정보학연구 제3권 제3호 2000년 9월(pp.117-134)
The Journal of Information Technology, Vol.3, No.3

그림자 트랜잭션을 이용한 지연 로킹 기법의 성능 평가

권혁민*

요 약

데이타전송(data-shipping) 모델에 근간을 둔 클라이언트-서버(client-server) DBMS는 트랜잭션간 캐싱(inter-transaction caching)을 허용함에 의해 클라이언트의 자원을 효율적으로 이용할 수 있다. 그러나 트랜잭션간 캐싱을 허용하면 각 클라이언트는 데이타베이스의 일부분을 동적으로 캐싱할 수 있기 때문에 트랜잭션 캐쉬 일관성 유지(transactional cache consistency maintenance: TCCM) 기법의 필요성을 야기한다. 검사기반 TCCM 기법은 클라이언트 중복사본의 유효성을 비동기적으로 검증하면 캐쉬 일관성을 유지하기 위한 메시지 부담을 줄일 수 있기 때문에 높은 트랜잭션 처리율을 보일 수 있다. 그러나 트랜잭션들이 유효하지 않은 중복사본을 액세스할 수 있기 때문에 트랜잭션 철회율이 높은 단점이 있다. 이 단점에 대처하기 위하여 본 논문에서는 철회되는 트랜잭션 대신에 실행되기 위하여 관리되는 백업 목적의 트랜잭션인 그림자 트랜잭션의 개념을 제안한다. 본 논문은 그림자 트랜잭션의 개념에 기초하여 DL-ST로 명명된 새로운 검사기반의 TCCM 기법을 제안한다. 그리고 모의실험을 통하여 트랜잭션 처리율과 철회율 관점에서 그림자 트랜잭션의 효과를 평가한다.

Ⅰ. 서론

고성능 워크스테이션의 등장과 네트워크 기술의 발전으로 클라이언트-서버 데이타베이스 관리 시스템(client-server DBMS)은 분산 계산 환경에서 중요한 요소가 되었다. 클라이언트-서버 DBMS는 정보교환 단위에 따라 데이터전송(data-shipping) 방식과 질의전송(query-shipping) 방식으로 분류된다[5, 8, 9]. 질의전송 방식에서 질의는 서버에서 처리되고 그 결과만이 클라이언트로 전송되는데, 대부분의 상용 관계형 DBMS들은 이 방식을 채택하고 있다[9, 16]. 반면, 데이타전송 방식에서 서버는 질의가 클라이언트에서 처리될 수 있도록 필요한 데이타를 클라이언트로 전송하는데, 대부분의 객체지향(object-oriented) DBMS들은[7, 11, 12] 이 방식을 사용하고 있다. 데이타 전송 방식은 DBMS 기능들의 대부분을 클라이언트로 이관하기 때문에 비교적 풍부하고 값싼 클라이언트의 자원들을 효율적으로 이용할 수 있는 이점이 있지만, 클라이언트가 다량의 데이타 전송을 요청하면 네트워크의 사용집중에 따른 병목 현상으로 인하여 심각한 성능 장애를 야기할 수 있다.

데이타 전송 시스템에서, 트랜잭션의 완료후에도 클라이언트에 캐싱된 데이타를 계속 유지관리하여 다른 트랜잭션들이 이를 사용하도록 허용하는 트랜잭션간 캐싱(inter-transaction caching)은 네트워크의 사용집중을 해결하기 위한 매우 효과적인 기술이다. 그러나 트랜잭션간 캐싱을 허용하면 각 클라이언트는 데이타베이스의 일부분을 동적으로 캐싱하게 되므로 공유되는 데이타의 중복사본(replica)이 여러 클라이언트에 존재하게 된다. 그러므로 트랜잭션간 캐싱은 기본적으로 동시성 제어(concurrency control)

* 세명대학교 소프트웨어학과 전임강사

와 중복사본 관리(replica management)의 양면성을 지닌 트랜잭션 캐쉬 일관성 유지(transactional cache consistency maintenance: TCCM) 기법의 필요성을 야기한다. TCCM 기법은 트랜잭션들이 일관성 있게 중복사본을 액세스한다는 것을 보장해야 하므로 구현하기가 복잡하고 실행시키는데도 부담이 뒤따른다. TCCM 기법의 잠재적인 이점 및 이에 연관된 실행 부담사이의 절충(trade-off)은 부하(workload)의 특징에 따라서 다르게 나타나기 때문에, DBMS의 성능은 채택된 TCCM 기법과 사용 환경에 따라 상당한 차이를 보일 수 있다.

따라서 다수의 TCCM 기법들이 제안되고 그들의 성능이 비교되었다 [1, 5, 8, 9, 10, 14, 17, 18]. 이들은 트랜잭션 실행결과의 정확성을 보장하는 방법의 차이에 따라 회피기반(avoidance-based) 기법과 검사기반(detection-based) 기법으로 분류된다[9]. 회피기반 기법은 비최신의 데이타가 클라이언트 버퍼에 캐쉬되는 것을 방지함에 의해 트랜잭션들이 유효하지 않은 데이타를 액세스하는 것을 원천적으로 불가능하게 하는 반면, 검사기반 기법은 비최신의 데이타가 일시적으로 클라이언트 버퍼에 캐쉬되는 것을 허용한다. 회피기반 기법은 일반적으로 ROWA (read-one write-all)[4] 방식을 채택한 분산 로킹 기법에 근간을 두고 있기 때문에 읽기 연산은 매우 효율적으로 진행되나, 최소한 트랜잭션들이 완료하기 전에는 그의 갱신 결과를 다수의 클라이언트 버퍼에 원자적으로 반영해야 하는 부담이 있다. 검사기반 기법은 이와 같은 부담을 야기하지 않는 대신, 트랜잭션들이 비최신의 데이타를 액세스할 가능성이 있다. 그러므로 최소한 트랜잭션이 완료하기 전까지는 자신이 액세스한 모든 데이타에 대한 유효성을 보장받아야 하는 부담이 있다.

회피기반 기법중에서 성능이 가장 우수한 것으로 알려진 O2PL(optimistic two phase locking) 기법은[5, 6, 8, 9] 트랜잭션의 완료단계에서 그 트랜잭션의 갱신 결과를 다수의 클라이언트 버퍼에 반영하는 과정에서 심한 완료 부담을 야기한다. 비동기적으로 클라이언트 중복사본의 유효성을 검사하는 기법인 지연 로킹(deferred locking)[19] 기법과 AOCC(adaptive optimistic concurrency control)[1, 10] 기법은 이와 같은 부담을 야기하지 않으므로 O2PL 기법보다 우수한 성능을 보인다[19]. 뿐만 아니라, 이 기법들에서 한 트랜잭션의 일관성 유지를 위한 실행에는 그 트랜잭션을 제기한 단일 클라이언트와 서버만이 관여하므로 O2PL 기법에 비해서 구현이 용이하며 고장에 매우 견고하다고 할 수 있다. 그러나 DL과 AOCC 기법에서 트랜잭션들은 유효하지 않은 중복사본을 액세스할 수 있기 때문에 트랜잭션 철회율이 높은 단점이 있다. 이 단점은 트랜잭션들이 전적으로 프로그램 제어하에 실행되거나, 또는 사용자에 의해 제기된 트랜잭션들이 철회시에도 더 이상의 입력이 없이 다시 실행될 수 있는 응용환경에서는, 트랜잭션 철회로 인하여 시스템의 성능이 저하되지 않는 한 그다지 큰 문제가 되지 않는다. 그러나 트랜잭션들이 사용자와 상호 협력적으로 진행되는 사용자 대화식의 응용분야에서는 트랜잭션의 철회는 사용자 작업시간의 심각한 낭비를 초래하므로 바람직하지 않다. 이와 같은 응용분야에서는 설사 시스템의 부담을 가중시키더라도 트랜잭션 철회의 부정적인 영향을 줄이기 위한 방법이 필요하다. 이와 같은 관점에서 본 논문에서는 철회되는 트랜잭션 대신 실행되기 위하여 관리되는 백업 목적의 트랜잭션인 그림자 트랜잭션의 개념을 제안한다. 그리고 이 개념을 동기적 검사기반 기법인 DL과 AOCC 기

법에 적용하여 그림자 트랜잭션의 효과를 트랜잭션 처리율 및 철회율 관점에서 평가한다.

이 논문의 구성은 다음과 같다. 2장에서는 기존에 제안된 여러가지 TCCM 기법들을 살펴보고, 3장에서는 본 논문에서 제안된 기법에 대하여 설명한다. 그리고 4장에서는 제안된 기법과 기존의 기법과의 성능을 비교하기 위하여 모의실험 모형을 설계하고, 5장에서는 실험 결과를 제시하고 분석한다. 마지막으로, 6장에서 본 논문의 결론을 맺는다.

II. 관련 연구

이 장에서는 본 논문의 성능 평가시 선정된 TCCM 기법을 중심으로 기존에 제안된 기법들을 살펴본다. 본 논문에서는 동시성 제어와 데이타전송의 단위가 데이타 페이지라고 가정한다. 검사기반 기법의 TCCM 기법은 중복사본의 유효성을 검사하는 시점에 따라 동기적 그리고 비동기적 기법으로 분류할 수 있다. 동기적 기법의 대표적인 알고리즘으로 C2PL(caching two phase locking)이[5, 8] 있는데, 이 기법은 클라이언트의 중복사본을 액세스하기 전에 그의 유효성을 검사해야 하며 서버의 중앙 로크표(central lock table)에 적절한 로크를 설정해야 한다. C2PL은 중복사본의 로그 일련 번호(log sequence number: LSN)와 서버의 LSN을 비교하여 중복사본의 유효성 여부를 검사한다. 이 기법은 매 액세스마다 중복사본의 유효성을 검사하며 적절한 로크를 설정하기 때문에 트랜잭션 철회율이 낮으나 심각한 통신 부담을 야기하는 단점이 있다. 비동기적으로 유효성을 검사하는 기법으로는 AOCC[1, 10] 기법과 DL[19] 기법이 있다. AOCC 기법은 트랜잭션의 실행단계에서는 어떤 제한도 가하지 않고 중복사본의 액세스를 허용한다. 대신 트랜잭션의 완료단계에서 액세스한 데이타의 유효성을 보장하는데, 이를 위해 각 트랜잭션이 읽고 갱신한 데이타에 대한 정보를 각각 읽기집합(read-set)과 쓰기집합(write-set)에 유지 관리한다. 각 클라이언트는 트랜잭션의 완료단계에서 이 두 집합과 자신의 버퍼에서 갱신된 새로운 데이타를 완료 요청과 함께 서버로 전송된다. 서버는 그 트랜잭션이 읽은 데이타중에서 무효화된 데이타가 있는지를 조사하여 그 트랜잭션의 완료 또는 철회를 결정한다. AOCC 기법은 데이타 충돌의 해결을 전적으로 트랜잭션 철회에 의존하므로 철회율이 높은 단점이 있지만, 중복사본의 유효성 검사를 위한 통신 부담을 야기하지 않기 때문에 매우 우수한 성능을 발휘한다.

DL 기법은 C2PL과 같이 주사본 로킹 기법에 근간을 두고 있다. 그러나 DL 기법에서 각 클라이언트는 자신의 중복사본을 이용하여 연산을 실행하고, 그에 대한 로크 설정 및 유효성 검사는 추후에 캐쉬미스로 인하여 클라이언트와 서버 사이에 정보 교환이 반드시 필요한 시점에서 요청되어 비동기적으로 실행된다. DL 기법은 다수의 로크 요청과 데이타전송 요청 메시지를 하나의 메시지 전문(packet)으로 구성함에 의해 클라이언트에 캐싱된 데이타의 유효성을 검사하기 위한 메시지 부담을 상당히 감소시킬 수 있기 때문에 다양한 부하환경에서 우수한 성능을 발휘한다[19]. 그러나 DL 기법도 AOCC 기법과 마찬가지로 트랜잭션들이 유효하지 않은 중복사본을 액세스할 가능성이 있기 때문에 트랜잭션 철회율이 높은 단점이 있다.

회피기반의 TCCM 기법은 한 트랜잭션의 갱신 결과를 다수의 클라이언트 버퍼에 원자적으

로 반영한다. 회피기반의 대표적인 알고리즘으로 O2PL [5, 6, 8, 9] 기법과 CBL(callback locking) [5, 8, 9, 17] 기법이 있다. O2PL 기법은 트랜잭션의 완료단계에서 갱신 결과를 파급시키는 반면, CBL은 실제 갱신 연산이 제기된 순간에 그 결과를 파급시킨다. O2PL은 ROWA 방식을 채택한 분산 낙관적 로킹[6] 기법에 기초를 두고 있다. 각 클라이언트는 트랜잭션의 실행단계에서는 지역 로크표에 적절한 로크를 설정하고 중복사본을 액세스하면서 연산을 진행한다. 트랜잭션이 완료단계에 도달하면 서버는 그 트랜잭션이 갱신한 데이타를 캐싱하고 있는 모든 클라이언트와 협조하여 그 갱신 결과를 각 클라이언트 버퍼에 원자적으로 반영한다. [5, 8, 9]에서 제시된 O2PL 기법중에서 변경될 데이타를 클라이언트 버퍼에서 제거하는 무효화 정책을 채택한 O2PL-invalidation(O2PL-I)은 다양한 부하 환경에서 우수한 성능을 발휘한다. 그러나 O2PL 기법에서 각 클라이언트에 설정된 로크들 사이의 충돌(conflict)은 트랜잭션의 완료단계에 가서야 감지되며, 갱신 트랜잭션이 완료를 시도할 때마다 캐쉬 일관성 유지를 위한 실행이 여러 클라이언트에서 실행되므로 완료 부담이 심한 편이다. 뿐만 아니라, 트랜잭션의 완료 단계에서 불필요한 클라이언트 버퍼 무효화가 발생할 수 있다[19].

CBL은 O2PL 기법과는 다르게 갱신 연산에 대한 권한을 트랜잭션 실행중에 전역적으로 승인 받는다. 따라서 완료단계에서는 오직 그 트랜잭션을 제기한 클라이언트와 서버만이 관여하므로, CBL 기법은 완료 연산의 실행을 위하여 필요한 부담이 적은 편이다. 그러나 트랜잭션의 실행단계에서 전역적인 갱신 권한을 획득하는 과정에서 다수의 클라이언트와 통신해야 하며 불필요한 클라이언트 버퍼 무효화를 야기할 수 있다. 회피기반 기법은 클라이언트의 캐쉬 일관성을 항상 유지시켜 읽기 연산은 매우 효율적으로 실행되며 트랜잭션 철회율이 낮은 장점이 있다. 그러나 최소한 어떤 트랜잭션이 완료하기 전에는 그 트랜잭션의 갱신 결과를 다수의 클라이언트 버퍼에 원자적으로 반영해야 하므로 쓰기 연산을 실행하기 위한 부담은 상당히 큰 편이다. 그리고 만일 어떤 클라이언트가 고장으로 인해 동작할 수 없다면, 그 클라이언트가 캐싱하고 있는 데이타를 갱신하려는 트랜잭션의 완료는 성공할 수 없는 단점이 있다.

III. 그림자 트랜잭션을 이용한 지연 로킹 기법

3.1 그림자 트랜잭션의 기본 개념

비동기적 검사기반 기법인 DL과 AOCC 기법은 클라이언트 캐쉬 일관성 유지를 위한 메시지 부담이 매우 낮기 때문에 우수한 성능을 발휘한다[19]. 그러나 이 기법에서는 트랜잭션들이 중복사본을 액세스한 다음에 그의 유효성이 검증되므로 트랜잭션의 철회율이 높은 단점이 있다. 그러나 이 요인도 트랜잭션 대기와 철회사이에 균형을 유지시키는데 어느 정도 기여를 하므로 성능을 향상시키는데 일조를 한다. 특히 높은 데이타 충돌 환경에서 어느 정도의 트랜잭션 철회는 빈번한 로크 충돌로 인하여 과다한 수의 트랜잭션들이 대기상태에 빠지는 현상을 완화시켜 주기 때문에 성능에 크게 기여한다[2, 5, 19]. 그러나 모든 환경에서 성능만이 유일하게 중요한 요소는 아니다. 예를 들어, 사용자와 상호 협조적으로 진행되는 트랜잭션이 많이 제기되는

환경에서는 사용자의 작업시간의 양도 시스템 성능만큼 중요하다. 이와 같은 관점에서 설사 시스템의 부담을 가중시키더라도 트랜잭션 철회의 부정적인 영향을 줄이기 위한 방법이 필요하다. 그림자 트랜잭션의 기본 개념은 트랜잭션의 철회에 대비하여 적절한 시기에 그 트랜잭션과 동일한 결과와 상태를 갖는 백업 목적의 트랜잭션인 그림자 트랜잭션을 준비하고, 만일 원래의 트랜잭션이 철회되어야 하는 경우에는 그 트랜잭션을 처음부터 다시 실행시키는 대신에 그림자 트랜잭션을 실행시켜 낭비되는 작업량을 줄이자는 것이다. 본 논문에서는 트랜잭션이 직렬화가능성을 위반할 가능성이 있는 경우에 그림자 트랜잭션을 생성한다.

3.2 그림자 트랜잭션 개념의 적용

본절에서는 [19]에서 제안된 DL 기법에 그림자 트랜잭션의 개념을 통합하여 DL-ST(deferred locking with shadow transaction)이라는 새로운 TCCM 기법을 제안한다. DL 기법은 클라이언트 중복사본의 유효성을 항상 보장하지는 않는다. 그리고 DL 기법은 중복사본의 유효성을 검사하지 않고 그들을 액세스하는 것을 허용하고, 추후에 트랜잭션들이 액세스한 중복사본이 유효하지 않다면 해당 트랜잭션을 철회한다. DL-ST 기법은 클라이언트의 중복사본을 액세스하는 것을 직렬화가능성을 위반할 잠재적 위협으로 간주한다. 그러므로 중복사본을 액세스하기 전에 현재 실행되고 있는 트랜잭션을 위해 그림자 트랜잭션을 생성한다. 그림자 트랜잭션은 원래의 트랜잭션이 액세스한 중복사본에 대한 유효성 여부가 파악될 때까지 대기 상태에 둔다. 원래의 트랜잭션은 클라이언트의 중복사본은 항상 유효하다고 낙관적으로 가정하고 이를 액세스하면서 자신의 연산을 실행한다. 그러므로 이 트랜잭션은 중복사본 관리 측면에서 낙관적(optimistic) 트랜잭션으로 간주되는 반면, 그림자 트랜잭션은 중복사본이 유효하지 않다고 가정하고 대기 상태에 있기 때문에 비관적(pessimistic) 트랜잭션으로 간주된다. 추후, 액세스한 중복사본이 유효하지 않다고 판정되면, 원래의 트랜잭션이 철회되어야 한다. 이때 그림자 트랜잭션들중 하나가 새로운 낙관적 트랜잭션으로 선택되어 원래의 트랜잭션을 대신하여 실행된다. 이와 같은 과정을 더 명확히 설명하기 위하여 다음의 실행 스케쥴을 살펴보자.

예 2(그림자 트랜잭션 개념의 적용): 트랜잭션 Ti가 그림 1의 시간 t2까지 성공적으로 실행되었다고 가정하자.

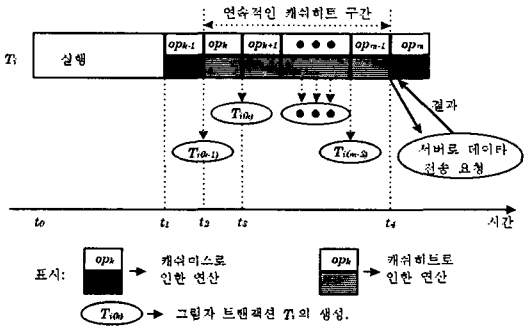

그림 1 그림자 트랜잭션의 생성

그림 1에서 opk-1 형태의 연산은 캐쉬미스로 인하여 서버의 로크표에 적절한 로크를 설정하고 데이타를 전송받은 후에 실행되는 연산을 의미하며, opk 형태의 연산은 캐쉬히트가 발생하여 중복사본을 액세스하면서 실행되는 연산을 의미한다. 그리고 Ti(k)는 Ti가 k번째 연산을 실행한 직후에 생성한 Ti의 그림자 트랜잭션임을 의미한다. 그러므로 Ti(k)는 Ti가 k번째 연산을

실행한 직후까지 산출한 동일한 결과와 상태를 가지고 있다. 본 논문에서 그림자 트랜잭션의 생성은 프로세스 포크(fork)로 구현한다고 가정한다. 그림 1에서 opk 연산은 캐쉉되어 있던 중복사본을 이용하여 실행되므로 Ti는 이 연산의 실행 직전에 Ti(k-1)을 생성한다. 이와 같은 방법으로 Ti는 캐쉬히트를 만날 때마다 새로운 그림자 트랜잭션을 생성한다.

Ti가 t4에서 캐쉬미스를 겪는다고 가정해 보자. 이 경우 Ti는 opm 연산을 수행하기 위하여 서버로 데이타전송 요청을 한다. 이 요청에는 t2에서 t4까지 액세스한 중복사본에 대한 로크 요청 및 그의 LSN이 첨부되어 전송된다. 서버는 클라이언트의 LSN에 기초하여 Ti와 그의 그림자 트랜잭션들의 운명을 결정하여 그 정보를 클라이언트로 전송한다. 예를 들어, Ti가 연산 opk+1의 실행시에 비최신의 데이타를 액세스했다면, Ti는 철회되어야 하며, Ti(k)가 대기상태에서 깨어나 Ti를 대신하여 실행되어야 한다는 정보를 클라이언트로 전송한다. 이 경우에 Ti의 다른 그림자 트랜잭션들은 Ti(k)보다 덜 진행되었거나, 또는 그들도 opk+1 연산의 실행시에 비최신의 데이타를 액세스했기 때문에 제거된다. 만일 Ti가 비최신의 데이타를 액세스하지 않았다는 것이 보장되면 Ti의 모든 그림자 트랜잭션들은 제거된다.

로킹 기법에서 충돌되는 로크들의 인증 순서는 그 로크를 요청한 트랜잭션간의 직렬화 순서(serial order)를 결정하므로 매우 중요하다. 그리고 이미 실행된 연산의 정확성은 로크에 의해 보장된다. 그러므로 그림자 트랜잭션의 결과는 그 트랜잭션이 생성될 때까지 원래의 트랜잭션을 위해 설정된 또는 설정될 것이라고 예상되는 로크에 의해 정확성이 보장되는 것이다. 그러므로 DL-ST 기법에서 트랜잭션 Ti가 철회되어 그의 그림자 트랜잭션들중의 하나로 대치될 때, 새로이 낙관적 트랜잭션으로 선택된 트랜잭션은 Ti의 로크를 계승받는다.

만일 트랜잭션이 n개의 데이타를 액세스한다면, 그 트랜잭션을 위해서 최대 n개의 그림자 트랜잭션이 존재할 수 있다. 이는 그림자 트랜잭션의 생성 및 그 결과를 저장하기 위하여 심각한 실행 부담과 버퍼 공간 부담을 야기할 수 있다는 것을 의미한다. 그러므로 시스템의 성능이 심각하게 저하되지 않도록 그림자 트랜잭션의 수를 제한하는 것이 바람직하다. 이와 같은 관점에서 본 논문에서는 하나의 트랜잭션을 위하여 한 순간에는 최대 k개까지의 그림자 트랜잭션을 허용하고, 이 기법을 DL-ST/k라 명명한다. 이 기법에서 어느 경우에 그림자 트랜잭션을 생성할 것인가는 매우 흥미있는 연구가 될 것이나, 본 논문에서는 알고리즘의 간략화를 위해 그림자 트랜잭션의 수가 k에 도달할 때까지 매 캐쉬히트마다 그림자 트랜잭션을 생성하는 정책을 채택한다. 데이타 충돌이 심한 환경에서 k 값은 크게 설정하는 것이 바람직할 것이다. 그러나 낮은 충돌환경에서 큰 k 값은 별다른 이점도 주지 못하면서 단지 그림자 트랜잭션과 연관된 부담만을 야기할 가능성도 있다. 이와 같은 사항을 고려하여 k 값은 데이타 충돌의 정도를 반영하여 한 시스템에서 동적으로 변경될 수도 있다. 그리고 k 값은 모든 트랜잭션에게 동일한 값으로 적용될 필요도 없다. 각 트랜잭션은 자신의 데이타 충돌 정도를 반영하여 자신만의 k 값을 가질 수도 있다. 본 논문에서는 이와 같은 연구는 미래의 과제로 남겨둔다.

Ⅳ. 모의실험 모델

이 장에서는 DL-ST 기법의 성능 평가를 위한 모의실험(simulation) 모델을 제시한다. DL-ST의 대표격으로 DL-ST/1을 선정하였는데, 그 이유는 비록 하나의 그림자 트랜잭션을 허용하더라도 그림자 트랜잭션으로 인한 성능 절충(performance tradeoffs) 현상을 충분히 파악할 수 있기 때문이다. 비교 대상으로는 DL 기법과 검사기반중 우수한 성능을 발휘하는 AOCC 기법을 선택하였다. 그리고 회피기반 기법중 우수한 성능을 발휘하는 것으로 알려진 O2PL-I의 성능도 제시한다. 모의실험 모델은 폐쇄 큐잉(closed queuing) 모델을[5, 8] 참고로 하여, MCC에서 개발한 CSIM[15] 언어를 이용하여 구현하였다.

서버의 트랜잭션 관리자는 서로 협조하여 트랜잭션의 실행을 관리하며, 적용된 각 TCCM 기법에 알맞게 동시성 제어 및 중복사본 관리의 실행을 책임진다. 클라이언트와 서버의 자원 관리자는 FIFO 원리로 디스크와 CPU와 같은 자원의 할당을 관리한다. 본 논문에서 클라이언트에는 디스크가 없고, 서버와 각 클라이언트는 각각 하나의 CPU를 소유하고 있다고 가정한다. 버퍼 관리자는 LRU 정책을 바탕으로 자신의 버퍼를 관리한다.

트랜잭션 및 시스템 자원, 그리고 실행 부담에 관한 입력 변수들은 각각 표1, 2에 제시되어 있다. 본 실험에서 데이타베이스는 DbSize의 페이지로 구성된다. MeanTranSize는 한 트랜잭션에 의해 액세스되는 평균 데이타 페이지 수를 정의하고, UpdateProbability는 액세스된 페이지가 갱신될 확률을 정의한다. 트랜잭션들이 실제

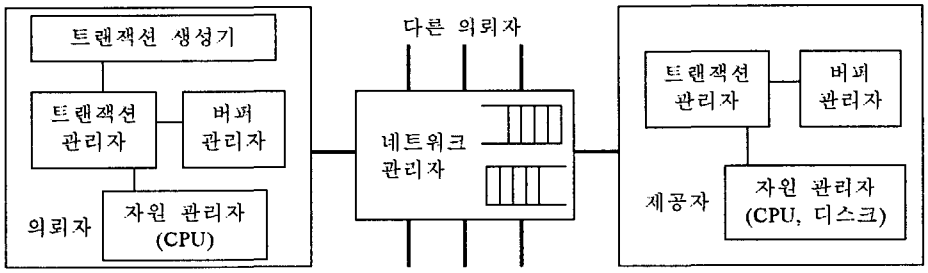

그림 2 모의실험 모델

모의실험 모델은 크게 네트워크 관리자, 클라이언트 모듈, 그리고 서버 모듈로 구성된다. 네트워크 관리자는 서버와 클라이언트의 정보교환을 관리하기 위하여 FIFO 큐 형태로 구현되었다. 트랜잭션 생성기는 한 트랜잭션이 성공적으로 종료하면 즉시 다음 트랜잭션을 생성하여 시스템에 제기한다. 이 논문에서 클라이언트만이 트랜잭션을 제기한다고 가정한다. 클라이언트와

액세스하는 페이지 수는 MeanTranSize의 20% 편차 구간에서 균등 분포(uniform distribution)를 이룬다. 모의실험에서 트랜잭션의 재실행 정책에는 기본적으로 진짜 재실행(real restart: RR)과 허위 재실행(fake restart: FR)의 두 방법이 있다. RR 정책에서 트랜잭션이 철회되면 그 트랜잭션이 처음부터 다시 실행된다. 반면, FR 정책에서는 원래의 트랜잭션이 다른 트랜잭션으

로 대치되어 실행된다. 본 논문에서는 트랜잭션이 철회되면 사용자는 원래의 트랜잭션을 다시 실행하기보다는 다른 트랜잭션의 실행을 원할 수도 있다는 점을 고려하여 FakeRestartPct로 명기된 위조 재실행 확률을 정의한다. 즉 트랜잭션이 철회되면, 그 트랜잭션은 FakeRestartPct의 확률로 다른 트랜잭션으로 대치되어 재실행된다.

은 제어 메시지는 CtrlMsgSize로 정의되는데, DL기반 기법과 AOCC는 각 제어 메시지에 다른 정보들을 첨부할 수 있기 때문에 이 크기를 두 배로 설정한다. 디스크 수는 4로 설정하며, 디스크 액세스 시간은 MinDiskAcc에서 MaxDiskAcc까지 균등 분포를 이룬다. 트랜잭션이 성공적으로 완료하면, 서버는 그 트랜잭션이 갱신한 페이지를 자신의 버퍼에 반영하고 그 페이

표 1 트랜잭션 및 시스템 자원 변수

입력 변수	의미	설정
DbSize	데이타베이스 크기	1000 페이지
PageSize	페이지 크기	4096 바이트
MeanTranSize	평균 페이지 액세스 수	20 페이지
UpdateProbability	갱신 확률	20%
FakeRestartPct	위조 재실행 확률	20%
NumClients	클라이언트의 수	1-25 클라이언트
NumDisks	디스크의 수	4 디스크
ClientCPU	클라이언트 CPU 속도	15 MIPS
ServerCPU	서버 CPU 속도	30 MIPS
NetBandwidth	네트워크 대역폭	10 Mbits/sec
ClientBufSize	클라이언트 버퍼 크기	25% of DbSize
ServerBufSize	서버 버퍼 크기	50% of DbSize

클라이언트 CPU는 트랜잭션이 한 페이지를 읽을 때마다 PageInst의 명령수를 실행해야 하며, 쓰기 연산을 위해서는 이 두 배의 명령수 처리 시간이 필요하다. 네트워크 전송 속도는 10Mbps를 기본값으로 설정하였는데, 네트워크 속도 변화에 따른 성능 추이도 살펴보았다. 메시지를 처리하기 위한 CPU의 부담은 각 메시지마다 반드시 필요한 FixedMsgInst와 메시지에 포함된 각 페이지마다 필요한 PageMsgInst로 모델링되었다. 이 명령수를 실행하는 부담은 메시지를 전송하는 사이트와 회신하는 사이트에서 모두 필요하다. 로크 및 데이타전송 요청과 같

지에 변경 표지를 남겨 놓는다. 변경 표지가 설정된 페이지는 버퍼 교체 알고리즘에 의해 희생자로 선택될 때 디스크에 기록된다. 단일 디스크 입출력을 위해서는 DiskOhInst의 명령수 처리시간이 필요하다. 한 데이타의 로크를 위해 LockInst 수의 CPU 처리 부담을 필요로 한다. DL기반 기법은 다수의 로크를 모아 다음에 전송될 메시지에 첨부하는데, 로크가 모아질 때마다 LockInst 수의 명령수 처리시간이 필요하다. AOCC는 트랜잭션의 읽기 집합에 속한 각 데이타가 무효화 정보에 포함되어 있는지를 조사하는 검증(validation) 방법을 채택하고 있다. 각

데이타의 검증을 위해서, 무효화 페이지가 적을 경우에는 페이지당 10 명령수, 무효화 페이지가 30이상이면 고정적으로 300 명령수 처리 부담이 필요하다. 특정 데이타를 캐싱하고 있는 사이트를 조사하는 시간, 특정 데이타에 대한 새로운 캐싱 사이트의 추가 및 제거를 위한 처리 부담은 RegisterInst로 정의한다.

크하고 20개의 데이타 페이지가 다 복사될 수 있도록 충분히 크게 설정하였다. 한 그림자 트랜잭션의 결과와 상태를 저장하기 위해 필요한 공간 부담은 SpaceForShadow로 정의하였다.

본 논문에서 사용된 주요 성능지수는 초당 처리되는 트랜잭션 처리율과 트랜잭션 응답 시간인데, 폐쇄 큐잉 모델에서 두 성능 중 하나의

표 2 시스템 부담 변수

입력 변수	의미	설정
CtrlMsgSize	제어 메시지의 크기	256 바이트
PageInst	페이지 읽기 연산을 위한 명령수	30,000
FixedMsgInst	메시지 처리를 위한 고정 명령수	20,000
PageMsgInst	페이지 수당 추가되는 명령수	10,000
LockInst	로크 설정/해제를 위한 명령수	300
PageValInst	페이지당 검증에 필요한 명령수	10 ~ 300
RegisterInst	캐싱 사이트 조사 / 추가 / 삭제를 위한 명령수	300
ForkShadowInst	그림자 트랜잭션을 생성하기 위한 명령수	100,000
SpaceForShadow	그림자 트랜잭션의 결과 및 상태 저장을 위한 부담	10 페이지
DiskOvhdInst	디스크 I/O를 위한 명령수	5000
MinDiskAcc	최소 디스크 액세스 시간	10 millisec.
MaxDiskAcc	최대 디스크 액세스 시간	30 millisec.

그림자 트랜잭션은 기본적 어떤 트랜잭션의 복사본이므로 프로세스 포크(process fork)나 쓰레드 포크(thread fork)를 이용하여 구현할 수 있을 것이다. CVAX 단일 프로세서 워크스테이션에서 Ultrix 널 프로세스(null process)를 생성하는데 11,300s의 실행시간이 필요하고, 쓰레드를 포크하기 위해서는 948s의 실행시간이 필요하다고 보고 되었다[3]. 본 논문에서는 프로세스 포크를 이용하여 그림자 트랜잭션을 생성한다고 가정하고, 그림자 트랜잭션이 생성될 때마다 ForkShadowInst 의 명령수 실행 부담을 클라이언트 CPU에 부과한다. ForkShadowInst의 값을 100,000으로 설정했는데, 이 값은 프로세스를 포

결과를 알면 다른 값은 계산을 통하여 파악할 수 있기 때문에 트랜잭션 처리율만을 제시한다. 트랜잭션 철회율과 완료되는 트랜잭션당 필요한 메시지 수 및 버퍼의 히트율과 같은 보조 성능지수들이 성능 결과를 분석하기 위하여 조사되었다. 그리고 트랜잭션 대기율이 조사되었는데, 이는 시스템에 제기되어 있는 전체 트랜잭션 대비 로크 인증을 기다리며 대기 상태에 빠진 트랜잭션 수의 비율을 의미한다. 본 모의실험은 복사 방법(replication approach)을[13] 사용하여 실행되었는데, 실험 시작시의 초기 편향(initial bias)을 제거하기 위하여 초기 800개의 트랜잭션 완료의 결과는 무시하였다. 본 논문의 결과값은

6개의 다른 임의 수를 사용하여 실시된 모의실험 결과의 평균값으로서, 각 복사(replication)는 5000개의 트랜잭션이 완료할 때까지 실시되었다. 이렇게 산출된 결과는 95%의 신뢰 수준을 만족한다.

V. 실험 결과 및 분석

본 모의실험은 다양한 데이타 공유 형태를 반영하기 위하여 UNIFORM, HIGHCON, 그리고 HOTCOLD라고 명명된 세가지의 부하(workload)에서 실시되었는데 각 부하의 특징은 해당 절에서 설명한다. 본 논문에서는 O2PL-I의 전역 교착상태 검사 주기를 [5, 8]에서와 같이 1초로 설정하였다. 그리고 한 트랜잭션의 갱신 결과를 다수의 클라이언트에 브로드캐스팅(broadcasting) 방식으로 파급시키는데, 이 메시지를 보낼 때 필요한 부담은 하나의 제어 메시지를 보내는 것과 동일하게 설정하였다. DL-ST/1 기법에서는 캐쉬미스시에 어떤 트랜잭션 중간결과의 유효성이 성공적으로 검증되고 나면, 그 중간결과의 유효성은 이후에도 계속 보장된다. 그러므로 DL-ST/1 기법에서 캐쉬미스 이후에 그림자 트랜잭션을 생성해 놓으면, 추후에 트랜잭션이 비최신의 데이타를 액세스하여 철회되면 그림자 트랜잭션이 원래의 트랜잭션을 대신하여 실행될 수 있다. 그러나 AOCC-ST/1(AOCC with shadow transaction) 기법에서는 그림자 트랜잭션을 생성해 놓더라도 다른 트랜잭션이 그의 결과를 아무런 제한없이 무효화시킬 수 있기 때문에 그림자 트랜잭션의 생성을 액세스하는 데이타 수의 관점에서 트랜잭션 평균 길이의 중간에서 하는 것이 바람직하다. 그러므로 본 논문에서 AOCC-ST/1 기법은 트랜잭션 평균 길이의 중간에서 그림자 트랜잭션을 생성한다.

5.1 실험 1- UNIFORM 부하

UNIFORM 부하에서 각 트랜잭션은 데이타베이스에서 임의의 페이지를 선택하여 액세스한다. 그러므로 모든 페이지는 동일한 확률로 액세스된다. 이 부하는 낮은 접근 국부성(reference locality)을 보이며, 데이타 충돌(data contention)의 정도는 HIGHCON 부하와 HOTCOLD 부하의 중간 정도이다.

트랜잭션 처리율과 트랜잭션 완료당 필요한 평균 메시지 수가 그림 3과 4에 있다. 본 논문의 모든 부하 환경에서 AOCC-ST/1 기법은 AOCC와 거의 유사한 트랜잭션 처리율을 보인다. 그러나 트랜잭션 철회율에 있어서 AOCC-ST/1이 AOCC-ST보다 우수한 성능을 보인다. 따라서 본 논문에서 AOCC-ST/1 기법의 결과는 트랜잭션 철회율만을 제시한다. UNIFORM 부하의 실험에서 대체로 DL-ST/1, DL, 그리고 AOCC 기법 순으로 우수한 성능을 발휘한다. NumClients의 증가에 따라 각 기법의 성능은 향상된다. 그러나 15-클라이언트 이상이 되면 성능이 저하되기 시작하는데, 이는 기법에 따라 약간 다른 원인에서 기인된다. 그림 5에서 O2PL-I 기법의 자원 사용율을 살펴보면, 15-클라이언트에서 서버 CPU와 네트워크 대역폭에 병목 현상이 발생되지 않는다는 사실을 파악할 수 있다. 이는 O2PL-I 기법의 경우 자원의 사용 경쟁으로 인하여 성능이 저하되는 것이 아니라 데이타 사용 경쟁으로 인하여 성능이 저하된다는 것을 의미한다. 즉, O2PL-I 기법에서 클라이언트의 수를 15 이상으로 증가시키면, NumClients의 증가가 동시성을 높이는데 기여하기보

다는 데이타 충돌만을 가중시켜 이로 인한 성능 저하를 야기한다는 것을 의미한다. AOCC는 기본적으로 데이타 충돌로 인한 트랜잭션 대기를 야기하지 않고, DL과 DL-ST/1은 비교적 낮은 트랜잭션 대기율을 보이므로 시스템의 자원을 매우 효율적으로 사용할 수 있다. 그렇지만 이 기법들은 15-클라이언트 이상이 되면 네트워크 사용 집중으로 인한 성능 병목 현상이 발생하여 성능이 더 이상 증가하지 않는다.

그림 4에서 트랜잭션 완료당 필요한 메시지 수의 변화 추이를 살펴보면, 회피기반 기법과 검사기반 기법은 뚜렷한 차이를 보인다. 회피기반 기법인 O2PL-I에서는 NumClients가 증가함에 따라 메시지 부담이 꾸준히 증가한다. 이의 근본적인 원인은 트랜잭션의 완료단계에서 그의 갱신 결과를 파급시켜야 할 클라이언트의 수가 증가하기 때문이다. 검사기반 기법에서 한 트랜잭션의 완료 연산에는 단일 클라이언트와 서버만이 관여하므로, 클라이언트 수가 증가하더라도 완료단계의 메시지 부담이 증가하지 않는다. 그러므로 DL-ST, DL, 그리고 AOCC 기법에서는 클라이언트의 수가 증가하더라도 메시지 부담이 그다지 크게 증가하지는 않는다.

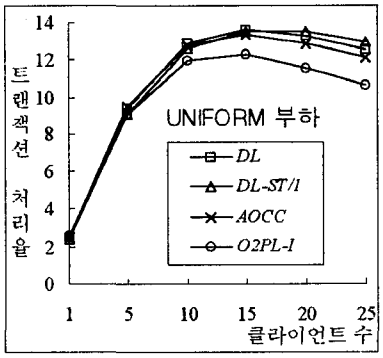

그림 3 트랜잭션 처리율

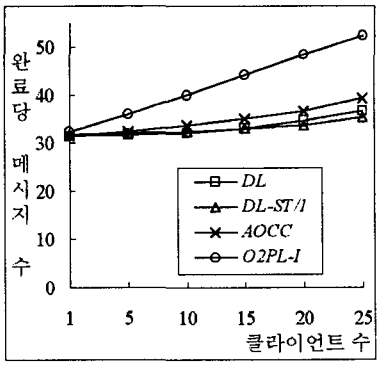

그림 4 완료당 전송되는 메시지의 수

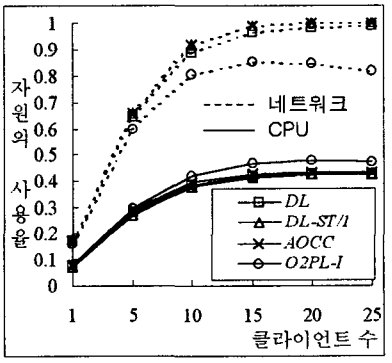

그림 5 자원의 사용율

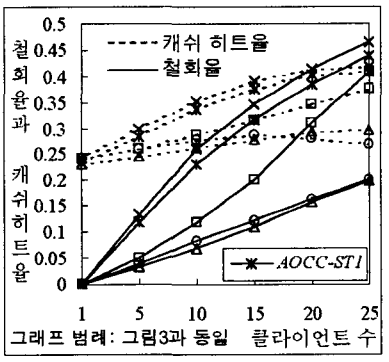

그림 6 철회율과 클라이언트 캐쉬히트율

회피기반 기법인 O2PL-I에서 각 클라이언트는 최신의 데이타만을 캐싱하고 있기 때문에 트

랜잭션들이 유효하지 않은 데이타를 액세스하는 것이 원천적으로 봉쇄된다. 그러므로 O2PL-I에서 트랜잭션 철회는 오직 로크 충돌에 의한 교착상태로 인하여 발생되므로, 클라이언트의 수가 증가하더라도 그 증가율이 그다지 크지 않다. 그러나 DL과 AOCC에서는 NumClients의 증가에 따라 트랜잭션들이 무효화된 데이타를 액세스할 가능성이 점점 높아지므로 철회율이 급격하게 증가한다. 두 기법에서 트랜잭션은 캐쉬미스마다 자신이 액세스한 데이타의 유효성 검사를 실시하는데, DL에서는 일단 검증된 데이타의 유효성이 완료단계까지 보장되는데 반해, AOCC에서는 그렇지 않다. 그러므로 AOCC는 DL보다 더 높은 철회율을 보인다. 그림자 트랜잭션을 적용한 기법을 비교하면, DL-ST/1 기법은 DL에 비해 50% 정도의 트랜잭션 철회율을 보이는데, AOCC-ST/1과 AOCC 기법은 철회율의 차이가 그다지 크지 않다. 이의 원인은 DL-ST/1에서는 그림자 트랜잭션의 결과가 로크로 인해 그 유효성이 보장되는데 비해, AOCC-ST/1에서는 그 결과가 다른 트랜잭션에 의해 언제라도 무효화될 수 있기 때문이다. 비록 각 기법이 매우 다른 철회율을 보이지만, 이것이 그들의 성능 및 메시지 부담에 심각한 영향을 미치지는 않는다. 왜냐하면 재실행되는 트랜잭션들은 그들이 필요로 하는 데이타를 이전 실행시에 자신의 버퍼로 옮겨 놓았을 가능성이 높기 때문이다. 캐쉬히트가 발생하면 트랜잭션 실행시에 클라이언트 CPU만이 사용되는데, 각 클라이언트에는 한 순간에 하나의 트랜잭션이 실행되므로 클라이언트 CPU 사용으로 인한 부담은 성능에 그다지 큰 영향을 미치지 않는다.

클라이언트 수의 증가는 캐쉬 히트율에 서로 상반되는 두 가지 효과를 가져온다. 첫째, NumClients가 증가하면, 데이타 충돌이 심화되므로 트랜잭션 철회율이 높아진다. 이는 트랜잭션 재실행시의 높은 캐쉬히트로 인하여 전체 캐쉬 히트율에 긍정적인 영향을 미친다. 반면, NumClients의 증가는 클라이언트 버퍼를 무효화시키는 속도를 가속시킨다. 이는 결과적으로 유효캐쉬크기(effective cache size)의 감소로 이어져 히트율에 부정적으로 작용한다. 여기서 유효캐쉬크기란 용어는 유효한 데이타 페이지를 캐싱하고 있는 버퍼 슬롯(slot)의 수를 의미한다. 그림 6에서 보는 바와 같이 15-클라이언트 지점까지 O2PL-I 기법의 캐쉬 히트율은 긍정적인 효과로 인하여 약간 증가하나, 15-클라이언트 이상에서는 부정적인 효과가 더 크게 작용하기 때문에 감소하기 시작한다. 반면, DL과 AOCC 기법에서는 높은 철회율로 인하여 긍정적인 효과가 부정적인 효과를 항상 압도한다. DL-ST/1 기법은 클라이언트의 버퍼가 그림자 트랜잭션의 결과와 상태를 저장하기 위하여 낭비되므로 작은 NumClients에서 다른 기법에 비해 낮은 캐쉬 히트율을 보인다. 그러나 NumClients가 증가함에 따라, DL-ST/1에서 트랜잭션이 철회되어 그림자 트랜잭션으로 대치될 확률이 증가한다. 그림자 트랜잭션은 원 트랜잭션이 철회된 지점까지 부분적으로 재실행되는데, 이때는 높은 캐쉬 히트율을 보일 수 있다. 따라서 DL-ST/1 기법은 NumClients가 증가함에 따라 캐쉬 히트율이 꾸준히 증가한다.

O2PL-I는 비최신의 데이타를 절대 클라이언트 버퍼에 캐싱하지 않기 때문에 유효캐쉬크기에 있어서 다른 기법에 비해 매우 우수한 성능을 발휘할 것 같지만 실제는 그렇지 않다. 비동기적으로 클라이언트 버퍼를 무효화시키는 DL과 AOCC 기법은 오히려 O2PL-I보다 큰 유효캐쉬크기를 보인다. 이는 O2PL-I에서는 한 트랜잭션의 갱신 결과를 다른 클라이언트로 파급시

키는 과정에서 불필요한 버퍼 무효화가 발생할 수 있기 때문이다. 참고로 25-클라이언트에서 AOCC, DL, DL-ST/1, O2PL-I는 각각 197, 193, 185, 180 페이지의 유효캐쉬크기를 보인다. 각 기법이 자신의 버퍼 용량 240 ~ 250 페이지를 최신의 데이타를 캐싱하는데 전부 사용하지 못하는 이유는 새로이 캐싱되는 페이지보다 무효화되는 페이지의 수가 더 많기 때문이다.

5.2 실험 2- HIGHCON 부하

이 부하는 모든 트랜잭션들이 HIGHCON이라 명명된 데이타베이스의 특정 부분을 매우 높은 확률로 액세스하는 경우를 모델링한다. 데이타베이스는 250 페이지의 HIGHCON 영역을 가지고 있고, 각 트랜잭션이 실행하는 연산의 80%는 이 영역을 액세스한다고 가정한다. 그러므로 이 부하는 매우 심한 데이타 충돌과 높은 참조 국부성의 특징을 지닌다. 그림 7에서 보는 것과 같이 DL, DL-ST/1, 그리고 AOCC는 O2PL-I에 비해 매우 우수한 성능을 발휘하는데, 그 이유는 이들은 상대적으로 낮은 트랜잭션 대기율을 보이며 트랜잭션 완료당 필요한 메시지 부담이 적기 때문이다. 참고로 DL과 DL-ST/1은 15-클라이언트에서 각각 24%, 28%의 트랜잭션 대기율을 보이는 반면, O2PL-I는 57%의 대기율을 보인다.

DL-ST/1 기법은 낮은 NumClients에서는 그림자 트랜잭션을 생성하는 실행 부담 및 그의 결과를 저장하기 위한 공간 부담으로 인하여 DL과 AOCC에 비해서 낮은 성능을 보인다. 그러나 NumClients의 증가에 따라 그림자 트랜잭션의 효과가 증대되어, 10-클라이언트에 도달하면 이들보다 더 우수한 성능을 발휘하기 시작한다. AOCC는 데이타 충돌로 인한 트랜잭션 대기를 야기하지 않을 뿐만 아니라, DL과 비슷한 수준의 철회율을 보인다. 그럼에도 불구하고 이들은 비슷한 성능을 보이는 이유는 트랜잭션 철회당 낭비되는 시간과 밀접한 관련이 있다. 철회당 낭비 시간은 트랜잭션이 제기되어 철회될 때까지의 평균 시간을 의미한다. 참고로 15-클라이언트에서 DL 기법은 AOCC 기법에 비해서 70% 정도의 철회당 낭비되는 시간을 보인다. 이와 같이 AOCC 기법에서는 트랜잭션이 비교적 많이 진행된 후에 철회되는 경향이 있기 때문에 자원의 낭비가 심한 편이다. 참고로 AOCC 기법은 15-클라이언트에서 53% 정도의 네트워크 낭비율을 보인다. 네트워크 낭비율이란 전체 네트워크 사용율 대비 철회되는 트랜잭션에 의해 사용되는 비율을 의미한다. 반면, DL 기법은 AOCC 기법과 비슷한 철회율을 보이지만 네트워크 낭비율이 40% 정도이며, DL-ST/1 기법은 그 낭비율이 25% 정도에 지나지 않는다.

이 실험에서 대부분의 데이타는 HIGHCON 영역에서 액세스되므로 각 기법은 UNIFORM보다 이 부하에서 높은 캐쉬 히트율을 보인다. 이 부하에서는 NumClients가 증가함에 따라 트랜잭션 철회율이 매우 급격하게 증가하는데, 트랜잭션 재실행시의 캐쉬히트가 UNIFORM의 경우만큼 그렇게 높게 예상되지는 않는다. 그 이유는 전 실행에서 클라이언트 버퍼로 가져다 놓은 데이타들이 곧 다시 무효화될 가능성이 높기 때문이다. 따라서 NumClients의 증가에 따라 메시지 부담은 UNIFORM의 경우보다 훨씬 급격하게 증가한다. NumClients에 따른 메시지 수의 변화 추이 및 요구되는 메시지 수의 상대적인 순서는 UNIFORM의 경우와 거의 동일하다.

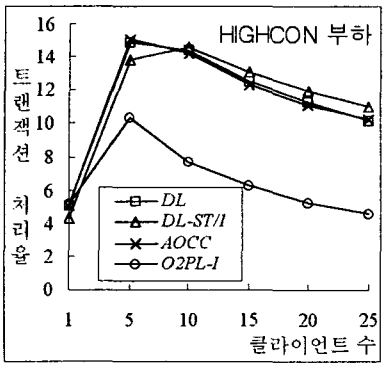

그림 7 트랜잭션 처리율

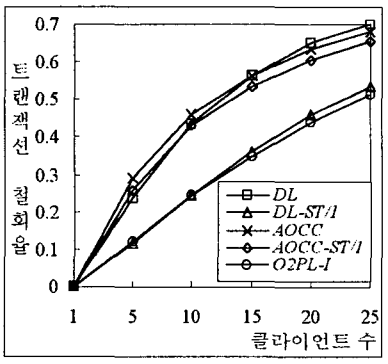

그림 8 트랜잭션 철회율

DL 기법은 트랜잭션들이 액세스한 데이타를 다른 트랜잭션들이 무효화시키지 않도록 로크를 설정하기 때문에 AOCC 기법보다 낮은 트랜잭션 철회율을 보일 것 같지만, HIGHCON 부하의 결과는 그렇지 않다. 데이타 충돌이 낮은 환경에서 로크 인증을 위하여 대기하고 있는 트랜잭션은 추후에 로크 인증을 받아 실행을 계속하게 될 가능성이 높기 때문에 로크는 철회율을 줄이는데 큰 공헌을 할 수 있다. 그러나, 데이타 충돌이 극심해짐에 따라 이와 같은 대기는 대부분 교착상태로 이어져 트랜잭션 철회에 대한 로크의 효과가 급격하게 줄어들게 된다. 또한 AOCC에서는 트랜잭션이 많이 진행된 후에 철회되는 경향이 있는데, 이도 재실행되는 트랜잭션의 철회율 관점에서는 긍정적으로 작용한다. 트랜잭션이 많이 진행된 후에 철회되어 처음부터 다시 실행되면, 그 트랜잭션은 이전 실행에서 자신이 필요로 하는 데이타의 상당량을 클라이언트 버퍼로 가져다 놓았을 것이므로 매우 빠르게 진행된다. AOCC 기법에서 어떤 트랜잭션이 매우 빠르게 실행되면, 그 트랜잭션의 실행동안에는 상대적으로 적은 수의 페이지가 무효화되므로 그의 완료 성공 가능성은 매우 높아진다. 그러나, DL 기법에서는 트랜잭션이 매우 빠르게 실행되더라도 결국에는 자신이 액세스한 데이타에 대하여 로크 승인을 받아야 한다. 이 과정에서 충돌되는 로크로 인하여 트랜잭션이 대기할 수도 있고 교착상태로 인하여 또 다시 철회될 수도 있다. 그러므로 트랜잭션 재실행시는 AOCC가 DL 기법보다 낮은 트랜잭션 철회율을 보일 것이다. 이와 같은 이유로 인해서 이 부하에서는 AOCC 기법과 DL 기법은 비슷한 철회율을 보인다. DL-ST/1 기법에서는 그림자 트랜잭션으로 인한 효과로 인하여 DL 기법보다는 훨씬 낮은 수준의 철회율을 보인다. 반면, AOCC-ST/1 기법은 AOCC 기법의 철회율을 줄이는데 그다지 크게 공헌하지 못한다.

5.3 실험 3- HOTCOLD 부하

이 부하에서 각 클라이언트는 데이타베이스에서 자신만이 자주 액세스하는 40 페이지의 HOT 영역의 가지고 있다고 가정한다. 한 클라이언트의 HOT 영역은 다른 클라이언트의 HOT 영역과는 완전히 분리되어 있고, 각 트랜잭션이 실행하는 연산의 80%는 자신의 HOT 영역을 액세스한다고 가정한다. 따라서, 이 부하는 매우 높은 참조 국부성과 낮은 데이타 충돌의 특징을

보인다. 이 실험은 버퍼 크기가 작더라도 높은 캐쉬 히트율을 보일 수 있기 때문에 Client-BufSize를 DbSize의 10%로 설정한다. 클라이언트 수에 따른 트랜잭션 처리율 및 철회율이 그림 9와 10에 있다. 이 부하에서 각 클라이언트는 자신만의 HOT 영역을 매우 높은 확률로 액세스하므로 각 클라이언트에 캐쉬된 데이타가 서로 겹칠 가능성이 다른 부하에 비해 매우 낮다. 이는 O2PL-I에서 완료 부담의 감소로 이어지고, DL, DL-ST/1, 그리고 AOCC 기법에서는 무효화된 데이타를 읽을 가능성의 감소로 나타난다. 그리고 이 기법들은 중복사본을 통신 부담없이 액세스할 수 있기 때문에 트랜잭션 실행

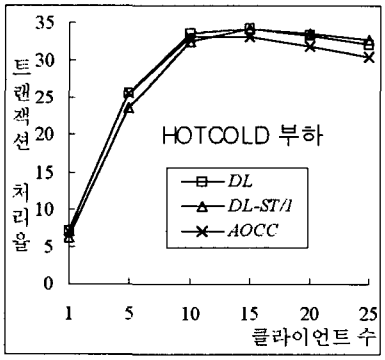

그림 9 트랜잭션 처리율

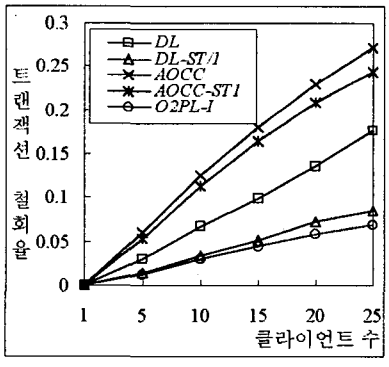

그림 10 트랜잭션 철회율

단계의 부담이 거의 비슷할 것이다. 그러므로 이들은 이 부하에서 비슷한 성능을 보인다. 그림 9의 그래프에는 생략되어 있지만, O2PL-I는 DL과 거의 동일한 성능을 보인다.

비록 각 기법들이 이 실험에서 거의 유사한 비슷한 성능을 보이지만 트랜잭션의 철회율에 있어서는 각 기법이 큰 차이를 보인다. 이와 같은 낮은 데이타 충돌 환경에서는 로크는 트랜잭션 철회를 줄이는데 큰 공헌을 한다. 따라서 데이타 충돌을 해결하기 위하여 로킹 기법을 사용하는 DL 기법은 AOCC 기법에 비해 낮은 트랜잭션 철회율을 보인다. DL-ST 기법은 O2PL-I와 거의 비슷한 수준으로 DL 기법의 철회율을 감소시키지만 AOCC-ST/1 기법은 AOCC 기법의 철회율을 10% 정도 감소시킬 뿐이다.

Ⅵ. 결론

검사기반의 TCCM 기법에서 비동기적으로 클라이언트 중복사본의 유효성을 검사하는 기법은 클라이언트와 서버사이에 메시지 부담이 작기 때문에 우수한 트랜잭션 처리율을 보인다. 그렇지만 비동기적 검사기반 기법에서는 트랜잭션이 유효하지 않은 중복사본을 액세스할 가능성이 있기 때문에 트랜잭션 철회율이 높은 단점이 있다. 본 논문에서는 이 단점을 해결하기 위하여 철회되는 트랜잭션 대신 실행되기 위하여 관리되는 백업 목적의 그림자 트랜잭션의 개념을 제안하고 이 개념과 DL의 로킹 기법을 통합하여 DL-ST 기법을 제안하였다.

본 논문에서는 모의실험을 통하여 DL-ST 기법의 성능을 DL, AOCC, 그리고 O2PL-I 기법의 성능과 비교하였다. DL-ST 기법은 낮은 데이타

충돌 환경에서는 그림자 트랜잭션을 생성하기 위한 실행 부담과 그의 결과를 저장하기 위한 공간 부담 때문에 DL 기법보다 낮은 성능을 보인다. 그러나 DL-ST 기법은 데이타 충돌이 심화되어 트랜잭션 철회가 빈번한 환경에서는 그림자 트랜잭션으로 인한 효과가 증대되어 DL 기법에 비해 매우 낮은 트랜잭션 철회율을 보인다. 따라서 데이타 충돌이 심한 환경에서 DL-ST 기법은 DL 기법보다 우수한 성능을 보인다. DL-ST 기법은 대부분의 부하환경에서 회피기반 기법중 가장 성능이 우수한 것으로 알려진 O2PL-I보다 더 높은 트랜잭션 처리율을 보이면서 비슷한 수준의 철회율을 보인다. 비록 AOCC 기법이 DL-ST와 유사한 수준을 성능을 보이지만, AOCC 기법은 데이타 충돌의 해결을 전적으로 트랜잭션 철회에 의존하므로 트랜잭션 철회율이 매우 높아 자원의 낭비가 심한 편이다. 그리고 AOCC 기법에 그림자 트랜잭션의 개념을 적용시키더라도 트랜잭션이 생성해 내는 중간 결과의 유효성을 보장하기 위한 방법이 없기 때문에 그림자 트랜잭션의 효과가 트랜잭션 처리율 및 철회율에 큰 영향을 미치지 않는다. 본 연구의 미래 연구 과제로서 비동기적으로 클라이언트 버퍼를 무효화시키는 회피기반 기법에 관한 연구가 진행될 것이다. 그리고 데이타전송과 질의전송(query-shipping)이 조합된 시스템을 위한 TCCM 기법에 대한 연구가 진행될 것이다.

참고문헌

[1] A. Adya, R. Gruber, B. Liskov, and U. Maheshwari, "Efficient Optimistic Concurrency Control Using Loosely Synchronized Clocks," Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 23-34, 1995.

[2] R. Agrawal, M. J. Carey, and M. Livny, "Concurrency Control Performance Modeling: Alternatives and Implications," ACM Trans. on Database Syst., Vol. 12, No. 4, pp. 609-654, 1987.

[3] T. E. Anderson, B. N. Bershad, E. D. Lazowska, and H. M. Levy, "Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism," ACM Trans. on Computer Syst., Vol. 10, No. 1, pp. 53-79, 1992.

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

[5] M. J. Carey, M. J. Franklin, M. Linvy, and E. J. Shekita, "Data Caching Tradeoffs in Client-Server DBMS Architectures," Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 357-366, 1991.

[6] M. J. Carey and M. Livny. "Conflict Detection Tradeoffs for Replicated Data," ACM Trans. on Database Syst., Vol. 16, No. 4, pp. 703-746, 1991.

[7] O. Deux et al, "The O2 System," Comm. of the ACM, Vol. 34, No. 10, pp. 34-48, 1991.

[8] M. J. Franklin and M. J. Carey, "Client-Server Caching Revisited," Technical Report #1089, Computer Sciences Department, University of Wisconsin-Madison, 1992.

[9] M. J. Franklin, M. J. Carey, and M. Livny, "Transactional Client-Server Cache Consistency: Alternatives and Performance," ACM Trans. on Database Syst., Vol. 22, No. 3, pp. 315-363, 1997.

[10] R. Gruber, "Optimism vs. Locking: A Study of Concurrency Control for Client-Server Object-Oriented Databases," PhD Thesis, MIT, 1997.

[11] W. Kim, J. F. Garza, N. Ballou, and D. Woelk, "The Architecture of the ORION Next-Generation Database System," IEEE Trans. on Knowledge and Data Eng., Vol. 2, No. 1, pp. 109-124, 1990.

[12] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb, "The ObjectStore Database System," Comm. of the ACM, Vol. 34, No. 10, pp. 50-63, 1991.

[13] A. M. Law and W. D. Kelton, Simulation Modeling & Analysis, McGraw-Hill, 1991.

[14] M. T. Özsu, K. Voruganti, and R. C. Unrau, "An Asynchronous Avoidance-Based Cache Consistency Algorithm for Client Caching DBMSs," Proc. Of Int. Conf. On VLDB, pp. 440-451, 1998.

[15] H. Schwetman, CSIM Users Guide for Use with CSIM Revision 16, Microelectronics and Computer Technology Corporation, 1992.

[16] M. Stonebraker, L. Rowe, B. Lindsay, J. Gray, M. Carey, M. Brodie, P. Bernstein, and D. Beech, "Third-Generation Data Base System Manifesto," ACM SIGMOD Record, Vol. 19, No. 3, pp. 31-44, 1990.

[17] Y. Wang and L. A. Rowe, "Cache Consistency and Concurrency Control in A Client/Server DBMS Architecture," Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 367-376, 1991.

[18] K. Wilkinson and M. Neimat, "Maintaining Consistency of Client Cached Data," Proc. Of Int. Conf. On VLDB, pp. 122-133, 1990.

[19] 권 혁민, "트랜잭션 캐쉬 일관성을 유지하기 위한 지연 로킹 기법의 성능 평가," 한국정보처리학회 논문지 제 7권 8호, pp. 2310-2326, 2000

Performance Evaluation of Deferred Locking With Shadow Transaction

Hyeok-Min Kwon*

Abstract

Client-server DBMS based on a data-shipping model can exploit client resources effectively by allowing inter-transaction caching. However, inter-transaction caching raises the need of transactional cache consistency maintenance(TCCM) protocol, since each client is able to cache a portion of the database dynamically. Detection-based TCCM schemes can reduce the message overhead required for cache consistency if they validate clients replica asynchronously, and thus they can show high throughput rates. However, they tend to show high ratios of transaction abort since transactions can access invalid replica. For coping with this drawback, this paper develops a new notion of shadow transaction, which is a backup-purpose one that is kept ready to replace an aborted transaction. This paper proposes a new detection-based TCCM scheme named DL-ST on the basis of the notion of shadow transaction. Using a simulation model, this paper evaluates the effect of shadow transaction in terms of transaction through rate and abort ratio.

* Dept. of software Semyung Univ.