리 방식이며 이는 분산 데이타베이스 시스템을 구축함으로써 실현 가능하다.

이러한 분산 데이타베이스 시스템을 이용한 분산 데이타 관리 방식은 특정 사용자의 업무에 국한되어 사용되어지는 데이타의 경우에는 그 데이타를 주로 사용하는 사용자 -즉 클라이언트- 가 그 데이타를 그들의 지역(local) 데이타로서 보유, 관리하도록 하고, 최종사용자가 그들에게 투명한(transparent) 방식으로 다른 지역 시스템(site) -네트워크 노드- 에 위치한 데이타에 접근할 수 있는 능력을 제공하는 반면, 여러 클라이언트에 의해 공유되어지는 공통의 데이타는 화일 서버 또는 데이타베이스 서버에 집중시켜 관리하도록 하는 것이다.

분산 데이타베이스 시스템을 이용한 데이타의 관리 방식은 조직적인 측면에서는 최종사용자로 하여금 그들 자신이 사용하는 데이타에 대한 직접 통제

권을 부여함으로써 조직내에서 업무에 대한 권한과 책임을 하부에 위양하여 분권화된 조직구조로 변화시키는데 있어서 매우 효과적인 도구로 사용될 수 있을뿐 아니라 시스템의 측면에 있어서도 시스템의 확장이 용이하고 보다 높은 시스템의 신뢰성을 보장받을 수 있으며 데이타가 사용자에 보다 가깝게 위치함으로써 반응시간과 커뮤니케이션 비용을 줄일 수 있다는 점에서 매우 효과적인 방법이라고 하겠다.

5. 결 론

지금까지 본 연구에서는 최근의 정보시스템 구축에 있어서 소형화, 분산화, 개방화 추세를 분산 컴퓨팅 환경이라는 새로운 정보기술 기반구조라고 규정짓고 이러한 분산 컴퓨팅 환경하에서 실행 가능한 컴퓨팅 아키텍처 중 대표적인 것으로 클라이언트/서버 아키텍쳐를 제시하였는데, 클라이언트/서버 아키텍쳐를 어플리케이션 서비스를 요청하는 클라이언트 시스템과 요청받은 서비스를 제공해 주는 서버 시스템간의 분산 협동 처리 구조라고 봄으로써 진정한 의미에서의 클라이언트/서버 아키텍쳐란 클라이언트 시스템과 서버 시스템간에 분산된 어플리케이션이 가장 효율적인 방법으로 협동 처리하는 컴퓨팅 구조라고 정의하였다.

또한 어플리케이션이 클라이언트 시스템으로부터 분산되어 나간 위치(분산점)에 따라 분산 표현, 원거리 표현, 분산 업무 논리, 원거리 데이타 관리, 분산 데이타 관리 방식으로 분류한 협동 처리 구조를 살펴 보았는데, 이러한 5

가지 협동 처리 구조 중에서 데이타의 분산에 관련한 원거리 데이타 관리 방식과 분산 데이타 관리 방식은 클라이언트/서버 환경하에서 사용 가능한 2가지 데이타 관리 방식이 되며 원거리 데이타 관리 방식은 화일 서버 또는 데이타베이스 서버에 의해서, 분산 데이타 관리 방식은 분산 데이타베이스 시스템에 의해서 각각 구현될 수 있다고 하였다. 결론적으로 본 연구에서는 클라이언트/서버 환경에서는 어플리케이션의 분산을 가정하지만 데이타는 집중(화일 서버 또는 데이타베이스 서버의 경우) 또는 분산(분산 데이타베이스 시스템의 경우)되어질 수 있으며 클라이언트/서버 환경하에서의 데이타의 관리 방식은 기업의 업무 성격과 조직 구조에 따라 달라져야 하지만 사용자에게 그가 사용하는 데이타의 통제에 대한 완전한 책임과 권한을 부여하는 분산 데이타베이스 시스템에 의한 데이타 관리 방식은 유연성을 가진 분권화된 조직 구조를 추구하는 현대 기업의 업무 환경에 보다 적합할 뿐만 아니라 시스템 운영의 측면에서 비용을 절감할 수 있다는 점에서 기존의 데이타 관리 방식에 비해서 보다 효과적인 데이타 관리 방안이 될 수 있다고 주장하였다.

參 考 文 獻

國內文獻

1. 노중호, 다운사이즈화 정보시스템 설계방법, 시에치노시스템컨설팅, 1992.
2. 안중호, 경영정보시스템, 1991.
3. 이석호, 데이타베이스론, 정익사, 1987.
4. 강성구 안중호, "정보시스템의 다운사이징에 따른 정보시스템 부문의 역할변화", 한국경영정보학회 93 춘계 학술대회 논문집, 1993.
5. 박주석, "클라이언트/데이타서버 체계와 분산 데이타베이스 체계의 특성 연구", KMIS '92 추계학술대회 논문집, 1992년 11월.
6. 박주석, "분산 데이타베이스 구현, 이론과 실제", 컴퓨터타임즈, 1994년 1월-4월.
7. 신상훈, "클라이언트/서버 구현방법", 경영과 컴퓨터, 1992년 10월 - 1993년 2월.
8. 이기현, "클라이언트/서버 환경 구축과 활용", 정보과학회지 제12권 제1호, 1994년 2월.

外國文獻

1. Andleigh, P.K. & Getzinger, M.R., Distributed Object-Oriented Data-Systems Design, Prentice-Hall International, Inc., 1992.
2. Atre, S., Distributed Databases, Cooperative Processing, & Networking, McGraw Hill, Inc., 1992.
3. Bell, D. & Grimson, J., Distributed Database Systems, Addison -

Wesley, 1992.

4. Berson, A., Client/Server Architecture, Mcgraw-Hill, Inc., 1992.
5. Date, C.J., An Introduction to Database Systems Volume 2, Addison-Wesley, 1985.
6. Dewire, D.T., Client/Server Computing, Mcgraw-Hill, Inc., 1993.
7. Khanna, R. (editor), Distributed Computing : Implementation and Management Strategies, P T R Prentice Hall, 1994.
8. Martin, J., Design and Strategy for Distributed Data processing, Prentice Hall, Inc., 1981.
9. Smith, P. & BSG, Client/Server Computing, SAMS, 1992.
10. Dale, R., "Client-Server Database: Architecture of the Future", Database Programming and Design, August 1990.
11. Kiernan, C., "Client/Server: Learning From History", Database Programming and Design, September 1993.
12. Sinha, A., "Client/Server Computing: Current Technology Review", Communications of ACM, Vol. 35, No. 7, July 1992.
13. Stephenson, P., "Client/Server or Cooperative Processing", Database Programming and Design, September 1991.
14. Winsberg, P., "Client/Server Forum: Designing for Client/Server", Database Programming and Design, July 1993.