트랜잭션의 연산정보를 이용한 클라이언트 캐시 일관성 유지 기법

(A Client Cache Consistency Method using Information of Transaction Operations)

유 제 혁 † 조 성 호 † 정 일 영 † 황 종 선 ††
(JeHyok Ryu) (SungHo Cho)(Il Young Chung)(Chong-Sun Hwang)

요 약 클라이언트-서버(Client-Server) 데이타베이스 환경에서 병목현상의 해결책으로 클라이언트 캐시(client cache)를 사용할 수 있다. 여러 클라이언트들이 같은 데이타베이스의 일부를 캐시한 복사본을 사용할 때, 다른 클라이언트들과 서버간에 일관성 유지를 보장해주어야 한다. 캐시의 일관성을 유지하기 위한 여러 가지 알고리즘들이 제안되어왔다. 그 중에서 O2PL(Optimistic Two Phase Locking) 기법은 동시성을 증가시킬 수 있는 주요 장점이 있다. 그러나 O2PL 알고리즘은 트랜잭션이 완료(commit)연산을 요구하는 시점에서 접근한 캐시데이타의 일관성을 위한 작업이 시작되므로 최종적인 완료까지는 지연이 발생하게된다. 이때 다른 클라이언트에서 잠금 충돌(lock conflict)에 의해 전역 교착상태(global deadlock)가 발생한다면 블락(block)되어 지연되는 시간은 더욱 증가하게된다.

본 논문에서는 향상된 O2PL기법으로 PN(Preemption by Notification)-O2PL 기법을 제안한다. 이 접근에서는 O2PL의 특성으로 얻을 수 있는 연산정보를 관련된 다른 트랜잭션에게 통지(notification)로 사용하고 제시한 조건에 해당할 경우 상대 자원을 바로 사용할 수 있게 한다. 이 조건은 대부분의 전역 교착상태에 해당하는 두 트랜잭션간의 교착상태를 조기에 신속히 감지할 수 있게 하여 트랜잭션 완료의 지연을 감소시키게 한다. PN-O2PL 알고리즘이 기존의 O2PL기법보다 빠른 응답시간을 얻을 수 있음을 모의실험을 통하여 보인다.

Abstract The client caching in client-server database environment can be used as one of methods for solving of bottleneck problem. When clients use their own cached data, the consistency must be ensured between other clients and the server. Various methods have been proposed for the consistency of client cache. The O2PL (Optimistic Two Phase Locking) method among them provides a major advantage of increasing concurrency. However, the O2PL algorithm makes some delay from the commit request to the commit of transactions, since the cache consistency action begins at the commit request time. At this time, if the global deadlocks by the lock conflict occur, the duration of the delay by the blocking increases more and more.

In this paper, we propose an improved O2PL called PN(Preemption by Notifiaction)-O2PL. In this approach, the operation informations achieved by the O2PL feature are used as notifications to another related transactions at their clients, and the condition offered in this paper is compared to utilize counterpart's resources. This conditions detect some deadlocks between two transactions early - these deadlocks correspond to most of global deadlocks, so it reduces the delay in transactions committing. In the simulation studies, we show that PN-O2PL algorithm achieves faster response time than that of the existing O2PL algorithm.

† 비 회 원 : 고려대학교 컴퓨터학과
jhryu@disys.korea.ac.kr
zoch@disys.korea.ac.kr
jiy@disys.korea.ac.kr
†† 종신회원 : 고려대학교 컴퓨터학과 교수
hwang@disys korea.ac.kr
논문접수 : 1999년 10월 26일
심사완료 : 2000년 8월 23일

1. 서 론

클라이언트-서버(Client-Server) 구조는 분산 컴퓨팅과 여러 응용프로그램의 수행을 위한 영역으로 널리 사용되고 채택되어왔다. 또한 최근 고성능 마이크로프로세

서와 메모리, 대용량 디스크 등의 하드웨어 개발과 값싼 제공은 개인 데스크탑 시스템에서의 컴퓨팅 능력을 향상시킬 수 있게 하였다. 이러한 개발과 동시에 진보된 네트워크 하드웨어와 소프트웨어는 데스크탑 시스템들이 서로간에 효과적이고 신속한 연결을 가능하게 하였고, 서버와 함께 거대한 구성으로 운영될 수 있게 하였다. 이러한 결과로 클라이언트-서버 구조는 더욱 활용범위가 늘어나고 있다.

그런데 페이지에 기반한 클라이언트-서버 DBMS (Database Management System)에서 트랜잭션을 처리할 때 여러 클라이언트들이 하나의 서버에 있는 페이지를 접근하기 위한 요청으로 발생하는 병목현상은 원활한 작업처리에 지연을 초래할 수 있다. 이러한 문제점을 해결하려는 방법으로 각 클라이언트들이 데이타베이스의 일부를 캐싱(caching)하는 클라이언트 캐시(client cache)를 사용할 수 있다. 이와 같은 클라이언트 자원을 사용하여 작업을 효과적으로 처리할 수 있다. 이러한 시도는 DBMS에서 클라이언트-서버 환경의 정확성(correctness)과 가용성(availability)의 제약들이 함께 고려되어야한다.

클라이언트-서버 시스템에서 클라이언트 캐싱은 서버로부터 획득된 데이타베이스 페이지의 복사본을 클라이언트의 지역에 보유하는 클라이언트의 능력이라 할 수 있다. 이러한 문맥에서 클라이언트 캐싱은 다음과 같이 표현할 수 있다[5].

클라이언트 캐싱(client caching) =
동적 복사(Dynamic Replication)
제 2등급 소유자격(Second-Class Ownership)

동적 복사는 클라이언트들의 요구에 따라 수행시간(runtime)에 페이지의 복사본을 생성하고 제거함을 의미한다. 따라서 어떤 주어진 시간에 시스템에 존재하는 복사본의 내역은 클라이언트들의 최근 행동에 의해 주도된다.

제 2등급 소유자격은 클라이언트에 캐시된 페이지의 복사본이 서버에서 유지되는 실제 데이타 페이지와 동등하게 고려되지는 않음을 말한다. 서버에서는 데이타베이스에 대한 트랜잭션 수행의 지속성(durability)을 보장할 필요가 있는 페이지, 로그 등의 자료를 지역에 갖고 있다. 그래서 클라이언트에서 캐시된 페이지는 완료(commit)된 갱신의 손실 없이 언제라도 제거될 수 있다. 이것은 페이지 서버 시스템에서 데이타의 가용성에 관하여 중요한 사실이다. 왜냐하면 서버는 어느 때라도 어떤 클라이언트가 현재 수행중인 트랜잭션이 단독으로 철회됨으로써 기능이 중단된(crashed) 클라이언트를 허용할 수 있다. 따라서 서버는 함께 작업하는 중이 아닌 클라이언트나 기능이 중단된 클라이언트에 의해 지연되거나 중단될 필요가 없다. 결국 페이지 서버 DBMS에서 제2등급 복사본은 일관성있는 데이타의 가용성을 촉진하는데 사용됨을 의미한다.

이 두 가지 성질의 클라이언트 캐싱은 클라이언트 자원의 이용율과 데이타의 정확성, 가용성간에 절충안이다. 클라이언트 캐싱은 서버로부터 데이타를 접근하는 것보다 더 빠른 시간에 접근할 수 있도록 허용하지만 클라이언트에게 페이지의 소유자격까지 부여하는 것이 아니고, 서버가 실제 데이타의 소유자이므로 어떠한 트랜잭션 수행의 정확성을 보장하기 위한 최종의 책임은 서버가 갖게된다. 서버가 그러한 책임을 갖는다는 것은 클라이언트의 트랜잭션 수행에 관련하여 현재 향상된 클라이언트 워크스테이션의 컴퓨팅 능력을 사용하면서도 서버에게 부담이 주어질 수밖에 없다. 이러한 서버의 부담을 덜기 위해 클라이언트의 캐시된 데이타도 항상 유효성을 갖도록 하는 캐시 일관성 유지 기법을 사용할 수 있다. 이렇게 클라이언트 캐싱 기법에서 지역에 캐시된 페이지가 최신의 상태임을 보장하고 트랜잭션의 연산을 지역에서 모두 수행한 후 완료 단계에서 서버와 상호작용을 하는 낙관적인(optimistic) 기법은 이러한 클라이언트 워크스테이션의 능력을 최대한 활용할 수 있다고 할 수 있다.

2. 관련연구 및 연구동기

2.1 클라이언트 캐시 일관성 유지 기법

클라이언트 캐싱은 서버 DBMS에서 성능과 가용성을 높이는 효과적인 기술이다. 이것은 서버로 보내지는 데이타 요구를 줄이기 위하여 DBMS가 클라이언트 시스템의 메모리를 사용할 수 있게 한다. 이러한 케싱은 여러 클라이언트 사이트에서 허용되기 때문에 모든 클라이언트 사이트에서 사용되는 데이타베이스의 일관성 있는 값을 갖도록 보장해주어야 한다. 페이지에 기반한 클라이언트-서버 DBMS에 의한 모델의 정확성은 트랜잭션의 직렬화 가능한(serializable) 수행이 요구되는 트랜잭션의 성질에 기반한다. 그리고 캐시된 페이지 복사본의 존재는 시스템의 사용자에게 투명(transparent)해야 한다. 그래서 중복된(replicated) 데이타베이스에서 트랜잭션들의 실행은 비중복(non-replicated) 데이타베이스에서와 같은 결과를 얻어야한다. 따라서 캐시 일관성을 유지하는 문제는 동시성 제어 및 시스템에서 사용된 트랜잭션 관리에 관계한다. 이러한 클라이언트 캐시 일관

성 유지 기법은 클라이언트의 트랜잭션이 접근하는 캐시된 페이지 복사본의 유효성(validity)에 따라 크게 탐지기반(detection based) 프로토콜 및 회피기반(avoidance based) 프로토콜로 분류될 수 있다[5, 6, 7, 10]. 탐지기반 기법과 회피기반 기법의 구별은 어떠한 방식으로 최신의 값이 아닌(stale) 데이타에 접근하지 못하도록 할 것인가에 놓여있다.

탐지기반 프로토콜에서는 클라이언트 캐시에 있는 최신의 값이 아닌 데이타 복사본의 존재를 허용한다. 따라서 최종 완료를 하기 전에 접근한 캐시된 페이지의 유효성 검사를 해야한다. 서버는 클라이언트가 유효성 검사를 할 수 있는 최신의 정보를 유지하는 책임이 있다. 이러한 유효성 검사에 의해 유효하지 않은 데이타를 접근한 트랜잭션의 완료를 허용하지 않게 한다.

회피기반 프로토콜들에는 무효한 데이타를 결코 접근하지 않게 한다. 이것은 중복 관리 기법이 ROWA (read-one write-all)에 기반하기 때문이다. 따라서 회피기반 프로토콜에서는 트랜잭션이 완료됐을 때 존재하는 모든 갱신된 복사본들이 같은 값을 갖도록 보장할 수 있다. 탐지기반 프로토콜들과 회피기반 프로토콜들의 성질 및 장단점을 비교해보면 표 1과 같다.

표 1 탐지기반 접근 및 회피기반 접근의 비교

	서버 의존성	복잡도	캐시 유효성 검사	통신 메시지 수
탐지기반	높다	낮다	lazy	많다
회피기반	낮다	높다	eager	작다

탐지기반 프로토콜들은 일관성 작업을 단일 클라이언트와 서버만이 참여하기 때문에 회피기반에 비하여 간결하다는 것이 주된 장점이다. 그러나 서버에 의존성이 높다는 것이 단점이다. 이것은 증가되는 통신비용으로 인한 성능상의 오버헤드를 초래할 수 있다. 반면에, 회피기반 프로토콜들은 지역에 있는 데이타를 접근하고 갱신된 사항을 나중에 최종적으로 서버와 통신함으로 서버에 대한 의존성이 적지만 갱신된 사항에 대한 추가의 일관성 유지 기법이 요구되므로 복잡도가 증가한다. 그리고 유효성 검사는 탐지기반 프로토콜의 경우 트랜잭션이 필요한 데이타를 접근할 때에 유효성을 검사하도록 요구되므로 게으른(lazy) 유효성을 유지하며 회피기반의 경우는 무효한 데이타가 클라이언트 캐시에서 원자적으로 바로 제거되므로 부지런한(eager) 유효성을 유지한다고 할 수 있다[10].

이러한 두 가지 분류에 따라 갱신을 할 때, 캐시의 일관성을 어떠한 기법으로 유지하는가에 따른 여러 알고리즘들이 제시되어 왔다. 이들은 표 2와 같이 크게 S2PL(Server-based Two Phase Locking), O2PL (Optimistic Two Phase Locking), Callback Locking으로 나뉠 수 있다.

표 2 클라이언트 캐시 일관성 기법의 세가지 분류

Server-based 2PL	B2PL[2,5,6,7] C2PL[2,5,6,7]	탐지기반
Optimistic 2PL	O2PL-I[2,5,6,7] O2PL-P[2,5,6,7] O2PL-D[2,5,6,7] O2PL-ND[5,6,7]	회피기반
Callback Locking	Callback-Read[5,6,7,11] Callback-All[5,6,7,11]	

S2PL에는 가장 기본적인 B2PL(Basic 2PL) 그리고 C2PL(Caching 2PL)이 있으며 O2PL에는 지역에 있는 구버전 캐시의 내용을 없애느냐 새로운 값으로 갱신하느냐에 따라 O2PL-I(Invalidation)와 O2PL-P (Propagation)가 있다. 그리고 이 두가지 방법을 선택적으로 적용할 수 있도록 조합한 O2PL-D(Dynamic)와 O2PL-ND(New Dynamic)가 있다. Callback Locking에는 페이지에 쓰기 위한 의도(write intention)가 한 트랜잭션에서만 유효한지 아니면 다음 트랜잭션까지 계속 유효한지에 따라 Callback-Read, Callback-All이 있다.

이 알고리즘들 중에서 본 논문의 연구 대상인 O2PL 알고리즘은 캐시된 페이지 복사본의 갱신에 대한 일관성 작업이 트랜잭션 끝까지 지연된다. 임의의 클라이언트에서 수행되는 트랜잭션이 지역에 캐시된 페이지 복사본에 갱신을 수행한다. 그러나 이 갱신된 사항은 트랜잭션이 완료 단계에 들어갈 때까지 서버에게 전송하지 않는다. 완료 단계에서 클라이언트는 트랜잭션에 의해 갱신되어온 각 페이지의 복사본을 포함한 완료 요청 메시지를 서버에게 보낸다. 그러면 서버는 쓰기 잠금과 같은 갱신 잠금(update-copy locks)을 자신의 캐시된 페이지에 대하여 취득한다. 일단 이들에 대한 잠금을 얻으면 서버는 갱신하고자 하는 페이지의 캐시된 복사본을 갖는 다른 모든 클라이언트들에게 완료 준비 메시지를 보낸다. 이 메시지는 결국 캐시 일관성 작업을 위한 요구로서 메시지를 받은 클라이언트들은 완료를 요청한 트랜잭션을 위해 갱신된 페이지에 갱신 잠금을 요구한다.

일단 모든 관련된 갱신 잠금이 취득되면 무효화(invalidation) 또는 전파(propagation)와 같은 특정 O2PL의 수행 중 한가지를 실행한다. 이 두가지 원격 갱신 방법의 좋은 특징을 조합하여 적응적으로 수행하는 동적(dynamic) 그리고 다른 동적(new dynamic) 기법도 있다. 결국 O2PL은 하나에서 읽고 모두에게 쓰는(read-one/write-all) 개념에 의한 회피기반 프로토콜이다.

기존 논문의 모의 실험에서 O2PL-I는 여러 가지 작업부하(workload)에서 좋은 성능을 보인 반면, O2PL-P의 경우는 특정 작업부하에서 더 좋은 성능을 보였다[5,6,7]. O2PL은 데이타의 경쟁(contention)이 극단적으로 높은 경우의 작업부하를 제외하고는 C2PL보다 좋은 성능을 보였다. B2PL과 C2PL은 서버가 모든 잠금의 요청을 다루어야하는 것과 대조적으로 O2PL 기법은 클라이언트 캐시에 해당 페이지가 없는 경우를 제외하면 수행하는 트랜잭션이 완료시점에 도달할 때까지 서버와 어떠한 상호작용 없이 수행할 수 있다. 그래서 O2PL의 경우는 각 클라이언트가 완전한 기능의 잠금 관리자를 갖게된다.

C2PL, Callback Locking 알고리즘과 달리 O2PL의 경우 클라이언트에서 수행되는 트랜잭션이 완료를 요청할 때까지 지역에 캐시된 페이지의 복사본을 가지고 서버와 통신 없이 수행할 수 있다. 따라서 메시지 사용을 줄이고 통신비용을 줄일 수 있으며 접근하는 페이지 충돌(conflict)을 나중에 처리하기 때문에 동시성(concurrency)을 증가시킬 수 있다[2]. 그러나 교착상태에 의해 철회(abort)를 초래할 수 있다는 문제점이 있는데, 이러한 전역 교착상태 발생에 따른 트랜잭션 처리의 지연은 앞서 서술한 장점을 저해하는 요인이다. 또한 O2PL에서는 캐시 일관성 작업이 완료 시점부터 함께 이루어지기 때문에 완료 요청에 따르는 시간이 상당히 지연될 수 있다. 이때 만일 경쟁이 높다면 지연되는 시간은 더욱 가중될 것이다.

2.2 교착상태와 통지(Notification)

클라이언트 캐시의 일관성 유지를 위한 작업에서 잠금충돌에 의한 전역 교착상태가 발생할 수 있는데, 일반적으로 전역 교착상태를 감지하는 방법으로는 타임아웃에 기반한 감지(timeout-based detection)와 그래프에 기반한 감지(graph-based detection)방법이 있다[3,8].

타임아웃에 기반한 감지는 트랜잭션이 너무 오랫동안 블락될 때마다 데드락이 발생하였을 것이라고 추측하는 것이다. 타임아웃 시점은 평균적인 트랜잭션 수행 시간보다 큰 시간을 갖는다. 이 감지 방법은 분산 환경에서도 추가의 복잡도나 오버헤드 없이 간결하게 작동할 수 있는 장점이 있다. 그러나 실제로 교착상태가 아닌 트랜잭션을 철회시킬 수 있고 반대로 이미 발생한 교착상태를 타임아웃 시점까지 방치할 수도 있다.

그래프에 기반한 감지는 트랜잭션의 정점들과 이들간에 충돌로 인하여 명확하게 잠금을 기다리는 상황을 나타내는 간선들로 WFG(wait-for graph)를 유지하고 주기적으로 사이클의 유무를 검사하여 교착상태를 감지한다. 따라서 타임아웃에 기반한 방법과 달리 새로운 간선이 추가될 때마다 사이클의 유무를 검사하여 정확하게 옳은 교착상태를 감지할 수 있다. 그러나 그래프를 계속 유지보수 해야하는 비용이 따르게 된다. 따라서 교착상태가 빈번하지 않은 경우에 간선이 추가될 때마다 검사하는 것 대신에 주기적으로 검사함으로써 비용을 줄일 수 있다. 왜냐하면 교착상태는 자연적으로 사라지지 않으므로 계속 유지되기 때문이다. 그러나 타임아웃에 기반한 감지와 마찬가지로 주기적으로 검사함으로써 이미 발생한 교착상태를 필요이상으로 오랫동안 감지하지 못할 수 있다.

한편, 갱신된 정보에 대한 일관성을 위해서 그리고 교착상태를 감지하기 위해 필요한 정보를 통지(notify)하는 기법들이 있다. [11]에서 유효하지 않은 캐시의 객체를 사용함으로써 야기되는 트랜잭션의 철회를 줄이기 위해 낙관적인 잠금의 일종인 무-대기 잠금(No-Wait Locking) 기법에 통지를 통합한 통지 무대기 잠금(No-Wait Locking with Notification)을 제시하였다. 여기서 통지는 어떤 캐시된 객체가 완료된 트랜잭션에 의해 갱신되었을 때 서버가 현재 갱신된 객체를 캐시한 모든 클라이언트에 이 갱신된 객체를 전달하는 것이다. 그러나 클라이언트가 그 통지를 받기 전에 유효하지 않은 객체를 읽을 수 있기 때문에 반드시 유효한 객체만을 읽게됨을 보장 할 수 없었다.

또 [4]에서 제시한 통지 잠금(Notify Locks)기법도 통지 무대기 잠금과 유사하나 서버에서 클라이언트로 통지 메시지를 보낼 때 일련 번호를 함께 전송하여 만일 메시지의 일련 번호가 낮은 트랜잭션이 완료를 시도하는지를 검사한다. 이 검사를 위해 클라이언트가 메시지를 받을 때마다 통지 메시지의 결과로 철회되었던 트랜잭션인가를 확인한다. 그리고 완료할 것인지 철회할 것인지를 서버에게 답변한다.

그리고 보수적(conservative or static) 2PL에서는 미리 선언(predeclare)된 읽기 연산집합과 쓰기 연산집합으로 교착상태를 회피(avoid)하여 트랜잭션을 결코 철회시키지 않는다. 여기서 미리 선언하는 읽기 연산집합과 쓰기 연산 집합이 일종의 통지 역할을 하는 정보라고

할 수 있다. 그러나 읽기 연산집합 및 쓰기 연산집합을 미리 선언하는 것이 사실상 비현실적이다[9]. O2PL 캐시 일관성 유지 기법의 경우 쓰기 의도 선언(write intention declare)이 완료 요구 시점까지 지연되기 때문에 그 동안 수행된 한 트랜잭션의 읽기 연산집합 및 쓰기 연산집합 정보를 알 수 있다. 그리고 이 정보는 아직 수행중인 다른 클라이언트의 트랜잭션에게 마치 보수적 2PL에서 미리 선언하는 정보와 같은 통지로써 유용한 정보이다.

2.3 연구동기

앞서 서술한 여러 캐시 일관성 유지 기법 중 근본적으로 높은 철회율의 특성을 갖는 O2PL기법에서 타임아웃을 사용하여 교착상태를 감지하는 것은 철회율의 증가를 가중시킬 수 있으므로 그래프에 기반한 교착상태 감지 방법을 사용하는 것이 타당하다. 기존에 서술된 O2PL기법을 살펴보면 갱신잠금(update lock)과 쓰기잠금 간의 충돌 또는 서버에서 전역 WFG(wait-for graph)를 유지함으로써 전역 교착상태를 감지하는 기법을 사용하였다[5]. 그러나 전역 WFG를 사용하여 주기적인 교착상태를 감지할 때 이미 발생된 전역 교착상태 감지를 지연시킬 수 있으며 지연되는 트랜잭션에 의해 또 다른 트랜잭션이 지연되는 현상이 발생할 수 있다. 또한 지역 WFG를 전달할 때 통신 지연도 전역 교착상태 감지를 지연시킬 수 있고 그래프 유지비용도 간과할 수 없다[9]. O2PL기법에서 이러한 지연과 비용은 완료 시점에서 캐시 일관성 유지 기법이 수행되고 이 때 교착상태를 함께 감지하기 때문에 완료 요청에 따르는 시간을 더욱 증가시키는 요인이 될 수 있다. 또한 대부분의(약 90%) 전역 교착상태 발생이 두 간선에 의한 사이클로 이루어진다는 일반적인 사실[1,9]에 따라 주기적인 그래프 기반 감지는 이미 발생한 교착상태 방치 현상을 현저하게 나타낼 수 있다.

그러나 앞절에서 언급하였듯이 O2PL 특성상 얻을 수 있는 연산정보를 통지정보로 사용함으로써 불필요한 연산의 수행 및 지연을 막고 두 트랜잭션간에 이미 발생한 교착상태의 방치현상을 없애고자 한다. 따라서 O2PL 기법의 수행에 따르는 블락된(blocked) 트랜잭션과 그들 간의 전역 교착상태로 인한 지연을 줄이는 기법을 제안하고자 한다.

3. 시스템 환경

3.1 클라이언트-서버 참조 구조

본 논문을 위한 환경으로 페이지에 기반한 클라이언트 서버 DBMS는 네트워크에 하나의 서버와 클라이언트들의 집합으로 구성된다. 페이지는 여러 데이타 객체(object)를 포함한다.[1] 각 클라이언트는 서버에 저장된 데이타베이스의 페이지 복사본을 캐싱하여 공유한다. 그리고 일련의 트랜잭션을 발행하여 응용프로그램(application)을 각 클라이언트 상에서 수행한다.

클라이언트에서 사용하고자 하는 특정 페이지가 없으면 서버에 요청하여 자신의 캐시에 그 페이지의 복사본을 저장한다. 그리고 페이지에 기반한 클라이언트 캐싱에서 클라이언트는 사용하고자 하는 객체와 페이지간의 매핑(mapping)에 대한 책임이 있다. 서버도 페이지를 캐시하여 복사본을 갖게되며 클라이언트로부터의 페이지 요청은 이 서버의 캐시를 통해 제공된다. 서버에서 캐시 미스(miss)인 경우, 디스크에서 읽어온다.

트랜잭션에 의한 갱신은 갱신될 페이지의 캐시에서 수행된다. 완료를 수행중인 트랜잭션은 서버에게 모든 갱신된 페이지들과 관련된 로그 기록을 완료 메시지와 함께 보내게된다. 서버가 완료를 허락하면 완료 요청시 보냈던 갱신을 저장하고 앞으로 다른 클라이언트에 의한 페이지 요청은 갱신된 페이지의 상태가 주어질 것이다.

그림 1은 이러한 클라이언트 캐싱을 사용하기 위한 참조 구조(reference architecture)를 보여준다. 클라이언트들과 서버가 함께 원자적 직렬화 가능한(atomic serializable) 트랜잭션의 수행을 제공하는 동시성 제어 구조를 만든다. 서버의 DBMS는 실제 데이타를 소유하므로 최종적으로 데이타베이스의 접근과 트랜잭션 성질을 유지하는 책임이 있다. 그리고 영구적인 데이타베이스의 버전과 로그를 안정된 저장소(stable storage)에 저장하여 관리한다.

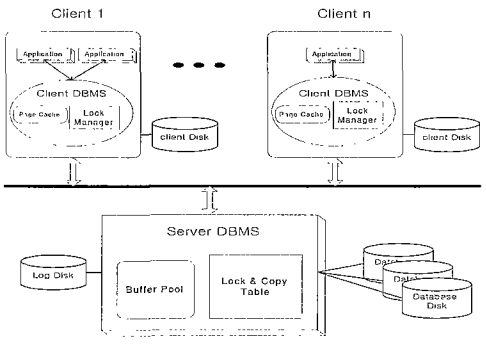

그림 1 페이지에 기반한 클라이언트-서버 DBMS참조 구조

1) 여기서 객체는 한 페이지의 일부에 해당하거나 여러 페이지에 걸쳐 나타날 수 있는 논리적인 데이타 단위이다.

각 클라이언트는 클라이언트 DBMS 프로세스를 실행시킨다. 이 클라이언트 DBMS 프로세스는 자신의 지역에서 수행하는 응용프로그램을 위한 데이타베이스 접근의 책임을 진다. 응용프로그램들은 그들의 실행을 지역 클라이언트 DBMS 프로세스로 요구한다. 그러면 서버 DBMS 프로세스에게 트랜잭션의 지원과 특정 데이타를 위한 요청을 전송한다. 서버 DBMS 프로세스는 그들이 소유한 데이타의 잠금 및 복사를 관리하는 기능을 제공한다. 그리고 데이타 페이지에 대한 가장 최근에 완료된 값을 재공하는 기능을 한다.

4. O2PL에서 신속한 완료 지원

4.1 O2PL-I 고찰

본 논문에서는 2장에서 살펴본 O2PL 캐시 일관성 알고리즘 중에서 전파 기법에 비하여 상대적으로 간결하고 통신이 적은 무효화를 통한 갱신을 사용하여 제안하는 기법을 전개한다. O2PL-I는 제안하고자하는 기법에서 통지의 메시지에 의하여 최소한 간결하고 빠른 트랜잭션의 완료에 적합하기 때문이다. 먼저 O2PL-I에서 완료시점에서 잠금 충돌에 의한 전역 교착상태의 예를 살펴보면 다음과 같다.

그림 2에서와 같이 클라이언트1(C1)에서 트랜잭션 T_1 : $r(x),w(x),r(y)$를 수행하고 클라이언트2(C2)에서는 트랜잭션 T_2 : $r(x),r(y),w(y),r(z)$를 수행하는 상황이라고 하자. 클라이언트 지역에서 읽기 및 쓰기 잠금의 요청이 r_req, w_req이고 그 요청에 의해 취득한 잠금이 각각 r_lock, w_lock이다.트랜잭션 T_1이 연산을 모두 수행하고 먼저 서버에게 갱신된 페이지 x의 복사본과 함께 완료를 요청하면 서버는 자신의 캐시에서 갱신된 페이지 x에 대하여 갱신 잠금을 얻고 페이지 x를 캐시한 C2에 캐시 일관성 작업을 요청한다. 이때 트랜잭션 T_1이 갱신한 페이지 x의 또 다른 복사본을 C2가 이미 읽은 상태이므로 잠금 충돌에 의하여 트랜잭션 T_1의 완료 요청에 따르는 상대 트랜잭션의 캐시 일관성 작업이 블락된다. 이때 클라이언트 C2는 블락된 트랜잭션간의 지역 WFG를 서버에 보내면 서버는 전역 WFG에 해당하는 간선을 추가한다.

그 후 C2의 트랜잭션 T_2에서 나머지 연산을 수행한 후 완료 요청을 서버에 하게되면 서버는 T_2에서 갱신한 페이지 y에 대하여 자신의 캐시의 해당 페이지에 갱신 잠금을 얻고 다시 페이지 y를 캐시한 페이지 복사본을 갖는 클라이언트 C1에 완료 요청을 하면, C1의 트랜잭션 T_1이 이미 페이지 y를 읽었으므로 캐시 일관성 작업이 블락된다. 따라서 C1의 지역 WFG를 서버에 보내면

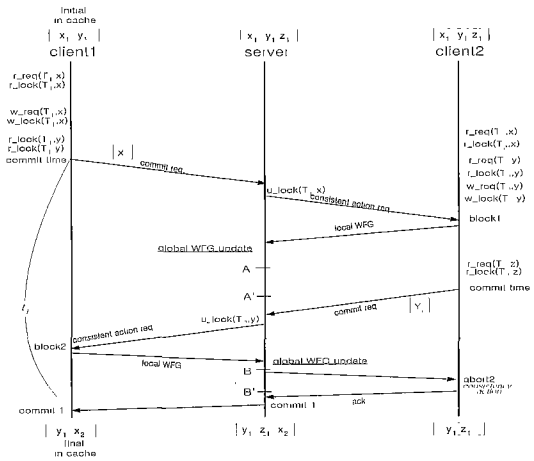

그림 2 두 트랜잭션간의 전역 교착상 감지 예

서버는 전역 WFG에 간선을 추가하여 그래프를 갱신한다.

결국 두 개 트랜잭션 T_1과 T_2에 의한 전역 교착상태가 발생한 경우인데 그림 2에서와 같이 서버의 전역 WFG를 주기적으로 검사하여 감지할 수 있다. 만일 A에서 검사가 이루어진 후 다음 검사 시점이 B인 경우, 교착상태가 감지되어 철회될 트랜잭션에 철회 메시지를 보내고 잠금을 얻은 완료 요청 트랜잭션의 일관성 작업을 "consistency action"에서 무효화로 수행한 후 서버에게 완료의 답변을 하면 해당 트랜잭션에 최종 완료 메시지를 보내게 된다. 따라서 갱신에 대한 완료를 요청한 C1과 서버만이 갱신된 페이지를 갖게 함으로써 캐시 일관성을 보장할 수 있게된다. 만일 그림에서 전역 WFG의 사이클 여부를 검사하는 시점이 A'와 B'였다면 그만큼 이미 발생한 교착상태를 즉시 감지하지 못하고 방치하는 결과가 된다. 이러한 트랜잭션의 블락된 상태는 완료를 요청한 트랜잭션의 지연은 물론 다른 트랜잭션의 새로운 블락까지 발생시킬 수 있다.

4.2 PN-O2PL 기법

4.1절에서 살펴본 O2PL-I 알고리즘은 지역에서 서버에 의존하지 않고 신속히 연산을 수행하지만 사실상 완료단계에서 발생할 잠금 충돌에 의한 블락과 교착상태의 문제점들은 트랜잭션의 원활한 수행과 자원의 활용측면에서 틀림없는 낭비이다. 따라서 본 절에서는 이러한 문제점을 최소화하는 해결방안을 제시한다.

2장에서 대부분의 전역 교착상태는 WFG에서 두 개의 간선에 의한 사이클로 이루어짐을 언급하였다. 따라서 이 두 트랜잭션간에 발생하는 전역 교착상태가 서버

에서 주기적인 전역 WFG를 검사할 때 발생 즉시 감지되지 않고 방치될 수 있었다. 여기서 두 트랜잭션간에 발생하는 전역 교착상태가 발생하자마자 즉시 감지되고 서버의 전역 WFG의 유지보수 비용을 줄일 수 있는 PN(Preemption by Notification)-O2PL 무효화 기법을 제안한다. 본 논문에서는 자연 발생의 우발적인 철회(spontaneous abort)에 의한 유령 교착상태(phantom deadlock) 문제는 고려하지 않는다.

제안하는 알고리즘은 기존의 O2PL-I 기법과 달리 트랜잭션이 완료를 요청할 때 트랜잭션이 수행한 읽기 연산집합을 통지의 정보로써 함께 전송하여 캐시 일관성 작업이 수행되는 다른 클라이언트의 트랜잭션 연산 집합들과 표 3의 조건을 경우 1, 경우 2 순서로 비교한다. 두 번까지 비교를 할 수 있는데 1차 비교는 처음 완료 요청으로 인한 캐시 일관성 작업이 요구될 때이다. 2차 비교는 1차 비교 후 나머지 연산을 수행한 후, 트랜잭션이 완료를 요청하려는 시점에서 추가된 자신의 읽기 연산집합 및 쓰기 연산집합과 다시 비교한다. 이와 같이 두 번 비교할 수 있는데 1차 비교에서 조건에 맞으면 2차 비교는 수행하지 않는다. 1차 비교에서 조건에 맞으면 교착상태를 미리 감지할 수 있어서 빠른 완료를 지원할 수 있으며, 2차 비교를 해야될 경우는 중앙집중형 전역 교착상태 감지기법을 위해 지역의 WFG를 서버로 전송해 다시 새로운 전역 WFG로 조합하고, 그 후 검사 주기에 사이클을 찾아보는 시간보다는 최소한 작거나 같게된다. 그 이유는 클라이언트에서 변경된 최종 지역 WFG를 서버가 받은 후 수정된 전역 WFG에 의해서 전역 교착상태가 감지될 수 있기 때문이다.

표 3 두 트랜잭션간에 교착상태 가능성 감지 조건

경우 1	$from\_ws(T_i) \cap to\_ws(T_j) \neq \varnothing$
경우 2	I) $from\_ws(T_i) \cap to\_rs(T_j) \neq \varnothing$ AND II) $from\_rs(T_i) \cap to\_ws(T_j) \neq \varnothing$

표 3에서 완료를 요청하는 클라이언트의 트랜잭션 T_i의 읽기 연산집합을 $from_rs(T_i)$으로 표현한다. 쓰기 연산집합(ws)은 가령 페이지 x에 대하여 갱신한 클라이언트가 완료 요청시 피기백(piggyback)되어 서버에게 오는 갱신된 페이지의 복사본으로 알 수 있는 정보이다. 그리고 from_ 의 연산 정보를 받는 클라이언트 측의 연산집합들은 to_ 의 단어머리로 구분하여 표현하기로 한다.

표 3의 조건은 두 트랜잭션간에 전역 교착상태가 발생할 수 있는 조건이다. 왜냐하면 먼저 경우 1, 경우 2는 모두 조건에 쓰기 연산을 포함하고 있으므로 상호 충돌관계임이 틀림없다. 그리고 O2PL기법에서 클라이언트에서 서버로의 완료 요청은 결국 갱신된 페이지의 일관성에 대한 처리이다. 따라서 경우 1은 결국 두 트랜잭션간에 쓰기 잠금으로 인한 교착상태가 T_i -> T_j -> T_i 사이클로 생성될 것이다. 경우 2에서 I)은 쓰기 잠금과 읽기 잠금에 의한 충돌 T_i -> T_j 이고 II)는 읽기 잠금과 상대트랜잭션이 쓰기 잠금간에 또는 앞으로 남은 연산에서 쓰기 잠금을 요청할 상황의 충돌이다. 따라서 완료를 시작하는 시점에서 캐시 일관성 작업을 요청하면 T_j -> T_i 형태로 블락될 것이다. 결국 T_i -> T_j -> T_i 의 사이클이 생성될 것이다. 경우 2에서 II)의 경우가 먼저 발생한 상태에서 그후 I)의 조건에 맞는 경우도 마찬가지이다. 그러므로 표 3는 두 트랜잭션간에 전역 교착상태를 발생시킬 조건이다.

제안하는 기법에서 표 3의 교착상태 감지 조건 비교는 두 트랜잭션간에 완료를 요청한 트랜잭션에게 신속한 완료를 지원할 수 있다. 표 3의 조건은 모두 교착상태가 발생될 경우이다. 그리고 처음 캐시 일관성 요청을 받은 클라이언트에서 표 3에 의한 1차 비교를 하여 경우 1 또는 경우 2에 해당할 때 to_ 측의 클라이언트는 전역 교착상태를 유발할 트랜잭션을 수행 중이므로 어차피 철회될 트랜잭션이다. 따라서 이 경우 바로 철회시켜 완료를 요청한 트랜잭션의 지연을 감소시킬 수 있다. 그러므로 제안하는 기법은 from_ 클라이언트측의 트랜잭션이 완료 요청 후, 발생할 교착상태를 바로 검사하여 트랜잭션의 완료를 신속하게 할 수 있게 한다.

다음은 PN-O2PL 기법의 클라이언트, 서버 프로토콜의 주요 부분을 보여준다. 그밖에 클라이언트 캐시의 관리 및 잠금 등은 O2PL-I와 동일하게 작동한다.

클라이언트 프로토콜

1. 서버로부터 다른 클라이언트의 캐시 일관성 요청을 받은 경우
   - 표 3의 교착상태 가능성 1차 비교 후
     - 경우 1 또는 2에 해당할 때, 자신을 철회하고 3번을 수행
     - 경우 2의 조건 I에만 해당하는 경우 변수 flag를 1로 셋팅 후, 상대 완료 요청을 블락시킨다. (조건 II에 해당하는 경우 변수 flag를 2로 셋팅)
2. 트랜잭션이 완료요청을 시도할 때
   - 나머지 연산을 수행한 경우, 수정된 연산집합과 교착상태 가능성 2차 비교
     - flag가 1인 경우, 경우 2의 조건 II가 성립할

경우 자신을 철회하고 3번을 수행
(마찬가지로 flag가 2인 경우는, 경우 2의 조건 I의 성립 여부 비교)
- 서버에게 갱신한 페이지의 복사본과 읽기 연산 집합을 함께 완료 요청 메시지 전송

3. 일관성 유지를 위한 작업을 수행
- 해당 잠금을 얻고 갱신 작업으로 무효화를 수행 후, 완료 요청에 대한 응답을 서버에게 보낸다.
- flag 변수를 초기화

서버 프로토콜

1. 클라이언트로부터 완료 요청 메시지를 받을 때
- 갱신잠금 취득
- 갱신된 페이지의 복사본을 캐시한 클라이언트에게 읽기 연산 집합과 함께 완료 요청

2. 완료 또는 철회 메시지를 받을 때
- 최종 완료 또는 철회를 결정하여 완료를 요청하였던 클라이언트에 결과를 알린다.
- 전역 WFG에서 해당하는 트랜잭션간의 간선이 있을 경우 삭제한다.

4.3 두 개 트랜잭션간에 적용

다음은 앞서 제안한 기법이 실제 두 트랜잭션간에 발생할 전역 교착상태의 상황을 예를 들어 설명한 것이다. 그림 3에서 4.2절에서 보인 예와 같은 트랜잭션 수행에서 제안하는 기법을 함께 표현하였다.

제안하는 알고리즘을 기존 알고리즘과 구별하기 위해 굵은 이탤릭 필체로 알고리즘의 행동 및 결과를 묘사하였다. 트랜잭션 T_1이 연산을 모두 수행하고 먼저 완료를 요청하면 갱신된 페이지 x에 대하여 서버는 갱신 잠금을 얻고 x를 캐시한 C2에 캐시 일관성 작업을 요청한다. 그러면 C2의 "*compare w, r set*"에서 표 3에 의하여 1차 비교를 한다. C1의 from_rs(T_1)={x,y}, from_ws(T_1)={x}와 C2의 to_rs(T_2)={x,y}, to_ws(T_2)={y}는 경우 2에 해당하므로 T_2를 철회시킨다. 이 시점에서 C2가 현재 수행중인 T_2의 나머지 연산을 수행하는 것은 불필요한 비용임이 자명하다. "*consistency action*"동안은 갱신 잠금을 얻고 무효화하고 잠금을 해제하는 작업이 이루어진다.

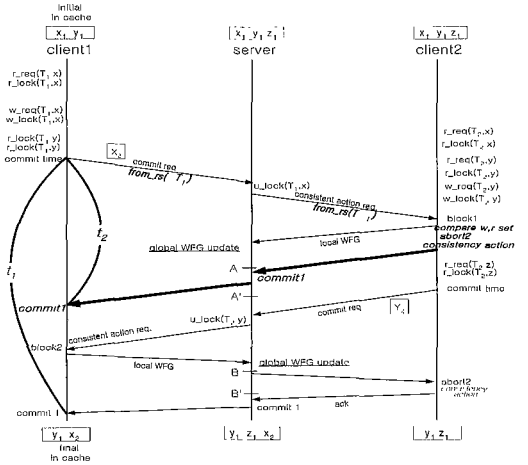

그림 3 PN-O2PL에 의한 전역 교착상태 감지 예

기존의 O2PL에서 C1이 완료하기까지의 지연 t_1과 제안하는 알고리즘에서 교착상태의 가능성을 미리 감지하고 완료하기까지의 시간 t_2간에는 t_1-t_2만큼의 차이가 있다. 이 시간차이에는 전역 교착상태를 감지하기 위해 사용한 전역 WFG를 유지관리하고 중앙에서 주기적으로 감지함으로써 발생하는 지연이 포함된다. 즉 시간 t_2이후에 수행된 모든 작업들이 수행되는 비용을 절감할 수 있다. 또 C2의 완료 요청의 메시지 및 지역 WFG의 서버로의 전달이 필요 없게 되었으며 서버에서 새로운 간선에 대한 추가작업도 절감할 수 있다. 결국 C1에서 트랜잭션 T_1을 수행하고 완료 요청 후 t_1-t_2만큼의 지연을 줄이고 신속히 완료할 수 있게된다. 이와 같이 완료 요청 후 잠금 충돌에 의한 지연을 최소화하고 전역 교착상태를 신속히 처리함으로써 또 다른 트랜잭션의 블락을 방지하게 함으로써 전체적인 응답시간에 향상을 가져올 수 있게된다.

5. 성능 평가

이 장에서는 기존의 O2PL-I알고리즘과 본 논문에서 제안한 PN-O2PL알고리즘의 성능을 비교하고 분석한다.

5.1 시스템 매개변수 설정

앞서 서술한 O2PL-I와 PN-O2PL의 성능을 실험하기 위하여 클라이언트, 서버, 트랜잭션들과 같은 시스템 구성요소의 매개변수를 설정한다. 시스템을 구성하는 이 요소들 가운데 서버, 네트워크, 디스크 등을 시스템 모델로 언급하며, 반면에 데이타베이스에 대한 트랜잭션의 수행에 관련되는 것을 작업부하 모델로 나누어 설정한다. 각 모델은 실험의 다양한 상황을 도출할 수 있는 매개변수 집합을 갖는다.

이 모의실험을 수행하는 동안 트랜잭션의 수행에서 사용되는 페이지의 수와 페이지를 접근하는 연산 유형의 변화는 성능의 변화에 영향을 끼치는 주요한 인자이다. 따라서 이 실험은 각 클라이언트가 캐싱할 수 있는 캐시의 절대적인 크기는 일정하게 유지하고 트랜잭션이 접근하는 페이지 수에 따라서 변하는 쓰기 연산의 비율

표 4 시스템 및 오버헤드 매개변수 설정

매개변수	설 정	의 미
PgSize	4 Kbytes	한 페이지의 크기
DBSize	1000 pages	데이타베이스 크기
NetBD	10 Mbps	네트워크 대역폭
NetDly	4 ms	메시지를 전송하는데 걸리는 지연시간
AccPg	25,50,100,200 pages	접근되는 페이지 수
CtCPU	100 MIPS	클라이언트 CPU의 처리 능력
SvCPU	200 MIPS	서버 CPU의 처리 능력
DiskIO	35 ms	I/O를 수행할 때 평균 오버헤드
DdlockCost	30 ms	WFG에 의한 교착상태 감지의 비용

표 5 작업부하 매개변수 설정

매개변수	설 정	의 미
TotalTR	1-50 개	수행되는 트랜잭션의 수
WrtProb	20, 40 %	전체 연산에서 쓰기 연산의 비율
MeanTRIntv	500 ms	트랜잭션이 서버에 도착하는 평균 간격
MeanOPIntv	250 ms	트랜잭션에서 연산간의 평균 간격
OPpTR	7-13 개	한 트랜잭션이 수행하는 연산의 수

을 주요 변화요소로 고려하여 실험하고 결과를 분석한다. 그리고 모의실험의 간결성을 위해 여러 클라이언트들에게 서비스하는 단일 서버 시스템을 실험 모델로 사용하며 각 클라이언트는 하나의 트랜잭션을 수행하고 완료 요청을 하기로 한다. 그리고 데이타베이스 크기는 비교적 작은 경우로 가정한다.

표 4에서는 시스템 모델의 매개변수 설정 및 오버헤드 설정을 나타낸다. 페이지 하나의 크기를 4 Kbytes로, 메시지를 전송하는데 걸리는 시간을 약 4 ms로 설정하였다. 서버의 CPU의 처리능력은 클라이언트보다 상대적으로 크게 설정하였다.

모의실험에 사용되는 작업부하의 매개변수 설정은 표 5에서와 같이 시스템에서 수행되는 전체 트랜잭션의 수를 설정하고 트랜잭션이 수행하는 연산의 비율을 두가지 경우로 나누어 설정하였다.

그리고 트랜잭션이 서버에 도착하는 평균 시간 간격과 트랜잭션에서 연산간의 평균 간격을 설정하였다.

5.2 모의실험 결과 및 분석

이 실험에서의 측정 결과는 수행하는 트랜잭션들이 접근하는 페이지의 수 및 작업 유형의 변화에 따른 응답시간(response time)이다.

5.1절에서 기술한 매개변수를 사용하여 기존의 O2PL-I기법과 본 논문에서 제안한 PN-O2PL 기법을 두 가지 매개변수의 변화에 따라서 성능 변화를 측정한다. 수행하는 트랜잭션들에 의해 사용되는 페이지의 수가 25, 50, 100, 200인 상황을 설정하고 각각에 대하여 쓰기 연산의 비율을 20%와 40%로 변화시켜 성능을 측정한다. 실제적으로 쓰기 연산이 읽기 연산보다 더 많은 경우가 희박하므로 쓰기 연산이 비교적 적고 많은 경우로 설정한다. 이러한 매개변수의 변화는 잠금 충돌 및 경쟁(contention)정도를 변화시킬 수 있는 요소들이다.

수행하는 트랜잭션들이 주로 접근하는 페이지가 매우 작은 수에 집중되어 있다는 것은 그만큼 다른 트랜잭션과 잠금 충돌에 의한 경쟁이 많아질 수 있는 환경이다. 그림 4에서와 같이 트랜잭션들이 주로 접근하는 어떤 페이지의 수가 25개인 경우, 200개인 그림 7에서 보다 평균 응답시간이 많이 걸린다는 것을 알 수 있다. 이것은 O2PL의 전반적인 단점으로 경쟁이 심할 경우 완료 요청 후, 함께 시작되는 일관성 작업을 수행함으로써 잠금 충돌로 인한 많은 지연이 발생하는 결과라 할 수 있다. 그림 4의 결과에서 나타나듯이 PN-O2PL의 경우는 불필요한 잠금 충돌에 의한 교착상태를 최소화하였기 때문에 그만큼 지연을 줄일 수 있었다. 그림 4에서와 같이 경쟁이 심한 경우에는 PN-O2PL의 40% 쓰기 연산의 작업에서의 평균 응답시간이 쓰기 연산이 더 적은 O2PL-I의 20% 쓰기 연산의 평균 응답시간 보다도 빠른 결과를 보여주고 있다. 이것은 대부분의 교착상태에 해당하는 두 트랜잭션간의 교착상태에서 제안하는 PN-O2PL의 읽기 연산에 대한 정보를 이용하여 표 3의 경우 2에 해당하는 상황을 고려한 성과임을 알 수 있다. 따라서 평균 응답시간을 경쟁이 심한 경우에 더욱 줄일 수 있었다.

이러한 경쟁이 있는 경우, 완료 요청 후의 지연되는 시간이 많아짐에 따라 또 다른 트랜잭션에 의한 교착상태로 철회의 수가 점점 늘어날 수 있다. 수행되는 트랜잭션의 수가 적은초기에는 두 기법간에 큰 차이가 없지만 점차 수행되는 트랜잭션의 수가 증가할 때 그러한 지연시간이 증가함으로써 교착상태가 발생할 가능성이 많아지고 철회되는 수가 많아지게 된다. 따라서 수행하는 트랜잭션의 수가 더 많아질수록 성능의 차이가 점차 벌어짐을 알 수 있다. 이러한 상황에서 PN-O2PL은 O2PL-I에 의해 조기에 감지할 수 있는 교착상태뿐만 아니라 두 트랜잭션간에 읽기 연산에 대한 정보를 이용법의 원래 장점은 그대로 살릴 수 있음을 잘 나타낸다.

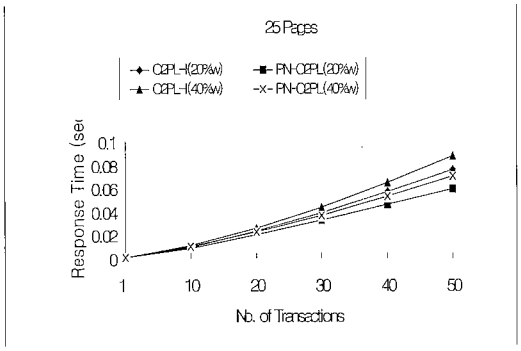

그림 4 25pages, 20% / 40% write operations

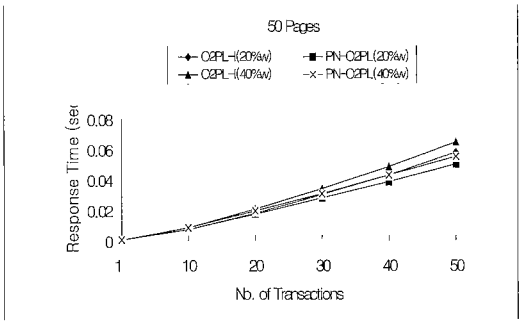

그림 5 50 pages, 20% / 40% write operations

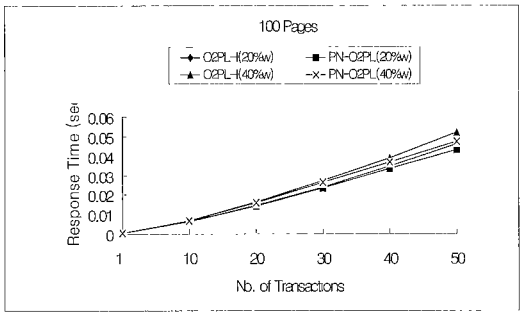

그림 6 100pages, 20% / 40% write operations

감지하여 신속히 완료하게 함으로써 완료 요청상태에서 지연되는 시간을 좀 더 줄여나가고, 다른 트랜잭션과 블락되어 또 다른 교착상태를 발생시키고 철회되는 수의 증가를 억제할 수 있게된다.

그리고 트랜잭션들에 의해 주로 접근되는 페이지수가 많아지는 그림 5, 6, 7에서 그 차이가 극명하지는 않지만 O2PL-I의 결과에 비하여 점차 약간의 성능향상이 있게 되는 것은 경쟁이 적은 상황에서 낙관적인 접근방

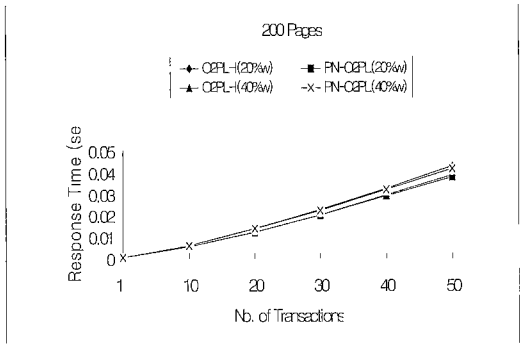

그림 7 200 pages, 20% / 40% write operations

하여 상대측과의 잠금 충돌에 의한 교착상태를 초기에 결국, 그림 4에서 O2PL-I의 응답시간과 PN-O2PL의 응답시간 사이의 차이가 그림 5, 6, 7에서 보다 가장 잘 드러남을 알 수 있는데, 그만큼 완료 요청시 발생하는 지연을 줄임으로써 얻어낼 수 있는 결과라 할 수 있다. 이것은 낙관적인 접근에서 고질적인 단점으로 말할 수 있는 높은 경쟁상태에서 성능저하를 개선한 점에 의의가 있다.

6. 결 론

이 논문은 클라이언트 캐시 알고리즘들 중 O2PL에서 캐시 일관성 작업 및 전역 교착상태 감지가 완료 요청 이후에 함께 이루어짐으로써 발생하는 지연이 성능저하의 주요 원인 중에 하나임을 지적하였다. 따라서 일관성 작업을 수행할 때, 연산에 대한 잠금 충돌로 인하여 발생하는 전역 교착상태의 증가를 억제하고 트랜잭션의 완료단계에서 지연을 줄일 필요가 있다. 그런데 이러한 전역 교착상태가 대부분 두 개 트랜잭션간에 발생한다는 일반적인 사실에 따라 두 트랜잭션간에 발생하는 전역 교착상태로 인한 지연을 최대한 줄임으로써 트랜잭션이 신속하게 완료를 할 수 있도록 지원한다.

본 논문에서는 이러한 지연을 막고 완료의 신속한 처리를 위하여 PN-O2PL을 제안하였다. PN-O2PL은 O2PL에서 트랜잭션이 지역의 캐시 데이타를 이용하여 연산을 모두 수행한 후, 완료 요청 시점에서 갱신에 대한 일관성 작업이 수행됨으로써 얻을 수 있는 읽기 연산집합 정보를 전역 교착상태의 문제를 해결하기 위한 통지로 사용하였다. 따라서 서버에서 전역 WFG로만 전역 교착상태를 감지할 때 이미 존재하는 전역 교착상태를 방치하는 가능성을 줄일 수 있었다. PN-O2PL기법은 전역 교착상태에 의해 철회될 트랜잭션이 불필요한

연산을 계속함으로써 발생하는 비용을 제거하였다. 이것은 완료 요청시 지연되는 트랜잭션에 의해 다른 트랜잭션이 블락되는 것을 억제하고 이것으로 인하여 또 다른 전역 교착상태의 가능성도 줄일 수 있게 하였다.

모의 실험 결과에서와 같이 트랜잭션이 접근하는 자원의 변화에서 각 작업 유형에 따라 PN-O2PL은 기존의 O2PL보다 거의 모두 우세한 성능을 보여주었다. 특히 잠금충돌과 경쟁이 심화되는 경우, 좋지 못한 성능을 보였던 O2PL-I와 달리 PN-O2PL은 O2PL-I보다 빠른 평균 응답시간으로 트랜잭션들을 처리할 수 있었다.

참 고 문 헌

[1] Andrew S. Tanenbaum, *Distributed Operating Systems*, Prentice-Hall Inc., 1995.

[2] Carey, M., Franklin, M., Livny M. and Shekita, E., "Data Caching Tradeoffs in Client-Server DBMS Architectures," *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, June, 1991.

[3] E. Knapp, "Deadlock Detection in Distributed Databases," *ACM Computing Surveys, Vol.19, No.4*, December 1987.

[4] K. Wilkinson and M. Neimat, "Maintaining Consistency of Client-Cached Data," *Proceedings of the 16th International Conference on VLDB*, August, 1990.

[5] M. Franklin, "Caching and Memory Management in Client Server Database Systems," *Tech. Report(Ph.D.) 1168, Computer Science Dept., Univ. of Wisconsin-Madison*, July 1993.

[6] Michael J. Franklin, Michael J. Carey and Miron Livny, "Transactional Client-Server Cache Consistency: Alternatives and Performance," *ACM Transaction on Database Systems, Vol. 22, No. 3*, September, 1997.

[7] M. Oszu, U Dayal and P. Valduriez, *Distributed Object Management*, Morgan Kaufmann Publishers, Inc, 1994.

[8] P. A. Bernstein and Eric Newcomer, *Principles of Transaction Processing*, Morgan Kaufmann Publishers, Inc, 1997.

[9] P. A. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company, 1987.

[10] Robert E. Gruber, "Optimism vs. Locking: A Study of Concurrency Control for Client-Server Object-Oriented Databases," *Tech. Report, MIT*, 1997.

[11] Y. Wang and L. Rowe, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June, 1991

유 제 혁

1997년 고려대학교 전산학과 이학사. 1999년 고려대학교 일반대학원 컴퓨터학과 이학석사. 1999년 ~ 현재 고려대학교 일반대학원 컴퓨터학과 박사과정. 관심분야는 동시성 제어, 분산 데이타베이스, 이동 컴퓨팅 시스템.

조 성 호

1994년 한국외국어대학교 전산학과 이학사. 1997년 고려대학교 일반대학원 전산과학과 이학석사. 2000년 고려대학교 일반대학원 컴퓨터학과 이학박사. 관심분야는 동시성 제어, 이동 컴퓨팅 시스템, 분산 운영체제

정 일 영

1994년 고려대학교 전산과학과 이학사. 1996년 고려대학교 일반대학원 컴퓨터학과 이학석사. 1996년 ~ 현재 고려대학교 대학원 컴퓨터학과 박사과정. 관심분야는 동시성 제어, 분산 데이타베이스, 이동 컴퓨팅 시스템.

황 종 선

1978년 University of Georgia, Statistics and Computer Science 박사. 1978년 South Carolina Lander 주립대학교 조교수. 1981년 한국표준연구소 전자계산실 실장. 1995년 한국정보과학회 회장. 1982년 ~ 현재 고려대학교 컴퓨터학과 교수. 1996년 ~ 현재 고려대학교 컴퓨터과학기술대학원 원장. 관심분야는 분산 데이타베이스, 이동 컴퓨팅 시스템, 결함포용 시스템.