임시 테이블을 사용한 분산 데이타 할당 방법 및 구현

허 계 범[†] · 이 종 섭[††] · 정 계 동[††] · 최 영 근[†††]

요 약

분산 데이타베이스 할당 방법의 데이타 복제 기법은 많은 화일을 갱신시에 복제본 사이의 일관성 유지를 위해 모든 복제본을 함께 갱신해 주어야 하는 오버헤드가 발생한다. 데이타 이주 문제에 있어서도 수직, 수평 분할 이주시 데이타 통신 비용이 증가된다.

본 논문에서는 화일 복제와 이주시 관련된 테이블의 항목들을 그룹화하여 임시 테이블에 저장일괄 처리한다. 이 방법은 데이타 입력 및 갱신이 많은 분산 트랜잭션 처리에 있어 시스템 버퍼 사용 효율을 높여준다. 결국 디스크 I/O 처리 시간과 지역간 통신 비용을 줄임으로써 트랜잭션 처리의 병행성을 높이고 데이타 처리 속도를 빠르게 한다.

Distributed Data Allocation Methods and Implementations using the Temporary Table

Kwae Bum Heo[†] · Jong Sup Lee[††] · Kwae Dong Jeong[††] · Young Eun Choi[†††]

ABSTRACT

Data replication techniques of distributed database allocation methods occures to overhead to change together all replicas for consistency maintenance at updates. In case of data migration horizontal or vertical fragment migration has caused to increase data communication.

In this paper we purpose batch processing method groups the associated items of table and stores temporary table associated with file relication and migrations. This method increase the usability of system buffer in distributed transaction processing with a number of data inputs and updates. After all, it improved the performance of distributed transaction system by reducing disk I/O processing time and the cost of data communication among local sites.

1. 서 론

분산 환경이란 여러 컴퓨터를 기반으로 시스템과 어플리케이션이 구축되는 것으로서, 프로세싱과 여러 시스템 요소들이 하나의 플랫폼이 아닌 여러 플랫폼에 분산 되어 있는 것을 의미한다.

이러한 시스템의 장점은 모든 자원이나 제어 기능을 각 노드에 복제(replication) 또는 분산(distribution) 시킴으로써 전체 시스템의 성능 향상과 유용성(usability), 확장성(extensibility), 신뢰도(reliability)등과 같은 면에서 중앙 집중 처리 시스템보다는 잇점을 얻게 된다 [8, 13]. 그러나 분산 처리의 성능은 다음과 같은 단점

† 정 회 원:광운대학교 전자계산학과 박사과정
†† 준 회 원:광운대학교 전자계산학과 박사과정
††† 정 회 원:광운대학교 전자계산학과 교수
논문접수:1996년 7월 10일, 심사완료:1997년 4월 30일

이 있다[11, 13].

첫째, 분산된 중복 데이타에 대해 많은 수의 병행 갱신을 요청하는 작업의 처리시 오버헤드가 발생한다.

둘째, 분산 처리 시스템 설계 및 구현은 복잡도가 증가한다. 따라서 시스템 개발 비용과 유지 보수 비용이 증가한다.

셋째, 분산 데이타베이스 처리는 제어가 어렵다. 그러므로 장애가 발생하면 회복 수행이 훨씬 더 어려워진다.

이와같은 문제점을 해결하기 위해 본 논문에서는 임시 테이블을 사용한 데이타 할당(data allocation) 방법을 논하고자 한다. 할당 방법은 데이타 분할후 컴퓨터 네트워크상에서 화일에 대해 질의, 갱신 및 실행을 위해서 서로 다른 노드들에게 화일을 할당하는 것을 말한다. 이러한 방법으로는 데이타 복제(data replication) 방법과 데이타 이주(data migration) 방법이 있다[8, 13]. 복제 방법은 모든 지역에 동일한 전체 데이타베이스를 유지하는 완전 복제(fully replicated) 방법과 데이타베이스 일부분은 분할된 상태로 존재하고, 일부분은 여러 지역에 부분 중복 존재하는 부분 복제(partially replicated) 방법이 있다. 완전 복제 방법은 모든 검색 요구를 지역적으로 처리할 수 있으므로 검색 비용이 매우 낮아진다. 그러나 동일한 데이타가 여러 지역에 존재하기 때문에 갱신 비용이 높을 뿐 아니라 각 데이타들의 일관성(consistency) 유지를 위한 방법이 아주 복잡해진다[3]. 이주 방법은 많이 참조하는 지역과 컴퓨터 중앙 처리장치의 오버헤드를 고려하여 해당 지역에 데이타를 이주시키는 방법으로 중복을 허용하는 복제 방법에 비해 기억 장소의 낭비가 없고 여러 복제본 사이의 일관성 유지를 위한 오버헤드도 없다. 그러나 전체 화일 이주가 아닌 수직 분할(Vertical Fragmentation : VF) 및 수평 분할(Horizontal Fragmentation : HF) 이주시 많은 트랜잭션이 일어날때 발생하는 통신 비용과 최소의 응답 시간을 보장 받을 수 없다.

본 논문에서는 화일 복제와 이주시 관련된 테이블의 항목들을 임시 테이블에 저장하여 처리 한다. 이 방법은 데이타 입력 및 갱신이 많은 분산 트랜잭션 처리에 있어 시스템 버퍼 사용 효율을 높여준다. 결국 디스크 I/O 시간과 지역간 통신 비용을 줄여 데이타의 병행 속도를 빠르게 하는데 그 목적이 있다. 따라서 클라이언트/서버 환경에서 사용자로부터 최소로 응답시간을 보장 받을 수 있도록 분산 트랜잭션 화일들을 어떻게 효율적으로 처리할 수 있는가에 대한 것으로 연구 범위는 아래와 같다.

첫째, 분산 데이타베이스 설계 방법을 제시하며,

둘째, 설계 방법을 토대로 클라이언트/서버 환경에서 임시 테이블을 사용한 효율적인 분산 트랜잭션 처리방법을 제시하며,

셋째, 실 시스템을 적용 성능 평가 분석을 하여 우수성을 입증하고자 한다.

논문의 구성을 살펴보면 2장에서는 관련 연구들을 살펴보고, 3장에서는 분산 환경하에서 임시 테이블을 이용한 효율적인 분산 트랜잭션 처리 방법을 제시하고, 4장에서는 실 시스템 적용 예와 성능 평가 분석을 하며, 5장에서는 결론 및 앞으로의 연구 방향에 관하여 서술하고자 한다.

2. 관련 연구

분산 처리 시스템에서 각 컴퓨터에 있는 사용자나 응용 프로그램은 원하는 정보를 얻거나 갱신하기 위하여 데이타베이스에 접근한다. 이러한 처리를 효율적으로 수행하기 위하여 지금까지 연구한 방법들을 살펴보면 다음과 같다[10].

1)컴퓨터 시스템을 지역적으로 분산 설계 하는 방법

2)분산된 컴퓨터 시스템을 연결하는 통신망을 구성하는 방법

3)데이타들을 적절히 분할하는 방법

4)분할된 데이타 화일들을 분산된 컴퓨터 시스템에 할당하는 방법

5)사용자의 요구 사항을 효율적으로 수행해 내기 위한 시스템의 운용 방안 등이 있다.

본 논문에서는 3), 4)항목의 방법들을 중심으로 알아보고자 한다.

2.1 데이타베이스 분할 방법

데이타베이스의 분할은 전역 데이타베이스가 가져야 할 모든 정보를 잃지 않으면서 다루기 쉬운 크기의 작은 단편들의 집합으로 나누는 것을 말하며, 지역간에 전송 되어야 할 데이타량을 줄임으로써 전체 처리 비용을 줄일 수 있다. 따라서 데이타베이스 분

할의 목적은 전송 비용의 절감에 있다고 할 수 있다[13].

데이타베이스 분할 방법에는 수직 분할과 수평 분할이 있으며, 이러한 분할 방법에 데이타베이스의 일부를 복제(replication) 병행시켜 하나의 속성이 여러 단편에 나타내는 방법으로 분류 할 수 있다. 관련 연구를 살펴 보면 Chang은 수평, 수직 분할외에 Stacking, Qualifield HP 그리고 Qualified Stacking등 많은 분할법을 정의 하였다[12]. Ceri는 수평 분할 프로세스의 입력으로서 정밀한 트랜잭션 명세 테이블의 표현 방법을 제시하였다[14]. Eisner와 Severance는 min-cut-max-flow 네트워크 분할 법을 이용하여 두 단계(two level) 메모리에 맞는 두 단계 분할 방법을 제시하였다[10]. 그 밖에 Wilson과 Navath는 트랜잭션들의 의미가 변화할때 전체 데이타베이스를 동적으로 다시 설계하는 이론을 제시하였으며, Navathe등은 Bond Energy Algorithm(BEA)을 이용하여 강한 응집력을 가지는 속성들이 한 단편내에 포함 되도록 하는 수직 분할 알고리즘을 제시하였다[4].

2.2 데이타베이스 할당 방법

분산 시스템에서 또 다른 문제는 화일이 효과적으로 처리되도록 여러 노드에 배치하는 할당 문제이다. 이러한 문제를 해결하기 위한 기법들은 최소 비용 측면과 성능면으로 주로 연구되어져 왔다. 최소 비용 모델에는 보통 화일 저장 비용, 질의의 비용, 화일 갱신 비용, 그리고 통신 비용들을 고려하여 최소 비용을 가지는 노드에 화일을 중복 배치하는 것이고, 성능 측면에서는 질의 및 갱신 비용, 통신 비용, 각 노드의 읽기/쓰기 처리 등의 요소를 고려한 비용을 상수로 두어 사용하여 최소의 응답 시간과 최대 시스템 처리율을 목적으로 화일을 할당한다[6, 13, 15].

위의 요소들을 고려한 연구 사례들은 다음과 같다.

BACA[3]는 복제 알고리즘과 voting 알고리즘의 장점들을 가지고 결함 허용을 제공하고 적은 수의 화일 복제본(2개정도)을 가지고 복제된 화일을 처리하고 갱신하는 것을 제어하는 알고리즘을 제시하였다. HAC는 시스템에 있는 모든 화일에 대해서 시스템 자원을 요구하는 프로세스의 작업 부하와 시스템 특징에 따라 각 화일의 복제와 이주, 그리고 프로세스 이주 시기를 결정하는 알고리즘을 제한 하였다. 여기에서는 화일에 대한 전체 읽기/쓰기 처리 비율에 영향을 주는 변수로 k변수를 사용했는데 실험에 의해 2라는 값을 얻었다. 즉, 총 읽기/쓰기 처리 비율이 2:1 보다 크면 화일 복제를 발생시키고 그렇지 않으면 화일과 프로세스를 이주시켰다[1]. HAC는 읽기/쓰기 처리 수, 화일 크기, 통신 네트워크 이용률을 고려하여 화일의 중복과 이주, 프로세스 이주를 결정하는 분산 알고리즘을 제안 하였다[2]. KIM은 각 화일의 복제본 없이 할당할 노드의 위치들을 결정할때 그 위치들에 대한 각각의 총 통신 비용을 구하여 그 중에서 최소 값을 갖는 한 노드에만 화일을 할당하는 알고리즘을 제안했다. 이때 화일 수, 지역 수, 상수를 각각 F, S, C로 표시하면, 시간 복잡도는 $O(FS^3)$가 된다[15]. 그 밖에 ESWARAN은 질의 비용, 갱신 비용, 저장 비용을 고려하여 최적화하는 문제를 화일 할당 문제(FAP:File Allocation Problem)라 표현하여 이 모델에 통신 지연, 각 노드의 기억 용량, 시스템 신뢰도, 병행성, 가용성등과 같이 여러가지 제한을 두어 연구를 하였고, KOLLIAS는 컴퓨터의 네트워크에서 한 화일의 복제본을 할당하는 문제를 해결하기 위한 규칙들을 제안하였다.

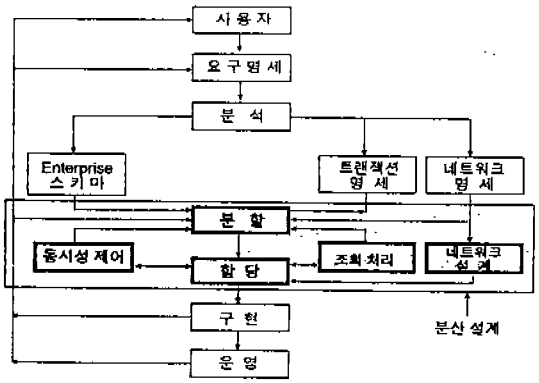
3. 효율적인 분산 트랜잭션 처리 방법

여기에서는 임시 테이블을 사용하여 분산 데이타베이스의 복제와 이주시 갱신 비용과 통신 비용을 최소화하여 운용 비용을 줄여 성능 향상을 높이는 방법을 제시 한다. 즉, 이 방법은 데이타 입력 및 갱신이 많은 분산 트랜잭션 처리에 있어 시스템 버퍼 사용 효율을 높여 디스크 I/O 시간을 줄이는데 그 목적이 있다.

3.1 분산 데이타베이스 설계 방법

분산 데이타베이스를 설계하기 위해서는 중앙 집중형 데이타베이스를 설계할때 고려되는 문제를 그대로 생각하고 같은 기법을 사용하여 해결 할 수 있다. 그러나 분산 데이타베이스 설계시에는 (그림 1)의 설계 방법에서 알 수 있듯이 분할과 할당이라는 중요한 문제를 감안하여야 하며, 본 논문에서 제안하는 임시 테이블을 데이타베이스 할당 설계시 트랜잭션이 많이 발생하는 응용 업무를 고려하여 추가적으로

설계할 것이다.



(그림 1) 분산 데이타베이스 설계 방법
(Fig. 1) Distributed database design method

1) 분할 스키마 설계

분할 스키마 설계는 데이타베이스 분산 설계시에 전역 스키마를 실제 데이타 할당의 논리적 단위가 되는 단편으로 분할하는 것이다[7]. 기본적인 목표는 각 단편들을 가장 자주 참조하는(mostly-referencing) 지역에 배당하여 네트워크 지역간의 전송 비용을 최소화하기 위한 것이다. 그러나 문제점은 관련된 트랜잭션과 속성수가 많을때 이 할당의 선행 작업인 데이타베이스 분할이 쉽지 않다는 것이다. 어떤 데이타베이스 객체가 m개의 속성을 가진다면 가능한 분할의 가짓수는 B(m), 즉, m번째 Bell수가 되는데, 이는 m이 커지면 m^m 단위로 급속히 증가한다[10, 11]. 이를 해결하기 위하여 많은 분할 방법들이 연구 되었는데[4, 7, 10, 12, 14], 이 방법들 또한 관련된 속성과 트랜잭션의 수가 커지면 상당히 복잡해진다. 따라서 데이타 복제 및 이주시 많은 트랜잭션이 발생할때 오버헤드가 증가 하게 된다.

본 논문에서는 위와 같은 문제점을 해결하기 위하여 데이타베이스에 대한 처리를 위한 관련 트랜잭션들을 그룹화 하여 하나의 임시 테이블로 만드는 방법을 제시하였다. 이 방법은 다음 절에서 설명할 데이타 복제 및 이주 방법에서 유용하게 사용될 것이다.

2) 할당 스키마 설계

위에서 생성된 분할들을 각 지역에 저장되는 할당 스키마를 설계하여야 한다. 여기에서 분할을 할당할 때 결정할 사항은 분할이 배정될 위치와 복제본의 존재 여부이다. 복제본이 존재하지 않은 분할을 할당할 경우에는 단순히 해당 분할을 향한 요청이 빈번한 지역을 선택하여 할당하는 것이 바람직하다. 그러나 중복이 발생 할때 문제는 복잡해진다. 왜냐하면 중복도에 따르는 이득과 손실을 계산하기 매우 어려우며, 한 분할에 접근하고자 할때 분할이 배정된 이미지 중 어느 것을 택하기 위한 결정이 모호해질 수 있다. 따라서 화일의 복제본 수, 읽기/쓰기 처리 확률, 화일의 크기, 통신 네트워크의 형태, 화일의 할당 형태, 화일 할당 시기 및 의사 결정 시간 등을 복합적으로 고려해야 한다[16]. 또한 분할과 할당의 두 문제는 서로에게 영향을 미치므로 효율성을 위하여 상호간에 긴밀한 연관성을 두고 설계 하여야 한다. 효율적인 데이타의 분산 설계를 위해서는 다음과 같은 특성이 고려되어져야 한다[8].

첫째, 분산 시스템은 지역 안에서 해결하고자 하는 지역 자치성을 통한 처리의 지역성을 강조하고 있다. 따라서 발생할 수 있는 요청의 근원 지역에 대하여 데이타에 대한 참조가 지역적인지, 전역적인지를 통계적으로 계산하여 가장 바람직한 방법으로 분할하고 할당하여야 한다.

둘째, 분산 시스템은 복제본의 관리를 통하여 전체 시스템의 신뢰도를 향상 시킬 수 있다. 그러므로 적합한 할당의 복제를 통하여 최소한의 처리 부하로 최대한의 신뢰도 향상을 얻는 것이 바람직하다.

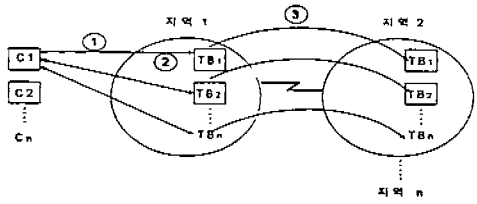
셋째, 여러 지역에 걸쳐 전역적으로 처리해야 할 트랜잭션에 대해서 분산 시스템은 각 지역에 처리 부하를 분산시킴으로써 처리 병행성을 얻는다. 그러나 처리 부하의 분산은 각 지역의 처리 지역성과 상반된 문제가 되므로 데이타 분산 설계시 충분히 고려하여야 한다.

그러나 위의 여러가지 분산 설계 기준들은 바람직한 성향을 제공하고는 있으나 위의 기준들을 모두 적용하여 최적화하기는 실제로 불가능하며 적용할 응용업무와 사용될 분산 시스템의 특성을 고려하여 각 기준의 우선 순위를 정해야 한다. 본 논문에서는 많은 트랜잭션이 발생하는 업무에서 질의 및 갱신 비용과 통신 비용 처리 값을 고려한다.

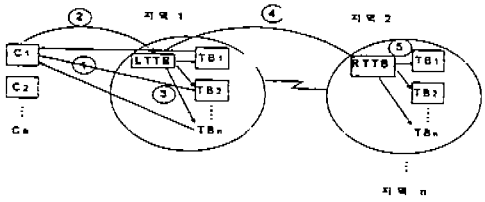
3.2 데이타 복제 방법

데이타 복제 방법은 모든 지역에 동일한 전체 데이타베이스를 유지하는 완전 데이타 복제 방법과 데이타베이스 일부분은 분할된 상태로 존재하고 일부분은 여러 지역에 중복 존재하는 부분 데이타 복제 방법이 있다[13].

본 논문에서는 모든 지역에 동일한 전체 데이타베이스를 유지하는 완전 데이타 복제 방법이다. 그리고 위에서 언급되어진 문제점 즉, 복제본을 갱신해 주어야 하는 오버헤드가 증가하여 응답 시간이 커질때 이를 해결하기 위하여 아래와 같은 (그림 3)의 방법을 적용한다. 이 방법은 데이타베이스에 대한 처리 유형이 비슷한 트랜잭션들을 그룹화하여 하나의 임시 테이블로 만드는 방법으로 이와 같은 임시 테이블은 관련된 테이블에 자동으로 트랜잭션을 완료시키고 초기화된다. 특히 이 방법은 트랜잭션의 발생 및 사용하는 클라이언트의 증가, 그리고 데이타베이스 분산 지역이 많을때 더욱 유용하다. (그림 3)의 처리 방법을 살펴보면, 지역 1의 클라이언트 C_i에서 n개의 테이블을 읽어서 해당 내역을 정정하여 지역 1의 n개의 테이블에 쓰기 작업을 한후 다시 지역 2에 복제하는 과정을 나타낸것이다. 이와 같은 방법은 자기 자신의 지역 1의 테이블과 지역 2 테이블의 복제본을 갱신할 때 버퍼의 사용 효율을 높여 I/O 시간을 줄임으로서 응답 시간을 빠르게 하는데 그 목적이 있다. (그림 2)의 기존의 복제 방법에서는 ①지역1 데이타 읽기, ②지역1 관련 테이블 갱신, ③원격 지역 관련 테이블 갱신의 처리 과정을 나타낸 것으로 처리 단계는 제안된 방법보다는 짧지만 갱신할 트랜잭션이 많을때 ②, ③의 처리 시간과 통신 비용이 많이 소요된다. 즉, 클라이언트 C_1, C_2, C_3,..., C_n으로 증가되고 지역 1, 2, 3,... 지역 n으로 증가될때 더욱 실행 시간이 증가된다. 그러나 (그림 3)의 방법에서는 많은 테이블의 단편화된 트랜잭션들을 하나의 근거리 임시 테이블(Local Temporary Table:LTTB)에 저장하여 LTTB를 지역 2의 원격 임시 테이블(Remote Temporary table:RTTB)에 복제하여 처리하는 과정을 나타내고 있다. 이와 같은 방법은 LTTB에서 RTTB로의 1:1 전송에 따른 통신 비용 절약과 ③, ⑤의 방법을 통하여 자동으로 처리되므로 처리 비용을 절약할 수 있다. 즉, ③, ⑤의 처리-비용은 관련된 트랜잭션 항목들을 하나의 임시 테이블에 저장하여 사용하므로 버퍼의 사용 효율이 높기 때문이다.



(그림 2) 기존 데이타 복제 방법
(Fig. 2) Existed data replication method



(그림 3) 제안 데이타 복제 방법
(Fig. 3) Proposed data replication method

기존 (그림 2)의 방법과 본 논문에서 제시한 (그림 3)의 제안 방법에 대한 처리 비용을 나타내면 아래와 같은 수식으로 정의 할 수 있다.

1) 기존의 복제 방법 처리 비용(Existed Replicated Process Cost : ERPC)

근거리 읽기 처리 비용(Local Read Cost:LRC)은,

$$LRC = \sum_{i=1}^{N} \sum_{k=1}^{T} (C_i \times LRT_k \times TS_k) \quad (1)$$

여기에서 각 기호들을 살펴보면,

T : 전체 테이블의 수,

N : 전체 클라이언트 수,

C_i : 클라이언트 수,

LRT_k : 테이블 T_k에 대하여 근거리 지역에서 Read 처리 수,

TS_k : 테이블 Tk의 크기

를 나타낸 것이다.

위 식을 살펴보면, 같은 근거리 지역에서 데이타 읽기 비용은 사용하는 클라이언트 수, 그리고 읽고자 하는 테이블의 수 및 크기에 따라 처리 시간에 영향을 미치는 것을 알 수 있다.

근거리 지역 쓰기 처리 비용(Local Write Cost: LWC)은,

$$LWC=\sum_{i=1}^{N}\sum_{k=1}^{T}(C_i\times LWT_k\times TS_k) \quad (2)$$

여기에서 기호 LWT_k는 테이블 Tk에 대하여 근거리 지역에서 쓰기 처리 수 이며, 위의 (1)의 근거리 지역 읽기 비용과 같이 사용하는 클라이언트 수, 그리고 쓰고자 하는 테이블 수 및 크기에 따라 처리 시간에 영향을 미치는 것을 알 수 있다. 그러나 여기에서 로킹(locking) 문제를 살펴볼때, 배타적 로크(exclusive lock)를 걸면 데이타 항목을 어떤 목적으로 접근할 수는 있지만 다른 트랜잭션은 그 데이타를 읽거나 변경할 수 없다. 공용 로크(shared lock)를 걸면 데이타 항목을 읽을 수는 있지만 변경할 수는 없다. 즉, 다른 트랜잭션은 이 항목을 갱신하지 않는 한 읽어 갈 수 있다. 본 논문에서는 공용 로크 방법을 적용하였다. 따라서 하나의 클라이언트에서 갱신 시간이 지연되면 전체 시스템의 반환 시간이 증가하게 된다.

원격 지역 쓰기 처리 비용(RWC)은,

$$RWC=\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}(C_i\times RWT_k\times TS_k\times ATK_r\times LRC_{lr}) \quad (3)$$

여기서에서,

R : 테이블 T_k가 원격 지역 r에 할당된 전체 경우의 수,

RWT_k: 테이블 Tk에 대하여 원격 지역에서 Write 처리 수,

ATK_r : 테이블 T_k가 원격 지역 r에 할당된 경우의 수,

LRC_{lr} : Local 지역 l과 원격 지역 r간의 통신 비용

을 나타낸다.

따라서 (1), (2), (3)을 종합적으로 나타내면 다음과 같은 식으로 정의된다.

$$ERPC=[LRC+LWC+RWC]=$$

$$[\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}\{(C_i\times LRT_k\times TS_k)+(C_i\times LWT_k\times TS_k)+(C_i\times RWT_k\times TS_k\times ATK_r\times LRC_{lr})\}] \quad (4)$$

2) 제안 복제 방법 처리 비용(Proposed Replicated Process Cost : PRPC)

근거리 지역 읽기 처리 비용(LRC)은,

$$LRC=\sum_{i=1}^{N}\sum_{k=1}^{T}(C_i\times LRT_k\times TS_k) \quad (5)$$

여기에서 각 기호들을 살펴보면,

T : 전체 테이블의 수,

N : 전체 클라이언트 수,

C_i : 클라이언트 수,

LRT_k : 테이블 T_k에 대하여 근거리 지역에서 읽기 처리 수,

TS_k : 테이블 T_k의 크기

를 나타낸다.

위 식은 (1)의 기존 방법과 동일 하다.

근거리 지역 쓰기 처리 비용(LWC)은,

$$LWC=[\sum_{i=1}^{N}\sum_{k=1}^{T}\{(C_i\times LTTB\times TS_k)+(LTTB\times LWT_k)\}] \quad (6)$$

여기에서 기호 LTTB은 하나의 근거리 지역 임시 테이블을 나타낸다. 이 방법은 처리 과정은 증가 하지만, LWT_k 즉, 테이블 T_k에 대하여 근거리 지역에서 쓰기 처리 수가 하나의 임시 테이블(LTTB)이 되므로 기존의 방법보다 빠르다는 것을 알 수 있다. 물론, 임시 테이블에서 테이블 T_k에 쓰기 단계는 남아 있지만 이러한 과정은 버퍼의 사용 효율이 높아 그 비용이 미세하다. 따라서 로킹 문제에 있어서도 시간이 감소되므로 전체 시스템의 반환 시간이 감소하게 된다.

원격 지역 쓰기 처리 비용(RWC)은,

$$RWC=[\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}\{(C_i\times LTTB\times TS_k\times RTTB_r\times LRC_{lr})+(RTTB_r\times RWT_k)\}] \quad (7)$$

여기에서 RTTBr는 LTTB가 할당된 원격 지역의

임시 테이블이며, 그 밖의 기호들은 앞에서 설명한 내용과 같다.

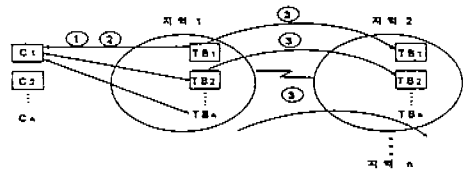
따라서 (5), (6), (7)을 종합적으로 나타내면 다음과 같은 식으로 정의된다.

$$PRPC=[LRC+LWC+RWC]$$
$$=[\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}\{(C_i\times LRT_k\times TS_k)+(C_i\times LTTB\times TS_k)$$
$$+(LTTB\times LWT_k)+(C_i\times LTTB\times TS_k\times RTTB_r$$
$$\times LRC_n)+(RTTB_r\times RWT_k)\}] \quad (8)$$

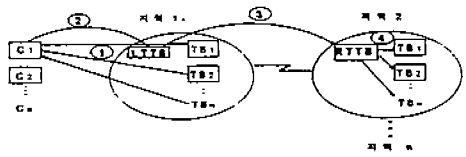
따라서 기존의 방법과 비교하여 볼때 읽기 처리는 같지만 쓰기 처리에서는 차이점이 있음을 알 수 있다. 본 논문에서 제안한 복제 방법은 관련된 트랜잭션들을 하나의 임시 테이블에 저장하여 버퍼의 사용 효율을 높여 해당 지역간의 통신 비용과 쓰기 비용을 최소화 한다.

3.3 데이타 이주 방법

데이타 이주 방법은 화일을 복제하지 않고 많이 처리하는 컴퓨터로 이주하여 처리하게 하는 방법으로 될 수 있으면 지역 참조가 많이 일어나게 하는 것이다. 분산 처리 시스템에서 화일 복제 방식보다는 화일 이주 방식이 장점이 많다고 할 수 있다[2, 8, 13]. 그러나 화일 이주 문제에서는 화일 이주를 위한 적절한 목표 컴퓨터의 선택이나 화일 이주 시점을 결정하는 것은 불가능하다. 오히려 잘못된 결정을 하면 화일만 컴퓨터 사이를 오가고 실제 실행 시간의 단축에 나쁜 영향을 주게 된다. 또한 화일 전체 이주가 아닌 수직 분할 및 수평 분할 이주시 많은 트랜잭션이 일어날때 발생하는 통신 비용으로 최소의 응답 시간을 보장 받을 수 없다. (그림 4)는 기존의 데이타 이주 방법이다. 본 논문에서는 위에서 언급되어진 문제점을 해결 하기 위하여 아래와 같은 (그림 5)의 방법을 적용한다. 이 방법은 데이타 복제 방법과 같이 데이타베이스에 대한 처리 유형이 비슷한 트랜잭션들를 그룹화하여 하나의 임시 테이블로 만드며, 이주할 트랜잭션의 발생 화일이 증가할 때 유용하다. 처리 방법을 살펴보면, 지역 1의 n개의 테이블을 지역 2에 이주하는 과정을 나타낸것이다. 이와 같은 방법은 지역 1 테이블의 데이타를 지역 2의 테이블에 이주할때 오버헤드를 줄여 응답 시간을 줄이는데 그 목적이 있다. (그림 4)의 기존의 이주 방법에서는 처리 단계는 짧지만 이주할 트랜잭션이 많을때 ③의 처리 시간이 많이 소요된다. 즉, 이주할 지역 TB_1, TB_2, TB_3,..., TB_n으로 증가하고 사용하는 클라이언트 수가 많을때 더욱 실행 시간이 증가된다. 그러나 (그림 5)의 방법에서는 많은 테이블의 단편화된 트랜잭션들을 하나의 임시 테이블에 저장하여 ②, ③, ④의 방법을 통하여 처리되므로 처리 비용을 절약할 수 있다. 즉, 지역 1에서 하나의 임시 테이블 1(LTTB)를 지역 2의 임시 테이블 2(RTTB)로 이주 하는데 대한 통신 비용이 절약되며, 또한 ④의 서버 내에서의 처리 비용은 아주 미세하기 때문이다.



(그림 4) 기존 데이타 이주 방법
(Fig. 4) Existed data migration method



(그림 5) 제안 데이타 이주 방법
(Fig. 5) Proposed data migration method

본 논문에서 제시한 (그림 4)의 방법과 (그림 5)의 제안 방법에 대한 처리 비용을 나타내면 다음과 같이 수식으로 정의 할 수 있다.

1) 기존의 이주 방법 처리 비용(Existed Migration Process Cost : EMPC)

근거리 읽기 처리 비용(LRC)은 (1), (5)와 같다.

$$LRC=\sum_{i=1}^{N}\sum_{k=1}^{T}(C_i\times LRT_k\times TS_k) \quad (9)$$

따라서 각 기호 및 내용은 생략하고자 한다.

그리고 근거리 지역 쓰기 처리 비용(LWC)은 이주 방법이므로 없으며, 원격 지역 쓰기 처리 비용(RWC) 역시 (3)과 같다. 그러나 여기에서 ATK_r은 전체 화일 이주에서는 필요 없으나, 본 논문에서 적용한 수평, 수직 분할 이주시에는 필요하다.

$$RWC=\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}(C_i\times RWT_k\times TS_k\times ATK_r\times LRC_{lr}) \quad (10)$$

따라서 (9), (10)을 종합적으로 나타내면 다음과 같은 식으로 정의된다.

$$EMPC=[LRC+RWC]=[\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}\{(C_i\times LRT_k\times TS_k)$$
$$+(C_i\times RWT_k\times TS_k\times ATK_r\times RC_{lr})\}] \quad (11)$$

2) 제안 이주 방법 처리 비용(Proposed Migration Process Cost : PMPC)

근거리 지역 읽기 처리 비용(LRC)은 (1), (5), (9)의 방법과 같다.

$$LRC=\sum_{i=1}^{N}\sum_{k=1}^{T}(C_i\times LRT_k\times TS_k) \quad (12)$$

근거리 지역 쓰기 처리 비용(LWC)은 이주 방법이므로 없으며, 원격 지역 쓰기 처리 비용 (RWC)은,

$$RWC=[\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}\{(C_i\times LTTB\times TS_k\times RTTB_r\times LRC_{lr})$$
$$+(RTTB_r\times RWT_k)\}] \quad (13)$$

여기에서 $RTTB_r$는 복제 방법과 같이 원격 지역의 임시 테이블이며, 그 밖의 기호들은 앞에서 설명한 내용과 같다.

따라서 (12), (13)을 종합적으로 나타내면 다음과 같은 식으로 정의된다.

$$PMPC=[LRC+RWC]=[\sum_{i=1}^{N}\sum_{k=1}^{T}\sum_{r=1}^{R}\{(C_i\times LRT_k\times TS_k)$$
$$+(C_i\times LTTB\times TS_k)+(C_i\times LTTB\times TS_k\times RTTB_r\times LRC_{lr})$$
$$+(RTTB_r\times RWT_k)\}] \quad (14)$$

따라서 기존의 방법과 비교하여 볼때 읽기 처리는 같지만 쓰기 처리에서는 차이점이 있음을 알 수 있다. 본 논문에서 제안한 이주 방법은 복제 방법과 같이 지역간의 통신 비용과 쓰기 비용을 최소화 하는데 그 목적이 있다.

3.3 성능 평가 방법

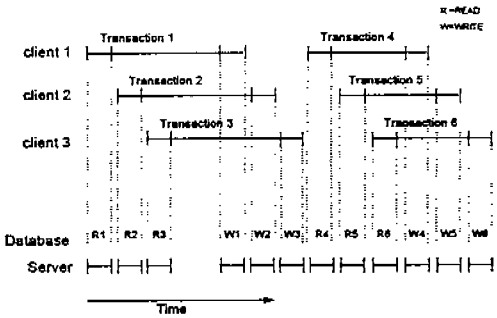
본 논문에서는 데이타 복제 및 이주시 임시 테이블을 사용한다. 따라서 버퍼의 사용 효율을 높여 디스크 I/O 시간을 단축하여 데이타의 병행 속도를 증가시키는데 그 목적이 있다. 그리고 성능 평가 방법에서 로크의 단위를 테이블 단위로 하고, 유형은 공용 로크 방법을 적용하여 클라이언트/서버 시스템 상에서 병행 처리 및 로킹 문제를 기존의 방법과 본 논문에서 제시한 방법을 비교 제시하고자 한다.

3.3.1 클라이언트/서버 시스템상에서 병행처리

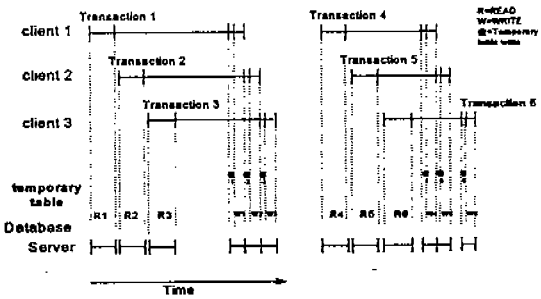
본 논문에서 제안되는 시스템 환경은 응용 프로그램을 처리하는 클라이언트와 데이타베이스를 처리하는 서버에서 최대한의 병행도를 이루기 위해 임시 테이블(temporary table)을 사용하는 방법을 제안하였다. 기존의 트랜잭션 처리 방법은 (그림 6)과 같이 1대의 서버와 3대의 클라이언트를 가지는 트랜잭션 과정을 나타내고 있다. 클라이언트는 서버가 데이타를 처리해주길 기다려야 하지만 클라이언트들 사이는 데이타베이스 활동을 제외한 모든 활동은 병행처리되는 것을 알 수 있다. 결국 클라이언트가 여러대일 경우에도 서버가 하나밖에 없으므로 서버상에서는 직렬화되므로 로킹이라는 문제점이 제기된다. 즉 서버에서는 트랙잭션을 읽고 수정된 데이타를 데이타베이스 안에 있는 모든 테이블에게 수정 하게 함으로서 읽고 쓰는 입/출력 시간이 많이 소요하게 된다. 그러나 트랜잭션된 데이타들을 데이타베이스내의 하나의 임시 테이블에 저장하여 임시 테이블에서 데이타베이스안에 있는 관련 테이블에 수정하게 함으로써 효율적인 병행처리가 이루어져 로킹 문제를 해결할 수 있다.

(그림 7)과 같이 클라이언트 1에서 트랜잭션 1을 임시 테이블에 저장한후 서버가 작업을 하며, 클라이언트 2에서 트랜잭션 2작업을, 클라이언트 3에서 트랜잭션 3작업을 임시 테이블에 저장한 후 작업을 하게 되므로 클라이언트/서버 시스템상에서 임시 테이블

사용은 효율적인 트랜잭션 처리가 가능하다.



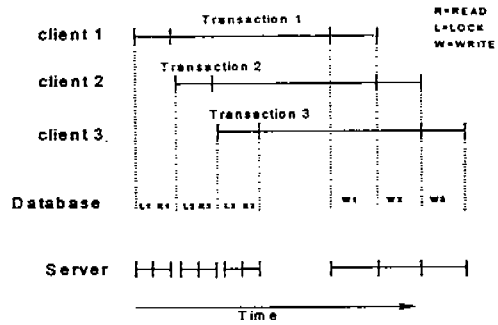
(그림 6) 기존 시스템상의 병행 처리
(Fig. 6) Concurrent process on existed system



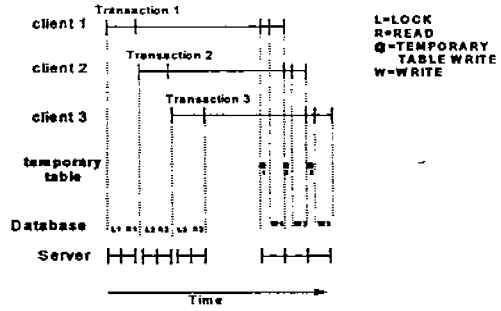
(그림 7) 제안 시스템상의 병행 처리
(Fig. 7) Concurrent process on proposed system

3.3.2 클라이언트/서버 시스템상의 로킹

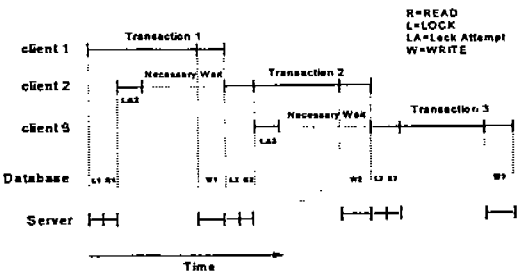
기존의 여러 클라이언트에서 동일한 테이블을 처리할 수 있기 때문에 갱신 분실과 모순적인 판독 문제를 방지하는 일이 필요하다. 이를 해결하기 위한 방법으로는 첫째, 비관적인 로킹(pessimistic locking)으로 충돌을 예상하고 미리 로크를 걸기 때문에 데이타베이스 관리 시스템은 시작 트랜잭션 이후에 실행되는 모든 명령에 대해 암시 로크를 건다[5]. 이러한 로크는 완료(commit)나 복귀(rollback)명령이 실행 될 때 까지 유지된다. (그림 8), (그림 9)는 두개의 트랙잭션 처리의 요구 시간을 보이고 있다. 서로 트랙잭션이 상이하므로 데이타 충돌이 발생하지 않는다. 서버는 클라이언트 1이 데이타 요청에 대한 처리 하고있는 시간동안 클라이언트 2는 기다린다. 그러나 그 이후의 두 클라이언트가 트랜잭션이 끝날때까지 병행처리된다. 여기에서 제안된 방법은 트랜잭션의 결과를 임시 테이블에 저장하여 서버가 지역적으로 작업을 하게됨으로서 서버와의 병행 속도를 증가 시킬 수 있다.



(그림 8) 기존 시스템상의 로킹(서로 상이한 데이타 처리)
(Fig. 8) Locking on existed system(different data process)

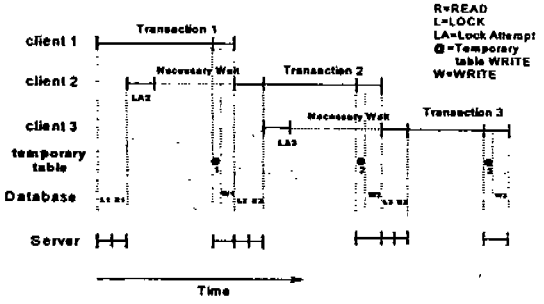


(그림 9) 제안 시스템상의 로킹(서로 상이한 데이타 처리)
(Fig. 9) Locking on proposed system (different data process)



(그림 10) 기존 시스템상의 로킹(서로 동일한 데이타 처리)
(Fig. 10) Locking on existed system(homogeneous data process)

(그림 10)은 동일 테이블을 처리하려는 두 트랜잭션을 보이고 있다. 이때 클라이언트 2는 클라이언트 1이 끝날때까지 기다려야한다. 이러한 트랜잭션 처리는 오랜 시간 동안 로크가 걸린다. 만약 클라이언트 1의 입력 데이타나 수정 데이타가 많다면 클라이언트 2에서 클라이언트 1의 데이타를 판독 및 변경 하고 대체 할 수 도 있다. 이러한 문제를 해결하기 위해서 행 단위로 로킹을 지원하지 못하면 더욱 악화 된다. 어떤 데이타베이스 관리 시스템에서는 페이지 단위 또는 테이블 단위로 로킹을 지원 하므로 임시 로크는 사람이 감지 할수 있을 정도로 불 필요한 지연을 야기할 수 있다. (그림 11)과 같이 이러한 방법에서도 제안된 클라이언트/서버 시스템상에서 임시 테이블 사용 기법은 로킹시간이 짧아지므로 갱신 분실과 모순적인 판독 문제를 방지 하면서 클라이언트 컴퓨터 상에서 최대한의 병행성을 유지할 수 있다.



(그림 11) 제안 시스템상의 로킹(서로 동일한 데이타 처리)
(Fig. 11) Locking on proposed system(homogeneous data process)

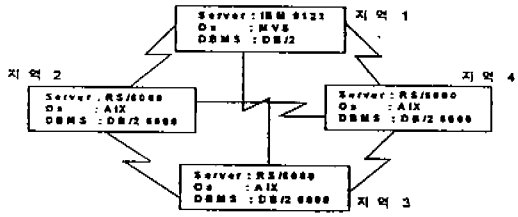
4. 실 시스템 적용 예 및 성능 평가 분석

본장에서는 실 시스템을 적용한 클라이언트/서버 분산 트랜잭션 환경과 실예를 보이고 이를 통하여 제안한 데이타 복제 방법과 이주 방법에 대하여 성능 평가 분석을 하고자 한다.

4.1 클라이언트/서버 분산 트랜잭션 환경

분산 트랜잭션 처리 모델에서는 (그림 12)과 같이 IBM 9121시스템과 RS/6000 3개로, 서버가 4개로 구성된 분산 시스템 환경이며, 본 논문에서는 다음과 같은 가정을 기본 전제로 한다.

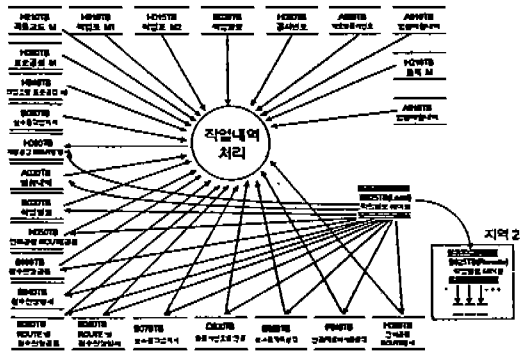
1) 시스템은 클라이언트/서버 구조로 각 노드들은 LAN으로 구성되어 있지 않다.
2) 통신망은 전용 회선을 이용하며 이들의 전송 속도 및 거리에 따른 통신 비용은 일정하다.
3) 네트워크상의 용량은 충분하여 병목현상(bottlenck) 및 교착상태는 발생하지 않는다.
4) 각 지역의 서버는 결함이 없다.
5) 로크의 단위는 테이블 단위로 이루어지며, 로크의 유형은 공용 로크(shared lock)이다.
6) 트랜잭션 단위(unit)는 입력 및 갱신시 각 테이블에 처리되는 레코드의 건수를 의미한다.
7) 하나의 트랜잭션당 데이타량은 약 57byte이다.
8) 지역간의 통신 모뎀 처리 속도는 9600bps이다.



(그림 12) 분산 트랜잭션 환경
(Fig. 12) Distributed transaction environment

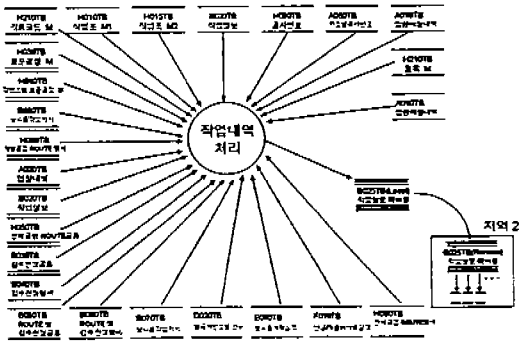
4.2 실 시스템 적용 예

본 논문의 실 시스템 적용은 차량 정비 시스템의 공정 관리 업무로서, 각 공장에서 작업조별로 업무 내역을 처리하는 작업 보고서 내역에 관련된다.



(그림 13) 제안 복제 방법 데이타 흐름 다이어그램
(Fig. 13) Proposed replication method data flow diagram

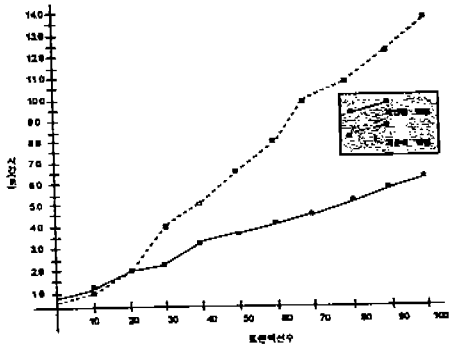
(그림 13), (그림 14)는 각각 제안된 복제 방법과, 이주 방법으로 임시 테이블을 사용하여 트랜잭션 작업을 수행하고 있는 데이타 흐름 다이어그램을 보여주고 있다.



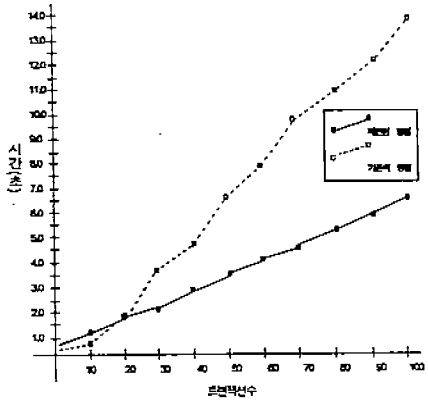
(그림 14) 제안 이주 방법 데이타 흐름 다이어그램
(Fig. 14) Proposed migration method data flow diagram

4.3 성능 평가 분석

트랜잭션은 로크(lock)를 요청하고 반납하는 기본 수행 단위라 할 수 있으며 트랜잭션의 수행 시간은 로크 요청 갯수에 직접적인 영향을 받는다. 따라서 자원을 점유하는 시간은 트랜잭션의 예상 수행 시간에 영향을 주게 되며, 성능 평가에서 자원 점유 시간이 크다는 의미는 트랜잭션 수행 시간이 비교적 길다는 의미와 같다[8, 9, 13].



(그림 15) 데이타 복제시 성능 평가
(Fig. 15) Performance evaluation at data replication



(그림 16) 데이타 이주시 성능 평가
(Fig. 16) Performance evaluation at data migration

본 논문에서는 분산 트랜잭션 처리에서 데이타 복제와 이주시에 발생되는 처리 시간을 측정 하였다. (그림 15), (그림 16) 평가에서 알 수 있듯이 발생하는 트랜잭션의 수가 20개 이하일 경우에는 기존의 방법과 차이점이 없으나 그 이상의 트랜잭션이 발생할 때는 제안한 방법이 우수하다는 것을 알 수 있다. 이 방법은 데이타베이스 할당 설계시에 응용 업무를 고려하여 적용한다면 데이타 통신 비용을 감소시키며, 또한 처리를 위해 기다리는 디스크 I/O 시간이 감소되므로 시스템 전체 작업들의 평균 반환 시간이 짧아진다.

5. 결 론

본 논문에서는 분산 환경에서 관계형 데이타베이스의 효율적인 트랜잭션 처리 방법을 통해서 화일 처리 비용을 최소로 하는 방법을 제시하였다. 제안되는 시스템은 클라이언트/서버 환경으로 서버에서 최대한의 병행성을 이루기 위해 임시 테이블을 사용하는 방법이다. 클라이언트 응용 프로그램에서 트랜잭션 처리 결과를 임시 테이블에 저장하여 서버내에서 자동으로 해당 테이블에 복제 또는 이주되어 서버는 지역적인 작업을 하게 됨으로써 병행 속도가 빨라진다. 그러나 트랜잭션의 수가 작을때는 기존의 방법과는 병행 처리에는 속도의 차이는 거의 없지만 트랜잭션의 수가 많을 때에는 병행 처리에서의 속도는 큰 차이가 있다는 것을 알 수 있었다. 결국 이러한 문제는 클라이언트가 여러 대 일 경우에는 더욱 증가되어 읽

고 수정된 데이타를 데이타베이스의 관련된 모든 데이블에 갱신하게 함으로서 읽고 쓰는 시간이 많이 소요하게 된다.

따라서 본 논문에서 제안하는 복제 및 이주방법은 관련된 테이블의 항목들을 그룹화하여 임시 테이블에 저장 일괄 처리하므로서 입력 및 갱신이 많은 분산 트랜잭션 처리에 있어 시스템 버퍼 사용 효율을 높여준다. 결국 디스크 I/O 처리 시간과 지역간 통신 비용을 줄임으로써 데이타의 병행 속도를 빠르게 한다. 앞으로의 연구 과제로는 OMG의 CORBA 규정을 만족하는 효율적인 객체 지향 분산 병행 시스템 설계 방법에 있다.

참 고 문 헌

[1] A. Hac, "File Migration and Process Migration in a Local Area Network Environment", Proc. of INFOCOM 86, pp. 488-495, 1986.

[2] A. Hac, "A Distributed Algorithm for Performance Improvement Through File Replication, File Migration, and Process Migration", IEEE Trans. on SE, vol. 15, no. 11, pp. 1459-1470, 1989.

[3] B. S. Bacarisse & S. B. baydere, "A Low Cost File Replication Algorithm", IEEE SE, pp. 191-196, 1989.

[4] B.T. WILSON and S.B. NAVATHE, "An Analytical Framework for the Redesign of Distributed Databases", Proceedings 6th Advanced Database Symposium, Aug. 1986.

[5] Blaustein, B.T., and Kaufman, C.W., "Updating Replicated Data during Communication Failures", Proc. 11th International Conf. on Very Large Database, Aug. pp. 49-58, 1985.

[6] C.G. Ming & C.S. Raghavendra, "An Algorithm for Optimal File Allocation in Distribution Computing Systems", Adv. in Dist. Sys. Reliability, IEEE Comp. Society Press, pp. 246-250, 1990.

[7] D. SACCA and G. WIEDERHOLD, "Database Partitioning in a Cluster of Processors", ACM TR. on Database Systems, Vol. 10, No. 1, Mar. 1985.

[8] G. Coulouris, J. Dollimore and T. Kindberg, "Distributed Systems Concepts and Design", Second Edition, Addison-Wesley, pp. 30-32, 311-349, 422-444, 1994.

[9] H. Garcia-morina & R. K. Abbort, "Reliable Distributed Database Management", Proc. of the IEEE, Vol. 75, No. 5, May. pp. 601-620, 1987.

[10] K.B. Irani and N.G. Khabbaz, "A methodology for the design of communication network and the distribution of data in distributed super computer systems", IEEE Trans. Comp. Vol. C-31, May. pp. 419-434 1982.

[11] Rothnie J. B., and Goodman N., "A Survey of Research and Development in Distributed Database Management", Proc. 3rd Conf. VLDB 1977.

[12] S.K. CHANG and W.H. CHANG, "A Methodology for Structured Database Decomposition", IEEE TR. on Software Engineering, Vol. SE-6, No. 2, Mar. 1980.

[13] S.Ceri and G. Pelagatti, "Distributed Databases: Principles and System", McGraw-Hill, Inc., 1984.

[14] S. CERI, S. NAVATHE, and G, WIEDERHOLD, "Distribution Design of Logical Database Schemas", IEEE TR. on Software Engineering, Vol. SE-9, No. 4, Jul. 1983.

[15] S. Kim & S. C. Moon, "Nonredundant Allocation of Files in Distributed Systems", TENCON 87, pp. 287-291, 1987.

[16] 김영시, "분산 시스템에서의 동적 화일 할당 알고리즘", 중앙대학교 박사 논문, 6월. pp. 2-4, 1991.

허 계 범

1989년 경기대학교 응용통계학과(학사)

1993년 광운대학교 전산대학원 전자계산학과(이학석사)

1988년~1995년 (주) 경인양행 전산부 과장

1995년~현재 광운대학교 전자계산학과 박사과정

관심분야: 객체 지향 소프트웨어 공학, 객체 지향 분산 컴퓨팅

이 종 섭

1990년 호서대학교 전자계산학과(학사)
1993년 광운대학교 전산계산학과(이학석사)
1994년~현재 광운대학교 전자계산학과 박사과정
1997년~현재 문경 전문대학 사무자동학과 전임강사
관심분야:객체 지향 프로그래밍 언어, 객체 지향 분산 컴퓨팅

정 계 동

1985년 광운대학교 전자계산학과(학사)
1992년 광운대학교 산업대학원 전산계산학과(이학석사)
1992년~현재 광운대학교 전자계산학과 박사과정
관심분야:객체 지향 프로그래밍 언어, 객체 지향 분산 컴퓨팅

최 영 근

1980년 서울대학교 수학교육과(학사)
1982년 서울대학교 계산통계학과(이학석사)
1989년 서울대학교 계산통계학과(이학박사)
1996년~1997년 미시간 주립 대학교 객원교수
1983년~현재 광운대학교 전자계산학과 교수
관심분야:프로그래밍 언어, 병렬 컴퓨터, 객체 지향 분산 컴퓨팅