데이터 복제 서버를 이용한 학사 관리시스템의 부하 분산에 관한 연구

한 광 록[†] · 이 승 원[††]

요 약

본 논문에서는 학사관리 시스템의 부하 집중 현상을 처리하기 위하여 기존의 중앙 집중형 시스템을 그대로 사용하면서 추가적으로 분산 환경의 복제 서버를 구축하고 이것을 이용하는 방법을 제안한다. 데이터의 수정을 요구하지 않는 일반 조회 트랜잭션이 DML 문장의 대부분을 차지하기 때문에 각 데이터의 특성에 따라 분리된 복제서버를 구성하여 복제 서버가 수정이 필요없는 조회 트랜잭션만을 담당하게 함으로써 사용자 분산과 데이터 분산을 동시에 실행하여 실제 처리 시간을 줄일 수 있도록 하였다. 또한 기존의 집중형 시스템의 자원들을 그대로 사용할 수 있기 때문에 구현상의 편리성과 경제성을 도모할 수 있다. 일반적으로 서버의 부하부산을 위한 방법은 크게 사용자의 프로그램이 서버 전단에서 서버의 부하정보를 획득하여 상대적으로 적은 부하를 가진 서버를 선택할 수 있는 클라이언트단에서의 부하분산과 응용계층 스케쥴링 기법, IP 계층 스케쥴링 기법 등을 사용한 서버단에서의 부하분산이다. 본 논문의 복제서버에서는 기존의 부하분산기법에서 발생하는 단점들을 제거 또는 보완하여 시스템의 부하 집중현상을 줄일 수 있다.

A Study on the Distribution of Overload in Academic Affairs Management System Using Replication Server

Kwang Rok Han[†] · Seung Won Lee[††]

ABSTRACT

In order to solve the overload of academic affairs management system, we propose a method that builds a distributed Replication server and uses this server with the present centralized system. Normal query transactions which are not required for data modification are composed of almost all DML sentences. So we construct the distributed replication servers according to the data characteristics and make them perform the query transaction without modification. In this way, we can simultaneously distribute users and data, and cut down processing time for every transactions. Also Replication server has the advantages of implemental efficiency and economical benefit because it uses resources of present centralized system without any additional configurations. Usually, to distribute the overload of server, they can use one way, Client-side overload distribution that user program get present overload status then can choose a less overloaded server, and the other way, Server-side overload distribution that make use of Application Layer Scheduling Technique and IP Layer Scheduling Technique. Our Replication server can reduce the overload of centralized system by eliminating or complementing those defects of overload distribution, referred to in the forehead.

키워드 : 데이터베이스(database), 분산(distribution), 복제(replicaqtion), 부하(overload)

1. 서 론

데이터베이스 시스템 기술이 나날이 발전하고, 데이터베이스의 응용분야가 다양화해짐에 따라 기존의 중앙집중식 데이터베이스 시스템으로는 처리하기 복잡한 다양한 기능들이 요구되고 있다. 정보통신의 발전으로 인한 사용자의 급속한 증가는, 1990년대 이후 데이터베이스 환경을 기존의 메인 프레임 중심에서 분산처리로의 변환으로 이끌고 있다.

관계형 데이터베이스를 이용한 정보통합시스템의 구축은 1970년 Codd 에 의하여 그 모델이 정립된 이래 일반화되어진 방법이다[11]. 이러한 관계형 데이터베이스를 이용한 시스템은 수학적으로 잘 정의되어 있고 모델이 간단하다는 장점이 있다. 이러한 관계형 모델들은 CAD/CASE, 멀티미디어와 같은 형태를 모델링 하기보다는 기업이나 은행, 학교등 레코드 형태의 데이터 처리에 적합하다[12].

관계형 통합시스템의 구조와 성능은 클라이언트 프로세스와 서버 프로세스에 어떠한 구조로 DBMS 모듈이 배치되는 가에도 크게 좌우된다. 클라이언트와 서버에 DBMS 모듈을 배치하는 방법에는 크게 세 가지 방법이 있을 수 있다. 즉, 서버가 단순히 데이터베이스 저장장치의 역할을

† 종신회원 : 호서대학교 벤처전문대학원 컴퓨터응용기술팀장
†† 준 회 원 : 명지대학교 종합정보화 프로젝트 참여
논문접수 : 2000년 12월 4일, 심사완료 : 2001년 8월 9일

담당하는 경우, 서버가 기본적인 관계연산자를 제공하는 경우, 그리고 서버에서 질의 처리기의 역할인 파싱(Parsing)과 질의 변환까지 수행하는 경우이다[1, 11].

현재 관계형 통합시스템의 구조는 다음의 경우 시스템 구축에 소요되는 시간, 경비 등의 이유로 인하여 하나의 서버에 DBMS 모듈이 집중되어 있는 경우가 많다.

그러나 다수의 사용자를 지원해야 하며 상당한 비율의 트랜잭션이 발생하는 시스템 상에서는 이러한 중앙집중형 시스템의 경우 성능과 효율성이 떨어질 가능성이 많으며 실제 구현된 중앙 집중형 시스템 상에서 다수의 조회 트랜잭션으로 인하여 낮은 빈도의 수정 트랜잭션 또한 영향을 받는 것이 사실이다. CIM(Computer Integrated Manufacturing)이나 사무자동화 응용 시스템에서는 클라이언트/서버 환경의 분산 처리가 필수적이라 할 수 있다.

이에 본 논문에서는 Oracle 8에서 지원하는 Server Replication[2]을 이용하여 학사 행정 시스템에서 다수 사용자와 잦은 트랜잭션으로 인한 서버의 성능 저하 요소를 제거하고, 서버 집중으로 인한 단점을 보완할 수 있는 시스템 환경을 제안한다.

본 논문의 구성은 2장에서는 서버의 부하를 줄이기 위한 기존의 분산 관련 연구들에 대하여 살펴보고 3장에서는 본 논문에서 제안한 분산환경에서 데이터 복제를 이용한 OLTP 성능개선 방안과 실제 시스템의 적용사항에 대해 기술하며, 4장에서는 이러한 적용사항의 성능개선 부분에 대한 실험결과를 제시하고, 마지막 5장에서 결론을 맺는다.

2. 기존 부하 분산 연구

국외의 경우 많은 대학과 연구소 그리고 기업들이 웹서버의 부하 분산 문제에 대한 연구를 매우 활발히 진행하고 있으며, 많은 상용화된 제품을 선보이고 있다. 이러한 부하 분산 기법은 크게 사용자 측면에서의 부하 분산과, 서버 측면에서의 부하 분산 기법으로 나누어 볼 수 있다.

2.1 사용자 측면에서의 부하 분산

사용자측에서의 부하 분산 기법은 사용자가 클러스터링된 웹서버들 중 부하가 상대적으로 적게 걸리는 서버를 동적으로 선택할 수 있도록 하는 방법으로 Berkely smart client[3]와 Bandwidth Probing을 이용한 동적인 서버 선택 등이 있다[4].

Berkely smart client는 사용자에게 클러스터링된 웹서버들에 대한 부하 정보를 얻어올 수 있는 애플릿을 제공해 사용자의 접속을 분산시킴으로써 웹서버의 부하를 분산시키는 기법을 이용하고 있다.

그러나 사용자 측면에서의 부하 분산은 사용자의 응용프로그램을 바꾸어야 한다는 점과 네트워크 부하를 증가시킨다는 단점 때문에 대부분의 연구는 서버 측면에서의 부하 분산 문제에 대해 다루고 있다. 이러한 사용자 측면의 단점을 보완하고자 본 논문에서의 복제서버 이용은 실제 사용자가 접속하는 서버의 경로를 변경함으로써 전면적인 사용자 응용프로그램의 수정이 필요하지 않으며 각 단위 사용자들에 대한 복제서버를 제공 전체적인 네트워크 부하 감소의 장점이 있다.

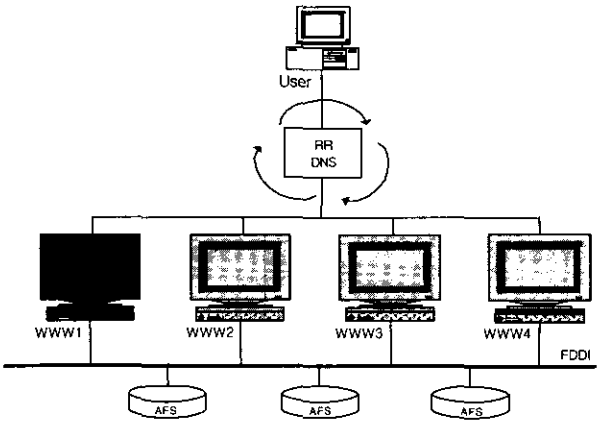
2.2 서버 측면에서의 부하 분산

서버 측면에서의 부하 분산은 사용기법에 따라 RR-DNS (*Round Robin Domain Name System*), 응용계층 스케쥴링 기법, IP 계층 스케쥴링 기법으로 나누어 볼 수 있다[5].

서버측에서 부하 분산을 사용하는 방법은 RR-DNS 기법을 사용하는 NCSA 확장가능 웹서버가 대표적이며[6], RR-DNS 기법과 응용계층 스케쥴링 기법을 사용한 것으로는 SWEB[7], 그리고 IP 계층 스케쥴링 기법을 채용한 것으로는 IBM사의 TCP_ROUTER[8]와 Cisco사의 Local Director[9] 등이 대표적이다.

2.2.1 NCSA 확장가능 웹서버

NCSA 확장가능 웹서버는 각기 다른 IP에 같은 도메인 주소를 할당하고 라운드 로빈 방식에 따라 사용자의 접속 웹서버를 다르게 하는 RR-DNS방식을 이용해 부하를 분산시킨다(그림 2-1). 그러나 이러한 방식은 RR-DNS에 의하여 부하 불균형이 일어날 수 있고, 시스템 정지 시 대처할 수 있는 능력이 없다는 것이 문제점이다. 이러한 단점을 본 논문에서는 각 단위 사용자에게 제공하는 복제 서버를 이용하여 중앙의 메인 서버 정지시에도 데이터의 입력/수정을 제외한 복제서버로의 조회 기능은 이용할 수 있는 장점이 있다.

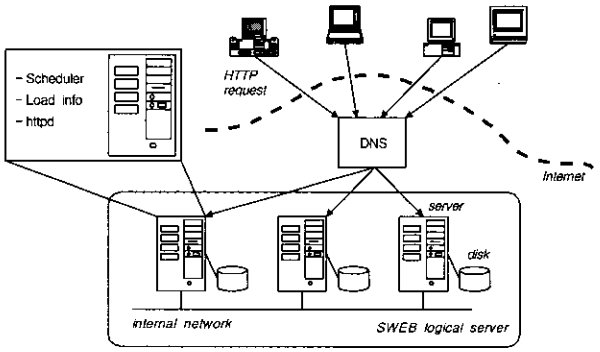


(그림 2-1) NCSA 확장가능 웹 서버

2.2.2 SWEB

SWEB은 먼저 RR-DNS에 의해 사용자의 접속요구를 분산시키고, 클러스터링된 웹서버들의 불균형 부하 분산을 막기 위해 모든 웹 서버들이 다른 웹 서버들에 걸리는 부하

에 대한 정보를 가지고 이에 따라 사용자의 접속을 분산시키는 기법을 이용한다(그림 2-2). RR-DNS의 문제점을 극복했지만, 모든 웹 서버들이 부하에 대한 정보를 유지해야 하므로 성능저하의 문제점이 있다.

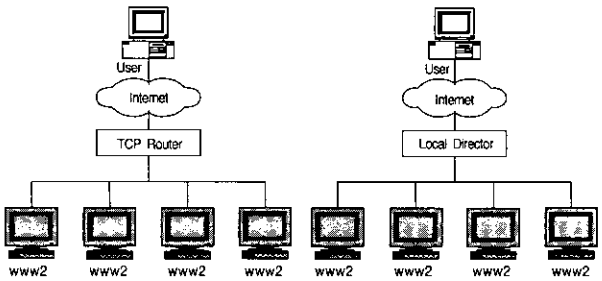


(그림 2-2) SWEB

본 논문에서의 복제서버는 각 단위 사용자들이 사용할 데이터 이외에 부가적으로 가지고 있어야 할 정보가 필요 없다는 장점이 있다.

2.2.3 TCP Router와 Local Director

(그림 2-3)에서 보여주고 있는 IBM의 TCP Router, Cisco 사의 LocalDirector 같은 경우 IP 계층에서의 네트워크 주소 변경기법을 채용한 고가의 라우터들이며, 성능에 비해 상대적으로 고가라는 단점이 있다[6]. 또한 사용자와 가상서버, 가상서버와 웹서버 사이의 추가적인 접속 연결이 필요하지는 않지만, 여전히 가상서버에 집중된 부하와 목적지 주소 변경으로 인한 가상서버의 부하로 확장성이 결여되게 된다.



(그림 2-3) TCP 라우터와 LocalDirector

2.2.4 부하 분산 기법의 비교

위에서 설명한 부하분산 기법들의 문제점을 분석해 보면 RR-DNS의 문제점을 해결하기 위해선 응용계층 스케쥴링 기법이나 IP계층 스케쥴링 기법을 사용해야 하나 이들을 사용하면 분산웹서버의 확장성이 떨어진다는 문제점을 갖고 있음을 알 수 있다.

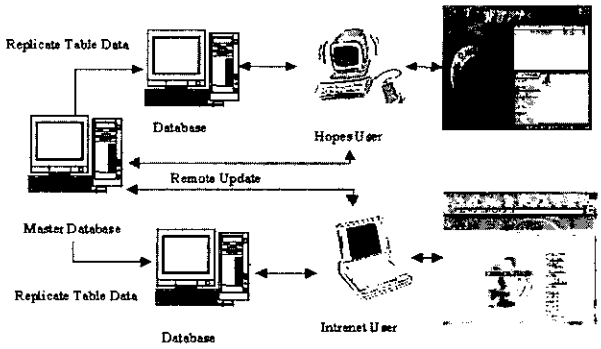
사용기법에 따른 부하 분산 기법과 본 논문에서의 개선사항을 비교 정리하면 <표 2-1>과 같다.

〈표 2-1〉 분산 웹서버의 부하 분산 기법

사용기법	문 제 점	해 결 책	Trade-off	본 논문의 개선사항
RR-DNS	부하 불균형 문 제	DNS의 엔트리테이블 TTL 값을 1에 가깝게 설정	DNS에서의 병목현상 발생	복 제 서 버 를 이용한 부하 분산 기존 시스템의 이용을 통한 비용 절감
		SWEB 사용	웹서버들의 부하	
		RR-DNS+TCP_ROUTER	고비용	
응용계층 스케쥴링	가상서버의 병목현상	-	-	복제서버에서의 부가 정보 유지 불필요
	네트워크 트래픽 증가	IP 계층 스케쥴링 기법 이용	가상 서버의 병목현상	
IP 계층 스케쥴링	가상서버의 병목현상	-	고 비 용	기존 시스템의 이용을 통한 비용 절감

3. 데이터 복제를 이용한 분산 시스템의 설계

현재 대부분의 대학 종합정보 시스템의 구성은 하나의 Oracle 서버상에 DBMS의 모든 기능이 집중되어 있는 중앙 집중식 서버 시스템으로 구성되어 있다. 이러한 구성의 서버 시스템은 다수의 사용자가 집중되었을 경우 그 효율성은 상당히 떨어지게 된다. 따라서 본 논문에서 (그림 3-1)와 같이 데이터 복제(data replication)를 이용한 분산 시스템을 구성한다.



(그림 3-1) 복제 구성을 통한 시스템 구성

복제 구성을 통한 시스템 구성은 기존의 중앙 집중식 서버 시스템에서의 문제점을 해결하고자 복제(Replication)이라고 하는 분산 환경을 이용하여 사용자를 분산시킴으로서 전체적인 시스템 성능을 높일 수 있다.

(그림 3-1)의 시스템은 하나의 이상의 복제 서버에 읽기 전용 데이터를 존재시키고 실제 사용자들은 데이터의 수정을 필요로 하는 트랜잭션 만을 중앙 데이터베이스에서 실행시키게 된다[2, 10]. 이러한 읽기 전용 데이터의 분리는 실제 한 조사에 의하면 일정기간동안 실행된 DML 중에서 75.9%가 조회, 즉 SELECT 문장이었음에 근거를 두고 있다[13]. 따라서 조회에 필요한 데이터에 대한 복제 서버 구축과 사용자의 소속 등, 서버분산 계획에 따른 데이터베이스 연결 모듈 작성이 필요하다. 특히 대학의 학사 행정에

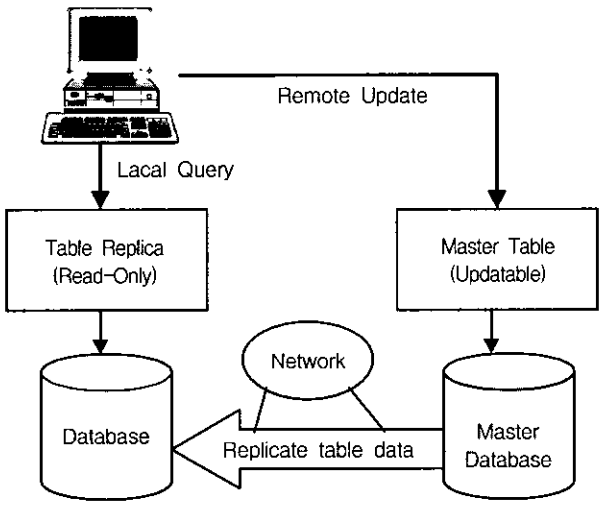
있어서 학부 교수와 학부 학생들 간에 이루어지는 빈번한 조회로 인하여 부하가 집중되기 때문에 사용자의 소속에 따른 분산이 요구된다.

3.1 복 제

복제(Replication)는 분산 데이터베이스 시스템을 구성하는 다수의 데이터베이스 상에 데이터베이스 객체들을 복재하고 유지하는 일련의 과정이다[2]. 마스터 데이터베이스로부터 만들어진 데이터 복제본을 제공함으로써 클라이언트 어플리케이션은 지역 데이터 복제본에 트랜잭션을 보낼 수 있고 수정이 필요할 경우에만 시스템을 통해서만 중앙 데이터베이스의 데이터에 액세스 할 수 있게 된다.

이 외에도 향상된 복제(Advanced Replication) 형태로 단순한 읽기전용 데이터 가 아닌 다수의 중앙 사이트를 구성할 수도 있다.

본 논문에서는 빈번히 발생하는 조회 업무에 대한 부하를 분산시키기 위해 전자의 방법으로 (그림 3-2)와 같은 읽기 전용 복제 시스템을 구축하였다.



(그림 3-2) 읽기전용 복제

3.2 복제 서버 구축

분산환경의 복제 서버를 구축하기 위해서는 다음과 같은 과정을 거친다.

① 복제 환경의 설계
읽기전용 데이터로 만들 기본 데이터의 결정.

② 링크 생성
각 사이트에 스키마와 데이터베이스 링크 생성.

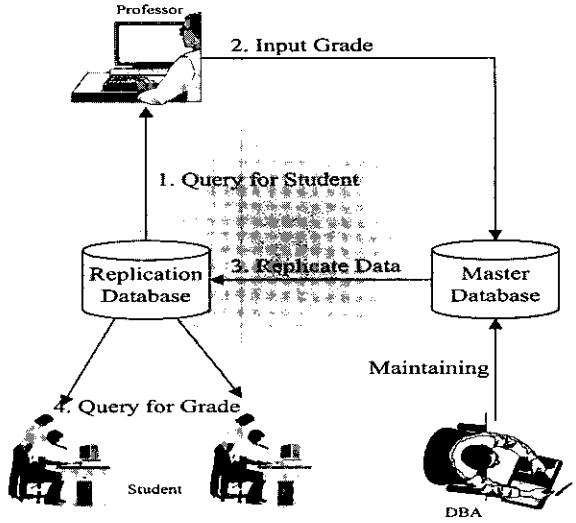
③ 기본 사이트 구성
기본 사이트에 복제환경을 지원할 수 있도록 설정.

④ 복제 서버 생성

⑤ Refresh Group 생성

⑥ 사용자에 대한 권한 부여

(그림 3-3)은 학사행정 시스템 중 가장 많은 동시 사용자가 집중되는 성적시스템에서 복제 서버를 활용하여 조회를 분산시키는 구성을 나타낸다.

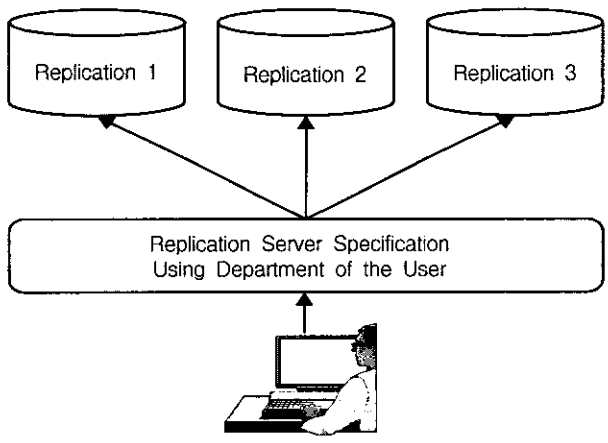


(그림 3-3) 복제를 이용한 성적시스템의 구성

〈표 3-1〉 사용자 구분

Client	Query 종류	Access DB
교 수	조 회 입 력	복제 서버 중앙 서버
학 생	조 회	복제 서버

교수가 개설과목에 대하여 성적을 입력하기 위해서는 해당 과목의 수강학생의 정보를 가져와야 한다. 이러한 단순 조회는 해당 데이터가 들어있는 복제 서버를 통하여 데이터를 가져올 수 있다. 최종적으로 교수가 성적을 입력하게 되면 해당 데이터는 기본 데이터베이스 상에 저장되며 DBMS 에 의하여 복제 서버에 복제가 이루어진다. 이러한 과정상에서의 데이터 복제 갱신 주기의 설정은 해당 데이



(그림 3-4) 사용자 소속에 따른 서버 할당

터의 크기와 트랜잭션의 종류, 네트워크와 서버의 용량에따라 달라진다. 학생들의 성적의 조회 또한 복제되어 있는 복제 서버를 통하여 가능하게 된다.

<표 3-1>은 사용자에 따른 Query의 종류와 접근하는 서버를 나타낸다.

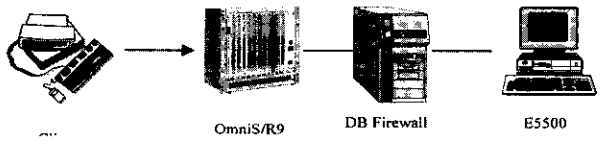
이러한 사용자의 분산은 (그림 3-4)과 같이 소속에 따라 이루어지게 된다.

4. 시스템 구현 및 실험

4.1. 시스템 구현

본 실험은 다수의 사용자가 하나의 서버에 집중되었을 때, 그 서버의 성능저하를 수치적으로 보이기 위하여 다수 사용자 환경의 시뮬레이션 프로그램을 작성, 동일한 트랜잭션의 수행시간 차이를 기술하고자 한다. 실험은 동시 연결된 사용자를 늘려가면서 같은 트랜잭션의 CPU 사용시간을 확인토록 하여 사용자 분산에 의한 성능개선 부분을 증명하려 한다. 사용한 트랜잭션문은 200개가 넘는 레코드를 조회하고 많은 디스크 입출력을 수행토록 작성하였다.

본 실험에서 사용된 DBMS는 Oracle8 Enterprise Edition Release 8.0.5.1.0과 PL/SQL Release 8.0.5.1.0이다. 데이터베이스 서버는 SUN Enterprise 5500이며 OS 는 Sun OS 5.7이다. CPU Clock Speed는 400MHz이며 메모리 캐쉬는 4MB, 채널 당 I/O Bandwidth는 200 Mbps이며 이러한 것은 총 클라이언트의 30%가 동시사용자이고 분당 트랜잭션 은 5건으로 추산된 것이다. 또한 시뮬레이션 프로그램을 작성하는 클라이언트와 데이터베이스 서버간의 네트워크는 10Base-T/100BaseT를 지원한다.



(그림 4-1) 실험환경

시뮬레이션 환경을 구현하기 위한 개발 도구는 파워빌더 6.5를 사용하였다. 파워빌더는 클라이언트와 데이터베이스 서버간에 Transaction이라는 객체를 지원한다. 본 실험에서는 파워빌더에서 제공하는 Transaction 객체를 사용하여 입력받은 수만큼 서버와의 세션을 설정하고 각 세션당 트랜잭션을 수행시키도록 하였다. (그림 4-1)은 실험을 위한 시스템 환경을 나타낸다.

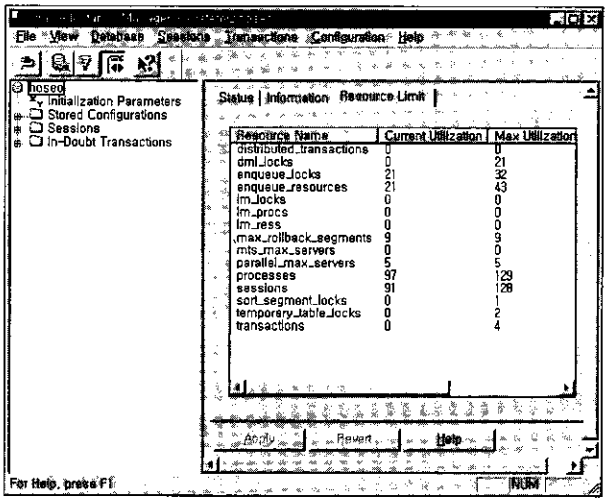
4.2 실험

본 논문의 실험은 중앙 집중식 서버 환경에서 50명과 70명이 동시 접근했을 때와 복제 서버를 이용한 분산환경에서 실험을 행하여 그 결과를 비교하였다.

비교 데이터는 지면 관계 상 70명에 대한 중앙 집중식과 복제 서버에 의한 분산 환경을 비교하였다.

4.2.1 중앙 집중식 서버 환경에서의 실험

(그림 4-2)에서 나타난 바와 같이 현재의 세션은 91개, 프로세스는 97개가 사용중이다. 이러한 상태에서 특정 트랜잭션의 수행시간은 오라클에서 제공하는 TKPROF 라는 유틸리티로 살펴볼 수 있다.



(그림 4-2) 오라클 인스턴스 관리자

이러한 서버 상태에서 계속해서 동일한 트랜잭션을 수행시킨 결과를 (그림 4-3)에 나타내었고, 실행 결과의 화면에 나타난 용어를 <표 4-1>에 정리하였다.

```
TKPROF : Release 8.0.5.1.0 - Production on Mon Nov 13 13 : 17 : 8 2000
(c) Copyright 1998 Oracle Corporation. All rights reserved.
Trace file : hoseo_ora_3327.trc
Sort options : default
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.03	0.03	0	0	0	0
Execute	2	0.04	0.04	0	0	0	0
Fetch	5	5.16	14.06	6358	72509	319	228
total	8	5.23	14.13	6358	72509	319	228

(그림 4-3) 91개의 세션 상태에서의 수행결과

〈표 4-1〉 용어 해설

결 과	의 미
count	number of times OCI procedure was executed
cpu	cpu time in seconds executing
elapsed	elapsed time in seconds executing
disk	number of physical reads of buffers from disk
query	number of buffers gotten for consistent read
current	number of buffers gotten in current mode (usually for update)
rows	number of rows processed by the fetch or execute call

본 논문에서는 이외에도 10명 50명의 동시 사용자 접속

에 대해서도 실험을 행하였고 그 결과를 간략하게 (그림 4-5)에 나타내었다. (그림 4-5)에서 알 수 있는 바와 같이 70명의 동시 사용자 환경에서의 동일 트랜잭션의 수행결과는 그 차이가 50명의 동시 사용자보다 그 차이가 더 커짐을 알 수 있다. 또한 디스크 입출력 횟수 또한 늘어났으며 이는 SGA(Shared Global Area)라는 오라클 메모리 구조에서 다수의 사용자 세션에 대하여 각각의 메모리 영역 할당으로 인한 Cache Hit 율의 저하로 인한 것임을 알 수 있다. 또한 모든 실험결과의 쿼리 문장의 파싱은 1회, 실행은 2회, Fetch는 5회로서 동일하게 동작하였으며 이것은 수행시간과 디스크 입출력 횟수 등을 제외한 수행과정은 동일함을 보여준다.

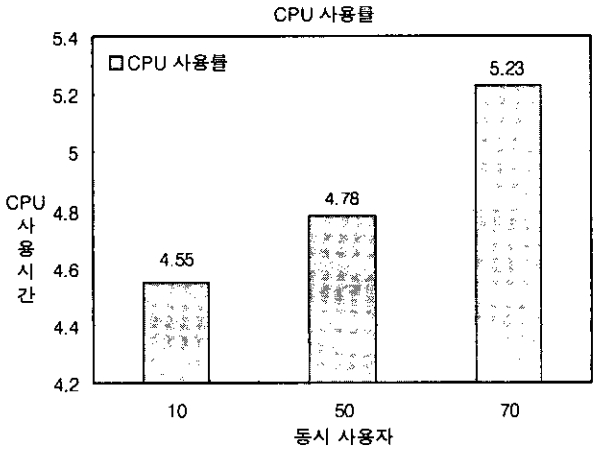
이러한 CPU 수행시간은 단 하나의 레코드를 리턴하는 트랜잭션을 수행한 결과를 보면 그 시간차이가 작지 않음을 알 수 있다.

```
select * from *****
where ***** = '799032'
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.01	0.01	0	0	0	0
Execute	1	0.01	0.01	0	0	0	0
Fetch	1	0.00	0.00	0	3	0	1
total	3	0.02	0.02	0	3	0	1

(그림 4-4) 간단한 트랜잭션의 수행결과

이 트랜잭션은 단 하나의 레코드를 리턴하며 CPU 시간은 0.02가 걸렸다. 이러한 결과를 토대로 사용자의 증가에 의한 4.55/304(CPU 시간/ 디스크 입출력) -> 5.23/6358의 증가가 상당한 서버의 성능 저하를 가져올 수 있음을 알 수 있다.



(그림 4-5) 동시 사용자 수에 따른 평균 CPU 사용률 변화

4.2.2 사용자 분산 실험

본 논문에서 제안한 구성도에 따른 서버의 성능 개선에 관한 실험을 하였다. 실제 실험은 70명의 동시사용자 환경에서 두가지로 나누어 실험하였다. 첫 번째는 n, n+1의 세션은 조회 트랜잭션을, n+2는 수정 트랜잭션을 수행하도록 하였고, 두 번째는 n, n+1의 세션은 아무런 동작을 하지 않으며 n+2의 세션에서 첫 번째와 동일한 수정 트랜잭션을 수행하도록 작성하였다. 이것은 조회부분은 복제 서버가 맡은 부분을 가정한 것이다.

이러한 두 가지 상황하에서 10000개 이상의 레코드를 검색한 후에 그 결과를 삽입하는 특정 트랜잭션을 수행하였을 때의 성능 차이는 (그림 4-6)와 같다.

```
insert into ******
(select rownum, **********
from ********)
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.02	0.02	0	0	1	0
Execute	1	4.10	30.68	4754	8738	31423	104148
Fetch	0	0.00	0.00	0	0	0	0
total	2	4.12	30.70	4754	8738	31424	104148

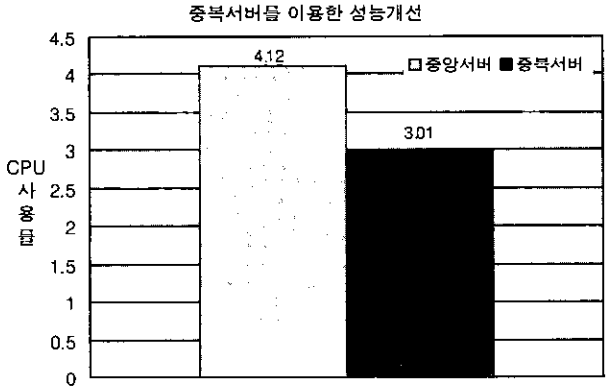
```
insert into ******
(select rownum, **********
from ********)
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	3.01	11.40	1527	7097	13984	104165
Fetch	0	0.00	0.00	0	0	0	0
total	2	3.01	11.40	1527	7097	13984	104165

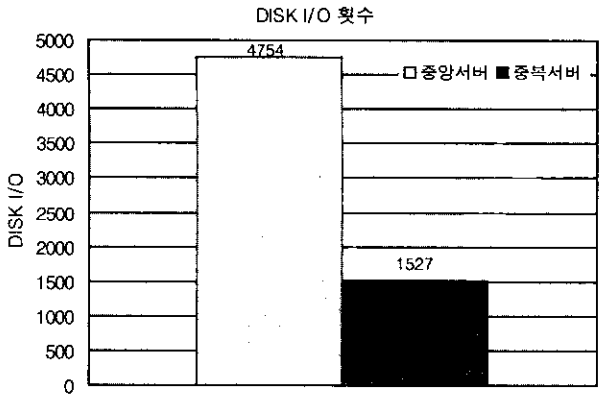
(그림 4-6) 분산된 복제 서버 환경에서 삽입 트랜잭션의 수행결과

(그림 4-6)의 결과는 26.9%의 CPU 사용시간 감소와 68%의 DISK I/O 의 감소를 보이고 있다. 즉, 복제 서버의 구성을 통하여 조회와 수정 트랜잭션을 분리함으로써 동일 서버의 성능개선을 이룰 수 있음을 알 수 있다.

(그림 4-7)과 (그림 4-8)에 중앙 집중형 서버와 복제 서버 환경에서의 특정 트랜잭션의 수행결과를 비교하여 나타내었다.



(그림 4-7) CPU 사용시간의 비교



(그림 4-8) Disk/IO의 비교

4.3 고 찰

실험결과로부터 다음과 같은 내용을 확인할 수 있었다.

① 사용자 분산으로 인한 중앙서버의 부하 감소
초기 접속이후의 작업을 복제 서버를 사용함으로써 중앙서버의 부하 분산을 통한 성능 향상을 기대할 수 있다.

② 신뢰성과 가용성의 향상
중앙서버에 고장이 발생하더라도 데이터 수정 이외의 작업은 가능하다.

③ 확장의 용이성
기존의 중앙서버에 변화를 주지 않으면서 복제 서버를 구축함으로써 편리성과 경제성을 도모할 수 있다.

데이터 일관성의 문제는 모든 데이터베이스 시스템 평가에서 중요한 관점이었다. 본 논문에서 제시하는 복제 서버 시스템 또한 갱신 주기에 따른 시간상의 일관성문제는 존재한다. 그러나 이러한 단점은 각 시스템의 조회항목에 따른 조정을 정해서 조회기간을 설정할 수 있기 때문에 이러한 단점은 배제하였다.

5. 결 론

본 논문에서는 중앙 집중식 서버 형태가 사용자의 증가와 잦은 디스크 입출력을 수반하는 트랜잭션 상태에서 서버의 효율성의 저하를 필연적으로 수반함을 실험을 통하여 증명하였으며 이러한 문제점을 해결하기 위하여 분산 데이터베이스 환경의 복제 서버 형태를 제안하였다.

복제 서버 환경에서 사용자의 소속 부서에 따른 지역 데이터베이스 할당으로 사용자의 분산이 가능하게 되었으며 대다수의 조회를 중앙 데이터베이스가 아닌 복제 데이터베이스를 이용함으로써 빠른 결과를 획득할 수 있으며 이는 전체적인 시스템 성능 향상 또한 가능하게 할 수 있다.

네트워크 부하를 줄일 수 있는 방안과 함께 앞으로 DBMS의 기능과 분산객체의 서비스 등을 이용하여 사용자가 조회하는 데이터와 실제 중앙 서버상의 데이터의 일관성 문제 등을 해결하여 보다 정확하고 실제적인 연구를 진행할 예정이다.

참 고 문 헌

[1] Ramez Elmasri, "Fundamentals of Database Systems 2nd Edition," Addison Wesley, 1994.

[2] Oracle8™ Server Replication Release 8.0, Part No.A54651-01, June, 1997.

[3] Chad Yoshikawa, *et al*s, "Using Smart Clients to Build Scalable Services," http : //now.cs.berkeley.edu, USENIX '97, 1997.

[4] Robert L.Carter, Mark E.Crovella, "Dynamic Server Selection Using Bandwidth Probing in Wide-Area Networks," http : //www.ncstrl.org, Boston University Technical Report, 1996.

[5] Wensong Zhang, Shiyao Jin, Quanyuan Wu National Laboratory for Parallel & Distributed Processing, "Creating Linux Virtual Servers," http : //proxy.iinchina.net/~wensong/ppfvs/linuxexpo.html.

[6] Eric Dean Katz, Michelle Butler, and Robert Mcgrath, "A Scalable HTTP Server : The NCCA Prototype," Computer Networks and ISDN Systems, pp.155-163, 1994.

[7] Daniel Anderson, Tao Yang, Vegard Holmedahl, and Oscar H. Ibarram "SWEB : Towards a Scalable World Wide Web Server On Multicomputers," Proceedings of International Conference on Parallel Processing, pp.15-19 April, 1996.

[8] D. Dias, W. Kish, R. Mukherjee and R. Tewari, "A Scalable and Highly Available Server," COM-PCON 1996, pp.85-92, 1996.

[9] Cisco System, "Cisco Local Director," http : //www.cisco.com/warp/public/751/lodir/index.html.

[10] Meghraj Thakkar, "Oracle 8i on Windows NT," SAMS, 1999.

[11] 여지황, 김형주, "SQL 질의 처리기의 클라이언트-서버 모듈 배치에 따른 성능 비교", 정보과학회논문지(C) Vol.3, No.3, pp.217-227, 1997.

[12] 박상원, 김형주, "관계형 데이타베이스의 객체지향적 인터페이스를 위한 게이트웨이의 설계 및 구현", 정보과학회논문지(C), Vol.3, No.4, pp.333-342, 1997.

[13] 이석호, "SQL 환경에서의 관계 데이터베이스 작업부하 분석", 한국정보과학회논문지 Vol.17, No.2, pp.152-162, 1990.

한 광 록

e-mail : krhan@office.hoseo.ac.kr

1984년 인하대학교 전자공학과 졸업(공학사)

1986년 인하대학교 대학원 정보 공학전공 (공학석사)

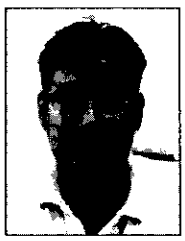
1989년 인하대학교 대학원 정보 공학전공 (공학박사)

1989년~1991년 한국체육과학원 선임 연구원

1991년~2000년 현재 호서대학교 컴퓨터공학부 교수

1999년~현재 호서대학교 벤처전문대학원 컴퓨터응용기술팀장

관심분야 : 정보검색, 자연언어처리, 기계번역, HCI, 지능형 에이전트 등



이 승 원

e-mail : seungwlee@lgeds.lg.ac.kr

1998년 호서대학교 컴퓨터공학과 졸업 (학사)

2001년 호서대학교 컴퓨터공학과 졸업 (석사)

1999년~2000년 호서대학교 전산실

2000년~현재 명지대학교 종합정보화 프로젝트 참여

관심분야 : 분산시스템, 데이터베이스