MOERS : 그룹 통신의 메시징 순서를 확장한 즉시 갱신 중복 기법

(MOERS: An Eager Replication Scheme using Extended Messaging Order of Group Communication)

문 애 경 † 남 궁 한 †† 조 행 래 †††

(Aekyung Moon) (Han Namgoong) (Haengrae Cho)

요 약 기존에 제안된 대부분의 중복 기법은 원본 트랜잭션을 완료한 후 비동기적으로 중복사본에 대한 갱신 요구를 방송하는 지연 갱신 기법을 가정하였다. 지연 갱신 기법은 즉시 갱신 기법에서 발생하는 빈번한 교착상태 발생 문제를 해결할 수는 있지만 데이타 일관성 유지가 사용자 책임이라는 단점을 갖는다. 최근 그룹 통신의 메시징 순서를 적용한 즉시 갱신 중복 기법들이 제안되고 있다. 이들 기법은 메시징 순서를 이용하여 교착상태 발생률을 줄였지만, 송신 노드는 갱신 요구 메시지를 방송한 후 전역 직렬성을 검증하는 낙관적 기법을 채택하기 때문에 동시성이 증가할수록 철회 트랜잭션의 실행 오버헤드가 증가한다는 문제점을 갖는다. 본 논문에서는 철회 트랜잭션의 갱신 메시지 방송과 실행 오버헤드를 줄일 수 있는 즉시 갱신 중복 기법을 제안한다. 제안한 기법은 갱신 요구 메시지를 방송하기 전에 전역 직렬성 검사가 이루어지기 때문에 완료 트랜잭션은 한번의 메시지 방송으로 처리할 수 있다. 뿐만 아니라, 철회 트랜잭션은 다른 노드로 방송할 필요가 없기 때문에 메시지 전송 횟수를 줄일 수 있으며, 철회 트랜잭션의 실행으로 인한 디스크 액세스 수와 로크 대기 시간을 줄임으로써 성능을 향상시킨다.

키워드 : 분산 데이타베이스, 중복 데이타 관리, 트랜잭션 처리, 그룹 통신

Abstract Most of previous replication schemes assume lazy schemes, which asynchronously propagate replica updates to other nodes after the updating transaction commits. While lazy schemes are novel in the sense that they can resolve high deadlock frequencies of eager schemes, they suffer from the fact that users should be responsible for the data consistency. Recently replication schemes based on the messaging order of group communication have been studied. They can reduce the deadlock frequencies using the messaging order, but they have another problem of increasing execution overhead of aborted transactions at high degree of concurrency. This is because the sender node validates global serializability after multicasting replica update messages to other nodes. In this paper, we propose an efficient eager replication scheme for reducing the number of messaging and overhead of aborted transactions significantly. The proposed scheme validates any conflicts before multicasting and does not multicast when there is any conflict; hence, it can exploit performance improvement by alleviating the message traffic for replica update or by reducing the number of disk accesses and locking delay due to aborted transactions.

Key words : Distributed Database, Replicated Data Management, Transaction Processing, Group Communication

† 비 회 원 : 한국전자통신연구원 이동분산처리팀 선임연구원
akmoon@etri.re.kr
†† 비 회 원 : 한국전자통신연구원 책임연구원
nghan@etri.re.kr
††† 종신회원 : 영남대학교 전자정보공학부 교수
hrcho@ynucc.yeongnam.ac.kr
논문접수 : 2002년 3월 12일
심사완료 : 2002년 9월 6일

1. 서 론

분산 시스템 환경에서 가용성, 신뢰성, 그리고 고성능 트랜잭션 처리를 위하여 각 노드들은 중복사본을 갖는다. 중복사본의 일관성을 유지하기 위한 중복 기법은 "즉시 갱신(eager update)"과 "지연 갱신(lazy update)"

의 두 가지 기법으로 나눌 수 있다[1]. 즉시 갱신은 원본 트랜잭션과 다른 노드의 중복사본에 대한 갱신 요구 트랜잭션이 하나의 트랜잭션으로 구성되는 것으로, 원본 트랜잭션이 완료하기 전에 전역 직렬성 검사가 이루어지므로 데이타 일관성이 유지된다. 이와는 달리 지연 갱신은 원본 트랜잭션과 갱신 요구 트랜잭션이 비동기적으로 처리되므로 원본 트랜잭션의 완료 후 전역 직렬성 검사가 이루어진다. Gray[1]가 빈번한 교착상태 발생을 즉시 갱신의 문제로 지적한 이후 대부분의 중복 기법은 지연 갱신을 가정하고 있다[2,3,4,5]. 지연 갱신은 즉시 갱신보다 교착상태 발생률이 낮다는 장점을 갖지만 전역 직렬성이 위반된 트랜잭션이 완료되는 경우가 발생할 수 있고 데이타 일관성 유지가 사용자 책임이라는 단점을 갖는다[1,6].

최근 그룹 통신 개념이 대두되면서 즉시 갱신 중복 기법에 대한 연구가 진행되고 있다[7,8,9,10,11]. Holliday[8]는 그룹 통신 분야에서 많은 연구가 진행된 메시징 순서 기능을 이용하여 트랜잭션의 순서를 미리 정하는 것이 교착상태 해결에 효과적임을 제안하였다. Kemme와 Alonso[9,10,11]도 메시징 기능을 이용한 중복 기법을 제안하였다. 이 기법에서 송신 노드는 판독 연산을 실행한 후 갱신 요구 메시지를 모든 노드에 방송하고 수신된 메시지를 바탕으로 전역 직렬성을 검사한 후 그 결과를 다시 방송한다. 즉, 직렬성 위반 여부가 갱신 요구 메시지 방송 이후 결정되는 낙관적 기법이기 때문에 송신 노드는 철회될 트랜잭션에 대한 갱신 요구 방송 오버헤드, 수신 노드는 그 트랜잭션의 실행 오버헤드를 갖는다. 일반적으로 낙관적 기법을 이용한 트랜잭션 실행의 경우, 비관적 기법의 경우에 비해 트랜잭션 철회율이 훨씬 높기 때문에 철회될 트랜잭션으로 인한 오버헤드를 줄이는 것은 시스템의 성능에 매우 중요한 영향을 미친다. 따라서 본 논문은 철회될 트랜잭션의 갱신 요구 메시지 방송 오버헤드와 실행 오버헤드를 줄이는 것을 기본 개념으로 하는 MOERS (Messaging Order based Eager Replication Scheme)를 제안한다.

MOERS에서 송신 노드는 트랜잭션의 판독 연산을 실행한 후, 갱신 요구 메시지를 방송하기 전에 전역 직렬성을 검사한다. 전역 직렬성 검사는 송신 노드에서 판독한 데이타의 유효성으로 결정되며, 유효한 경우 갱신 요구 메시지를 방송한다. 그러나, 이미 무효화된 데이타를 판독한 경우는 갱신 요구를 방송하지 않는다. 즉, Kemme와 Alonso의 경우와 달리 MOERS는 갱신 요구 메시지 방송 전에 전역 직렬성 검사가 이루어지며, 완료 트랜잭션은 한번의 메시지 방송으로 처리되고 철회 트랜잭션은 다른 노드로 방송하지 않는다. 그 결과, MOERS는 메시지 전송 횟수를 줄이고 철회 트랜잭션 실행으로 인한 디스크 액세스 수와 로크 대기 시간을 줄임으로써 성능을 향상시킨다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구에 대해 살펴보고, 3절에서는 본 논문에서 제안하는 중복 기법인 MOERS를 설명한다. 4절에서는 성능 평가를 위한 그룹 통신 시스템 모형 및 트랜잭션 처리 모형을 제안하고 성능 분석을 위해 채택된 매개 변수를 설명한다. 5절에서는 제안된 성능 평가 모형을 바탕으로 수행한 성능 평가 결과를 비교하고, 끝으로 6절에서 결론을 맺는다.

2. 관련 연구

그룹 통신은 시스템 장애가 발생하는 경우에도 메시지 전송을 보장하는 통신 기법으로 메시징 순서 기능을 제공한다[9,12]. 메시징 순서 기능은 하나의 노드가 여러 개 메시지를 방송하는 경우, 모든 노드는 미리 정해진 순서에 따라 메시지를 수신함을 보장하는 것이다. 예를 들면, 두 노드 N_1과 N_2가 다른 노드로부터 메시지 m과 m′를 전송 받는다면, N_1과 N_2의 메시지 실행 순서는 모두 m → m′ 혹은 m′ → m임을 보장한다. 이러한 메시징 순서 기능은 트랜잭션의 순서를 미리 정하는 것과 같은 역할을 하므로 교착상태 해결에 용이하다. 이를 이용한 중복 기법으로는 Holliday의 BA (Broadcast All)[7]와 Kemme과 Alonso의 SER(Replication with Serializability)[9,10,11] 등을 들 수 있다.

BA의 경우 송신 노드는 트랜잭션의 전체 내용을 중복사본이 저장된 모든 노드에 방송하고 수신 노드는 메시징 순서에 따라 트랜잭션을 실행한다. BA의 주요 단점은 판독 연산도 모든 노드에서 실행해야 한다는 것이다. 트랜잭션 수를 G_n, 트랜잭션 당 연산수를 T_n이라고 가정하면 각 노드에서 실행해야하는 총 연산수는 $<G_n \times T_n>$이 된다. 따라서 BA는 트랜잭션 수 및 트랜잭션 당 연산수에 많은 영향을 받고, 그 결과 동시에 실행하는 트랜잭션 수가 증가하거나 길이가 긴 트랜잭션인 경우 급격한 성능 저하를 초래할 수 있다는 단점을 갖는다. BA의 또 다른 단점은 이질형 시스템에는 적용이 어렵다는 점이다. 모든 수신 노드는 동일한 트랜잭션을 실행하기 때문에 데이타 일관성을 유지하기 위해서 동일한 지역 교착상태 해결 기법을 이용해야 한다. 따라서 다른 지역 교착상태 해결 기법을 사용하는 시스템에는 적용이 어렵다.

SER은 BA와 달리 판독 연산을 지역적으로 처리하고

갱신 연산은 중복사본이 저장된 모든 노드에서 실행한다. 그리고 메시지 전송 오버헤드를 줄이기 위하여 갱신 연산을 하나씩 방송하는 것이 아니라 판독 연산 실행을 마친 후 갱신 연산을 한꺼번에 방송하며, 전송된 갱신 요구에 대한 로크를 원자적으로 요청함으로써 교착상태를 방지한다. 트랜잭션 T_1의 발생 노드가 N_1이면, N_1은 먼저 S 로크를 획득하고 판독 연산을 실행한다. 갱신 연산의 실행은 모든 판독 연산의 실행을 마칠 때까지 연기한다. N_1은 T_1의 판독 연산을 모두 실행한 후 갱신 요구 메시지를 모든 노드에 방송하고, 수신 노드들은 메시지에 포함된 갱신 데이타들에 대해 원자적으로 X 로크를 획득하고 실행한다. 이와 달리 송신 노드 N_1은 자신이 방송한 갱신 요구 메시지 이외의 메시지가 먼저 도착하면 이미 판독한 데이타의 유효성을 검사한다. 만약 T_1의 판독 데이타가 먼저 도착한 메시지의 갱신 데이타와 충돌된다면 모든 노드에 T_1의 철회 메시지를 방송하고 T_1을 철회한다.

SER은 다음과 같은 문제점을 갖는다. 첫째, 전역 직렬성을 보장하기 위하여 로크 관리자를 수정해야 한다. 갱신 요구 메시지를 전송 받으면 노드들은 X 로크를 획득한다. X 로크를 획득하는 과정에서 충돌하는 S 로크가 존재하고 그 로크를 획득한 트랜잭션의 갱신 요구 메시지가 아직 도착하지 않았다면, 무효화된 데이타를 판독한 것으로 전역 직렬성 결과는 철회이다. 따라서 SER을 적용하기 위해서는 S-X 로크 충돌 발생시 판독 데이타의 유효성 검사 기능을 로크 관리자에 추가해야 한다. 둘째, 송신 노드는 전역 직렬성 결과를 방송하고 수신 노드들은 그 결과가 도착할 때까지 갱신 연산에 대한 로크 해제를 지연해야 한다. 예를 들면, 송신 노드 N_1은 T_1의 갱신 요구 메시지를 방송하고 수신 노드는 전송 받은 갱신 연산을 실행한다. 이후 N_1은 T_1의 판독 데이타 유효성 검사를 통한 전역 직렬성 결과를 방송하고, 수신 노드는 N_1으로부터 그 결과(완료 혹은 철회)를 전송 받으면 T_1의 로크를 해제한다. 따라서 송신 노드는 전역 직렬성 결과 방송에 따른 추가적인 메시지 전송 오버헤드를 갖고 수신 노드는 갱신 연산을 모두 실행했더라도 로크 해제를 지연해야 한다. 마지막으로, 수신 노드는 경우에 따라 철회될 트랜잭션을 실행할 수 있다. 앞의 예에서, 수신 노드는 전송 받은 T_1의 갱신 연산을 실행하다가 철회 메시지가 도착하면 실행 취소하고 로크를 해제한다. 즉, 수신 노드들은 T_1이 철회될지를 사전에 알지 못하므로 송신 노드에서 철회 메시지가 도착하기 전에는 T_1을 실행하기 위하여 X 로크를 보유해야 한다. 따라서 철회 트랜잭션에 대한 불필요한 로크 액세스로 인하여 정상적인 트랜잭션 실행이 지연될 수 있다.

본 논문에서는 갱신 연산만 방송함으로써 BA의 단점을 해결한 SER과 동일한 정책을 채택하지만, 갱신 요구 메시지를 방송하기 전에 전역 직렬성 검사를 수행함으로써 SER에서 발생하는 추가적인 메시지 전송 오버헤드 및 철회 트랜잭션 실행 오버헤드를 줄일 수 있는 MOERS를 제안한다.

3. 즉시 갱신 중복 기법

MOERS의 기본 개념을 이해하기 위하여 먼저 기존의 SER에서 발생하는 불필요한 갱신 요구 메시지 방송에 대한 문제점을 이해하여야 한다. 이를 위해 아래 [예 1]을 살펴보자.

[예 1] 노드 N_1과 N_2가 데이타 a와 b를 중복 저장한다고 가정하자. 트랜잭션 T_1과 T_2의 내용은 아래와 같고 N_1은 T_1, N_2는 T_2를 각각 실행한다.

T_1 : $R_1(a)$ $W_1(b)$

T_2 : $R_2(b)$ $W_2(a)$

전역 직렬성을 만족하는 실행 스케줄($T_1 \rightarrow T_2$ 혹은 $T_2 \rightarrow T_1$)을 보장하기 위해서 SER을 적용하면 다음과 같다. 노드 N_1은 a의 S 로크를 획득하고 판독 연산을 실행한 후, T_1의 갱신 요구 메시지 M_1: $<W_1(b)>$을 방송한다. 마찬가지로 노드 N_2도 b의 S 로크를 획득하고 판독 연산을 실행한 후, T_2의 갱신 요구 메시지 M_2: $<W_2(a)>$를 방송한다. 단, 메시징 순서는 $M_1 \rightarrow M_2$라고 가정한다. 수신 노드는 갱신 요구 메시지를 받으면 갱신 데이타의 X 로크를 원자적으로 요청한다. 노드 N_1은 M_1을 수신하면, b에 대한 X 로크를 획득하고 T_1의 완료 메시지를 방송한다. 반면에 노드 N_2는 M_1의 X 로크를 획득하는 과정에서, T_2에 의해 설정된 b의 S 로크와 충돌이 발생하고 아직 S 로크를 획득한 트랜잭션의 갱신 요구 메시지 M_2가 도착하지 않았기 때문에 모든 노드에 철회 메시지(A_2)를 방송한다. 그 결과 모든 노드에서 트랜잭션 T_1은 실행되고 T_2는 철회됨으로써 전역 직렬성이 보장된다. 그러나 SER의 경우 T_2가 철회될 지를 사전에 알지 못하므로 N_2는 M_2를 모든 노드에 방송하고 N_1은 M_2가 도착하면 $W_2(a)$를 처리한다. 이로 인해 N_1에서 다른 트랜잭션이 데이타 a를 액세스하고자 하면 로크가 해제될 때까지 지연된다. N_1은 N_2로부터 A_2가 도착하면 T_2를 실행취소하고 로크를 해제한다. [예 1] 끝

일반적으로 SER과 같이 낙관적 기법을 이용하여 트랜잭션을 실행하는 경우, 비관적 기법을 이용하는 경우에 비해 트랜잭션 철회율이 훨씬 높기 때문에 철회될 트랜

잭션의 갱신 요구 메시지 방송 오버헤드와 실행 오버헤드를 줄이는 것은 시스템의 성능에 매우 중요한 영향을 미친다. 결국 MOERS의 주요 내용은 낙관적 기법을 채택할 때 철회 트랜잭션으로 인한 성능 하락을 방지하기 위하여, 다른 노드로 메시지를 방송하기 전에 해당 트랜잭션이 판독한 데이타가 유효한 데이타인지 아닌지를 판단하는 전역 직렬성 검사 방법을 개발하는 것이다.

3.1 자료 구조 및 가정

본 절에서는 MOERS에 필요한 기본적인 자료 구조 및 가정을 설명한다. 그룹 통신에 대한 대부분의 자료 구조 및 가정은 [13]와 [14]에서 채택된 내용들이다.

그룹 통신 관리자(Group Communication Manager: GCM)는 메시징 순서와 전역 직렬성 검사 역할을 담당하는 것으로 Amoeba 그룹 통신[15]과 같이 전체 시스템에 하나의 Sequencer가 존재하는 집중형 구조를 가정한다. 그 이유는 분산형 구조의 경우 메시징 순서 기능 구현이 너무 복잡하여 성능 저하를 초래할 수도 있음이 이미 언급된 바 있고[15], 이로 인하여 ISIS는 분산형 메시징 기능을 동적-집중형 구조로 변경한 경우가 있기 때문이다[16]. Sequencer는 메시징 순서 기능을 제공하기 위하여 메시지 일련 번호(MSN: Message Sequence Number)를 부여하는 역할을 담당한다. 각 트랜잭션은 액세스하는 데이타 식별자와 액세스 유형(판독 혹은 갱신)의 리스트로 표현되고, 데이타 식별자는 데이타가 포함된 페이지 번호와 페이지 내에서 해당 데이타의 슬롯 번호로 구성된다. 각 노드는 트랜잭션의 판독 연산을 실행한 후 갱신 연산을 방송하기 위하여 GCM에게 MSN를 요청하고, GCM은 Sequencer를 통하여 MSN을 부여하게 된다. 이때 MSN은 나중에 생성한 MSN이 기존의 MSN 보다 큰 값을 갖는 단조 증가 함수로 구현된다. MaxMSN은 GCM이 현재까지 할당한 가장 큰 MSN을 나타낸다. GCM이 관리하는 정보로는 "갱신 정보 테이블(U-TBL)"이 있으며, 이는 전역 직렬성 검사 기능을 위해 사용된다.

U-TBL은 아직 모든 노드에서 반영되지 않은 데이타 식별자와 해당 데이타의 갱신 시점을 기록한 것으로 <갱신 데이타, MSN>의 쌍으로 구성된다. 이때 MSN은 해당 데이타의 변경 시점을 의미한다. 노드 N_i는 트랜잭션 T_k의 판독 연산을 실행한 후 <판독 데이타 집합(RS_k), 갱신 데이타 집합(WS_k), LastMSN(N_i)> 정보를 포함한 MSN 요청 메시지를 GCM에 전송한다. LastMSN(N_i)는 노드 N_i에서 마지막으로 실행을 완료한 갱신 요구 메시지의 MSN을 의미한다. U-TBL은 RS_k의 유효성을 판단하기 위해 사용되는데 자세한 알고리즘은 3.2절에 설명한다. RS_k가 유효한 것으로 판단되면 새로운 MSN이 부여되고 WS_k와 함께 U-TBL에 기록된다. 메시지 전송 부담을 줄이기 위하여 RS_k와 WS_k는 데이타의 식별자로만 구성되고 MSN을 할당받은 노드에서 실제 갱신 데이타를 방송한다.

3.2 MOERS 알고리즘

MOERS에서 각 노드는 트랜잭션의 판독 연산을 진행하고 갱신 요구를 방송하기 전에 전역 직렬성을 검사한다. MOERS의 구체적인 알고리즘은 다음과 같다.

단계 1: 송신 노드 N_i는 트랜잭션 T_k의 판독 연산을 실행하기 위하여 S 로크를 요구한다. 모든 판독 연산을 실행한 후, <RS_k, WS_k, LastMSN(N_i)> 정보를 갖는 MSN 요청 메시지를 GCM에게 전송한다. RS_k는 T_k가 판독한 데이타의 식별자들로 구성되고, WS_k는 T_k가 갱신할 데이타의 식별자들로 구성된다.

단계 2: GCM은 $d_r \in RS_k$인 모든 d_r에 대해, LastMSN(N_i)와 U-TBL에 저장된 <d_r, MSN(d_r)>를 이용하여 d_r의 유효성을 먼저 검사한다. 단, U-TBL에서 MSN(d_r)은 d_r을 마지막으로 갱신한 트랜잭션의 MSN이며, U-TBL에 d_r의 갱신 정보가 없는 경우 MSN(d_r)은 $\varnothing$의 값을 갖는다고 가정한다.

(1) MSN(d_r) = $\varnothing$: U-TBL에 d_r의 갱신 기록이 없다는 것은 d_r이 최근에 갱신된 적이 없다는 것을 의미하므로 N_i가 판독한 데이타 d_r은 유효하다.

(2) LastMSN(N_i) ≥ MSN(d_r): d_r이 최근에 갱신된 적이 있고 N_i는 MSN(d_r) 시점에서 발생한 d_r의 갱신 요구를 이미 실행하였으므로 T_k가 판독한 데이타 d_r은 유효하다.

(3) LastMSN(N_i) < MSN(d_r): d_r이 최근에 갱신된 적이 있지만 N_i는 MSN(d_r) 시점에서 발생한 d_r의 갱신 요구를 아직 실행하지 않았음을 의미한다. 즉, N_i는 갱신 전의 d_r을 판독하였고, 그 결과 N_i가 판독한 데이타 d_r은 유효하지 않다.

단계 3: GCM은 $d_r \in RS_k$인 모든 d_r에 대한 유효성 검사 결과가 (1)이나 (2)에 해당된다면 MaxMSN 값에 1 증가한 새로운 MSN을 노드 N_i에게 할당하여 전송한다. 그리고 $d_w \in WS_k$인 모든 d_w에 대해 <d_w, 새로운 MSN> 쌍을 U-TBL에 저장한다. 이와는 달리 유효성 검사 결과가 (3)에 해당되면 GCM은 T_k를 철회로 결정하고 철회 메시지를 송신 노드 N_i에게 전송한다.

단계 4: 송신 노드 N_i는 GCM으로부터 전역 직렬성 검사 결과에 따라 MSN을 할당받거나 철회 메시지를 전송 받는다.

(1) MSN이 할당된 경우, 갱신 요구 메시지 <WS_k,

MSN>를 모든 노드에게 방송한다. 이때 WS_k는 각 노드의 중복사본에 갱신되어야 하는 실제 데이타 값을 저장한다. 수신 노드는 WS_k를 전송받으면 $d_w \in WS_k$인 모든 d_w에 대해 X 로크를 원자적으로 설정한 후, 전송받은 d_w 값을 기록한다. 만약 d_w에 대하여 다른 트랜잭션에 의한 S 로크 혹은 X 로크가 이미 설정되어 있으면 대기한다. 모든 d_w에 대한 기록을 완료한 후 X 로크를 해제하고 해당 노드의 LastMSN은 WS_k의 MSN으로 변경된다.

(2) 철회 메시지를 전송받은 경우, T_k를 철회하고 S 로크를 해제한다. ■

다음 [예 2]는 [예 1]의 트랜잭션에 대해 MOERS에서 수행하는 전역 직렬성 검사 과정을 보여준다.

[예 2] 노드 N_1과 N_2의 LastMSN과 GCM의 MaxMSN은 모두 1로 초기화되어 있다고 가정하자. 그리고, N_1은 트랜잭션 T_1, N_2는 트랜잭션 T_2를 각각 실행한다. 먼저 N_1은 a를 판독한 후 GCM에게 MSN 요청 메시지 M_1: <RS_1={a}, WS_1={b}, 1>을 전송한다. N_2도 b를 판독한 후 GCM에게 MSN 요청 메시지 M_2: <RS_2={b}, WS_2={a}, 1>를 전송한다. GCM의 메시지 큐에는 M_1, M_2 순으로 저장된다. GCM은 먼저 U-TBL에서 M_1의 <RS_1={a}>와 충돌되는 데이타가 있는지 검사한다. U-TBL의 MSN(a)는 ∅으로 초기화되어 있으므로, T_1이 판독한 데이타 a는 유효하다. 따라서 GCM은 MaxMSN에 1을 더한 새로운 MSN(=2)을 할당하여 N_1에게 전송하고 U-TBL에 <b, MSN(=2)>을 저장한 후 M_2를 처리한다. M_1을 처리할 때와 마찬가지로 MSN을 할당하기 전에 M_2가 판독한 b의 유효성을 검사한다. 즉, U-TBL에서 M_2의 <RS_2={b}>와 충돌되는 데이타가 있는지 검사하는데, LastMSN(N_2)(=1) < MSN(b)(=2)이므로 단계 2-(3)에 해당된다. 이는 N_2가 $W_1(b)$를 아직 실행하지 않았음을 의미하는 것으로 T_2가 판독한 b는 유효하지 않고, 그 결과 GCM은 T_2에 대한 철회 메시지를 N_2에 전송한다.

만약 N_2가 N_1에서 요청한 T_1의 $W_1(b)$를 실행한 후 T_2를 실행하면 위의 경우와 다른 결과가 발생한다. 즉, $W_1(b)$의 MSN이 2이므로 N_2의 LastMSN은 2가 된다. T_2의 판독 연산을 실행한 후 N_2는 GCM에게 MSN 요청 메시지 M_2: <RS_2={b}, WS_2={a}, 2>를 전송하고, GCM은 먼저 U-TBL을 이용하여 b의 유효성을 검사한다. 이때 LastMSN(N_2) ≥ MSN(b)이므로 단계 2-(2)에 해당하고, 그 결과 N_2에서 판독한 b는 유효하다. 따라서 GCM은 N_2에게 새로운 MSN(=3)을 할당하여 전송한다. [예 2] 끝

3.3 MOERS 알고리즘의 오버헤드 분석

SER이나 BA와 달리 MOERS에서 GCM은 전역 직렬성 검사를 위한 추가적인 오버헤드를 갖는데, 이는 U-TBL의 관리 오버헤드와 U-TBL내에서 MSN을 검사하는 오버헤드이다. 본 절에서는 각각의 오버헤드와 이에 대한 해결 방안을 설명하도록 한다.

U-TBL은 각 노드에서 발생한 갱신 정보를 기록하는 것으로 갱신 데이타와 MSN의 크기를 각 4바이트라고 가정한다면 1Kbyte 크기에 2^7개의 갱신 정보가 기록될 수 있다. U-TBL이 차지하는 정보의 양은 작지만 한정된 공간에 저장되기 때문에 매우 커질 경우 문제가 발생할 수 있다. 이에 대한 대안으로 GCM은 "안정 정보 테이블(S-TBL)"이라는 추가적인 정보와 해싱 검색 방법을 이용한다. S-TBL은 각 노드에서 마지막으로 실행한 갱신 요구 메시지의 MSN을 기록한 것으로 <노드 식별자, MSN>의 고정 크기를 갖는다. 노드 N_i가 트랜잭션 T_k의 판독 연산을 실행한 후 MSN 요청 메시지 <RS_k, WS_k, LastMSN(N_i)>를 GCM에 전송하면 S-TBL에 저장된 N_i의 MSN 값을 LastMSN(N_i)로 변경한다. 그리고 S-TBL에 저장된 가장 최소 MSN을 구하여 그 이하 MSN을 갖는 갱신 정보는 모든 노드에서 실행되었기 때문에 U-TBL에서 삭제한다. 본 논문은 주기적으로 S-TBL에서 가장 최소 MSN을 계산하고 U-TBL에서 해당 정보를 삭제한다고 가정하여 3.2절의 알고리즘에는 이 과정을 나타내지 않았다.

다음으로, MOERS 알고리즘의 단계 2의 전역 직렬성 검사에서 발생하는 가장 큰 오버헤드는 $d_r \in RS_k$인 모든 d_r에 대해 U-TBL에 저장된 <d_r, MSN(d_r)>을 검색하는 것이다. 이에 대한 해결 방안은 d_r의 해싱 값을 키로 사용하여 U-TBL을 해쉬 테이블의 형태로 구성하는 것이다. 해싱을 이용한 검색 시간은 해쉬 충돌 확률에 따라 결정되므로, S-TBL을 이용하여 U-TBL에 저장된 데이타 양을 줄이고 해쉬 테이블의 크기를 충분히 크게 한다면 d_r을 이용한 MSN의 검색은 대부분의 경우 해쉬 충돌없이 해결할 수 있다.

고려해야 할 또 다른 점은 MOERS를 분산형 구조에 적용하는 경우 다른 기법과 달리 수정이 필요하다는 것이다. 기존의 분산형 메시징 순서 기능에 대한 연구는 Totem과 같이 토큰을 이용하는 방식[17]과 특정 수신 노드에서 일방적으로 MSN을 방송하는 방식[16]이 있다. 이들 분산형 구조는 집중형 구조와 달리 송신 노드들은 중앙의 서버에 MSN을 요청하는 것이 아니기 때문에 U-TBL의 위치와 관리가 문제가 된다. Totem과 같은 토큰 방식에는 U-TBL을 토큰 정보에 포함시키고,

S-TBL을 이용한 불필요한 갱신 정보 삭제 주기를 짧게 하여 토큰 크기를 관리한다. Totem의 경우는 흐름 제어의 이유로 방송되는 메시지 수를 제한하기 때문에[14] 자체적으로 U-TBL의 크기가 조절되어 MOERS의 적용이 용이하다. 특정 수신 노드에서 일방적으로 MSN을 방송하는 방식은 MSN 할당 노드에 U-TBL을 저장한다. 그리고 메시지의 MSN을 방송할 때 U-TBL을 이용한 전역 직렬성 검사 결과를 함께 방송한다. 이 방식에서 수신 노드들은 갱신 요구 메시지를 받기는 하였지만 MSN 도착 전에는 수행하지 않기 때문에 철회 트랜잭션 실행 오버헤드는 줄일 수 있다. 그러나 송신 노드는 무조건 갱신 요구 메시지를 방송하기 때문에 SER과 마찬가지로 불필요한 메시지 방송 오버헤드를 갖고, MSN을 할당하는 노드가 동적으로 변경된다면 U-TBL의 관리가 어려워진다.

3.4 정성적인 성능 비교

본 절에서는 BA와 SER, 그리고 MOERS에서 액세스되는 데이타 수를 통한 정성적인 성능 비교를 한다. 단, 데이타는 모든 노드에 완전히 중복되어 있다고 가정하고, *wpct*와 *tr_length*, 그리고 #N은 각각 갱신 연산 비율, 트랜잭션이 액세스하는 데이타 수, 그리고 노드 수를 나타낸다. 노드마다 하나의 트랜잭션이 발생하고 모든 트랜잭션의 *tr_length*가 동일하다고 가정하면, 각 노드에서 동시에 실행되는 최대 트랜잭션 수는 #N개이다.

BA의 경우, 노드 N_i는 모든 노드에서 발생한 트랜잭션의 갱신 연산 및 판독 연산을 실행해야 하므로, N_i에서 액세스되는 최대 데이타 수 ($\#D(N_i)$)는 다음과 같이 계산할 수 있다.

$$\#D(N_i) = tr\_length \times \#N \quad (1)$$

SER과 MOERS의 경우에는 다른 노드에서 발생한 트랜잭션에 대해서는 갱신 연산만 실행하면 되므로, $\#D(N_i)$는 다음과 같다.

$$\#D(N_i) = tr\_length + tr\_length \times wpct \times (\#N-1) \quad (2)$$

일반적으로 *wpct*는 1보다 작으므로 BA의 $\#D(N_i)$ 값이 가장 크다는 것을 알 수 있다. $\#D(N_i)$가 클 경우 액세스되는 데이타 수가 증가하고 데이타간 충돌 확률이 높아지므로, BA의 로크 대기 시간이 가장 길것으로 예상된다.

식 (2)는 모든 트랜잭션이 철회없이 실행된다는 가정에서만 동작하며, 트랜잭션이 철회될 경우를 고려하면 MOERS와 SER에서의 $\#D(N_i)$ 값이 서로 달라진다. SER의 경우, 노드 N_i는 다른 노드에서 전송된 트랜잭션 T_k에 대해 철회 결정 메시지가 도착하기 전에는 갱신 연산을 실행한다. 만약 T_k의 갱신 연산을 모두 실행한 후 T_k가 철회된다면 *tr_length*×*wpct* 만큼의 데이타 액세스에 대한 오버헤드가 발생한다. 최악의 경우, N_i를 제외한 모든 노드에서 N_i에게 요청한 트랜잭션들의 갱신 연산이 완전히 실행된 후 각 트랜잭션의 철회 메시지가 도착한다면 *tr_length*×*wpct*×(#N−1) 만큼의 데이타 액세스 오버헤드가 발생한다. 이와는 달리 MOERS는 GCM을 통하여 전역 직렬성을 검사한 후 철회될 트랜잭션에 대해서는 갱신 연산을 방송하지 않으므로, 수신 노드에서 철회 트랜잭션에 의한 데이타 액세스 오버헤드가 발생하지 않는다. 그 결과 MOERS는 철회 트랜잭션으로 인한 로크 지연을 방지할 수 있고 전체적인 트랜잭션 응답 시간을 줄일 수 있을 것으로 예상된다. 특히 동시에 실행되는 트랜잭션 수가 증가하거나 데이타 충돌 확률이 높을 경우 MOERS는 SER보다 우수한 성능을 보일 수 있다.

4. 성능 평가 모형

본 절에서는 MOERS의 성능을 기존 기법과 비교하기 위한 성능 평가 모형을 설명한다. 모의 실험을 위해서 본 논문에서 개발한 중복사본을 갖는 분산 시스템 환경은 (그림 1)에 나타난다. 모의 실험은 미국의 MCC에서 개발한 CSIM 언어[18]를 이용하여 수행되었다.

그림 1에서 각각의 노드와 GCM은 네트워크를 통해 연결되어 있고 각 노드는 데이타베이스를 중복 저장하고 있다. 노드의 자원 관리자는 자신에게 할당된 CPU 작업을 모델링하며, 네트워크를 통한 GCM과의 통신 및 노드간 갱신 데이타 전송 과정을 수행한다. 트랜잭션 생성기는 트랜잭션을 차례대로 생성하는 역할을 수행한다. 하나의 트랜잭션이 완료된 후, 트랜잭션 생성기는 일정 기간 동안 대기한 후 다음 트랜잭션을 생성한다. 각 트랜잭션의 실질적인 실행(로크 요청 및 MSN 요청)은 트랜잭션 관리자에 의해 수행된다. 로크 관리자는 로크 관리를 위하여 레코드 단위의 두 단계 로킹 기법을 지원한다.

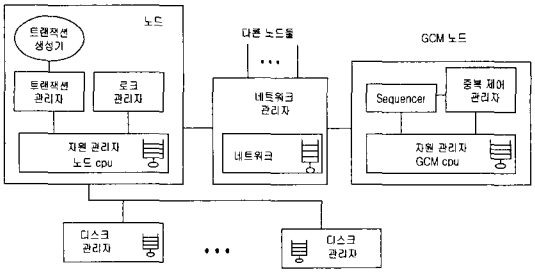

그림 1 성능 평가 모형

GCM은 MSN 할당 및 전역 직렬성을 검사하는 역할

을 담당하며, 이를 위해 Sequencer와 중복 제어 관리자를 갖는다. Sequencer는 메시징 순서를 보장하기 위하여 MSN을 부여하는 역할을 담당한다. 중복 제어 관리자는 노드의 MSN 요청 메시지 내용과 GCM에 저장된 U-TBL의 정보를 이용하여 각 노드에서 실행한 트랜잭션의 전역 직렬성을 검증하는 역할을 담당한다. BA와 SER은 GCM의 MSN 할당 기능만 필요하고, SER의 경우는 로크 관리자가 전역 직렬성 검사를 수행하며 BA의 경우는 로크 관리자에서 교착 상태 해결을 위하여 대기 그래프에 바탕을 둔 지역 교착 상태 탐지 및 해결 기법을 지원한다.

본 논문에서 사용한 입력 매개 변수를 표 1에 요약한다. 각 매개 변수의 구체적인 값은 [13]와 [14]에서 많이 참조하였다.

표 1 입력 매개 변수

시스템 구성 변수		
LCPUSpeed	노드 CPU의 속도	10MIPS
GCMCPUSpeed	GCM CPU의 속도	30MIPS
NetBandWidth	네트워크의 데이타 전송속도	10Mbps
NumNode	참여 노드 수	5~40
NumDisk	각 노드의 디스크 수	1disks
DiskTime	디스크 액세스 시간	20ms
ObjectsPerPage	페이지에 포함된 레코드 수	20
DBSize	데이타베이스에 저장된 레코드 수	10,000
CacheHitRatio	캐쉬 히트율	80%
오버헤드 변수 및 트랜잭션 변수		
FixedMsgInst	메시지 처리를 위한 고정 명령수	20,000
ControlMsgSize	제어 메시지의 길이	256Bytes
LockInst	로크 등록/해제를 위한 명령수	300
PerIOInst	디스크 I/O를 위한 명령수	5000
TranSize	트랜잭션 당 레코드 액세스 수	50records
WriteOpPct	갱신 연산 비율	10%~50%

GCM이 전체 시스템의 병목이 되는 것을 방지하기 위해서 GCM이 사용하는 CPU는 다른 노드에서 사용하는 CPU보다 성능이 우수하다고 가정하였다. 중복사본을 갖는 노드 수는 5에서 40까지 변화하고, 모든 노드는 10,000개의 중복 데이타를 가지고 있고 1개의 디스크에 저장된다. CacheHitRatio에 따라 캐쉬에 없는 데이타를 액세스하기 위한 디스크 액세스 시간은 0.02초이다. CPU와 디스크는 FIFO 큐를 이용하여 I/O 요청 및 로크 요청 등을 순서대로 처리한다. 네트워크 관리자는 10Mbps의 대역폭을 갖는 FIFO 서버로 구현되었다. 네트워크를 통해 메시지를 전송하는 과정을 표현하기 위해 노드의 CPU 및 GCM의 CPU는 메시지마다 Fixed MsgInst 만큼의 명령수와 ControlMsgSize 만큼의 추가적인 명령수를 실행한다. SER이나 BA와 달리, MOERS의 경우 GCM은 전역 직렬성을 검사하기 위해 LockInst 만큼의 명령수를 추가적으로 실행한다. 각 트랜잭션이 액세스하는 레코드 수는 50이고 액세스하는 각각의 레코드를 갱신할 확률은 WriteopPct이다. 본 논문에서는 중복사본을 갖는 노드 수(NumNode)와 WriteOpPct의 값을 다양하게 변화하면서 MOERS의 성능을 분석하고자 한다.

모의 실험에 사용된 주요 성능 평가 지수는 트랜잭션 응답 시간이다. 트랜잭션 응답 시간은 트랜잭션이 생성된 후 완료될 때까지의 시간을 의미하는 것으로 큐에서의 대기 시간 및 트랜잭션이 철회되어 재실행되는 시간도 모두 포함된다. 그 외에 사용되는 성능 평가 지수로 철회 트랜잭션 수와 로크 대기 시간을 들 수 있다. 철회 트랜잭션 수는 모의 실험 기간동안 철회된 전체 트랜잭션 수를 의미하고, 로크 대기 시간은 하나의 트랜잭션이 완료할 때까지 로크 획득에 걸리는 로크 큐 대기 시간으로 트랜잭션 응답 시간에 포함된다. 신뢰성 있는 모의 실험 결과를 얻기 위하여 배치 평균 기법(batch mean method)을 사용한다. 즉, 본 논문에서 나타난 실험 결과들은 30개의 다른 seed를 이용하여 산출된 결과들의 평균값이다. 각각의 실험들은 완료된 트랜잭션 수가 2,000개가 될 때까지 수행하며, 초기 200개가 완료될 때까지의 결과들은 무시한다. 이러한 기법을 이용하여 산출된 결과들은 90%의 신뢰 수준을 만족하였다.

5. 실험 결과 분석

본 절에서는 개발한 성능 평가 모형을 이용하여 수행한 실험 결과를 분석한다. 본 논문에서 구현한 중복 기법은 BA와 SER, 그리고 본 논문에서 제안한 MOERS이다. 다양한 데이타베이스 액세스 유형[13,19]에 따른 중복 기법의 성능을 분석하기 위하여 데이타베이스 액세스 유형을 높은 충돌 환경, 클러스터링 액세스 환경, 그리고 균일 액세스 환경으로 나누어 실험을 수행하였다.

5.1 높은 충돌 환경

높은 충돌 환경은 모든 노드에서 실행되는 트랜잭션들이 데이타베이스의 특정 부분을 높은 확률로 액세스하는 경우를 모델링한 것이다. 이 환경에서 각 트랜잭션이 실행하는 연산의 80%는 전체 데이타베이스의 특정 20%(200페이지)를 액세스한다. 트랜잭션의 나머지 20%의 연산은 나머지 데이타베이스를 무작위로 액세스한다.

그림 2는 노드 수를 다양하게 변화시킬 때, 각 중복

기법의 성능 변화를 나타낸다. 트랜잭션의 평균 갱신 비율은 30%로 두었다. 노드 수가 증가할수록 MOERS의 성능이 BA와 SER에 비해 우수해지며, 노드 수가 40인 경우는 SER과 BA에 비해 3.4배 정도의 성능 향상을 보인다. 그 이유는 참여 노드 수가 증가할수록 동시에 실행되는 트랜잭션 수가 늘어나고, 따라서 트랜잭션들 간에 동일한 데이타를 액세스할 확률이 커지기 때문이다. SER의 경우, 트랜잭션을 실행하기 위해 로크를 액세스하고 송신 노드로부터 완료 혹은 철회 결정 메시지를 받으면 로크를 해제한다. 그 결과 트랜잭션이 완료될 때까지 보다 많은 시간이 소요되고, 그 트랜잭션이 보유하고 있는 로크를 기다리고 있는 다른 트랜잭션의 로크 대기 시간도 증가하게 된다. 또한, 철회 트랜잭션의 경우에도 철회 메시지가 오기 전에는 로크를 보유하고 실행해야 하므로 로크 대기 시간이 증가한다. 높은 충돌 환경에서는 SER과 BA간의 성능 차이가 작다. 그림 2에서 나타나듯이 노드 수가 10인 경우 SER은 BA에 비해 2배 정도 성능이 좋지만 노드 수가 증가할수록 성능 차이가 줄어들어 40인 경우는 10%정도의 성능 차이만 나타낸다. 그 이유는 노드 수가 증가할수록 트랜잭션 철회율이 높아지기 때문이다.

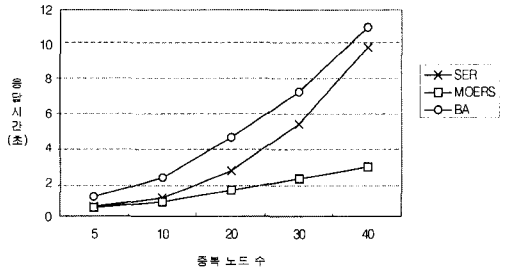

그림 2 높은 충돌 환경에서 노드 수 변화

표 2는 높은 충돌 환경에서 철회 트랜잭션의 수를 나타낸다. SER의 철회 트랜잭션 수가 가장 높고 BA의 철회 트랜잭션 수가 가장 적다. BA는 SER에 비해 동시성 정도는 떨어지지만 지역 교착상태에 의해서만 트랜잭션이 철회된다. SER은 직렬성 위반 여부가 갱신 요구 메시지를 방송한 후에 결정되는 낙관적 기법을 채택하고 있어 많은 수의 트랜잭션이 철회된다.

표 2에 나타난 각 기법들간의 철회 트랜잭션 수의 차이를 이해하기 위해 다음과 같은 예를 살펴보자. 노드 N_1, N_2, N_3가 데이타 *a*, *b*, *c*를 중복 저장하고 N_1은 T_1: $R_1(a)W_1(b)$, N_2는 T_2: $R_2(b)W_2(c)$, N_3는 T_3: $R_3(c)W_3(a)$를 동시에 실행한다고 가정하자. 그리고 갱신 요구 메시

표 3 높은 충돌 환경에서 노드 수 변화에 따른 철회 트랜잭션 수

중복 기법	중복 노드 수				
	5	10	20	30	40
SER	154	343	826	1568	2614
MOERS	144	291	570	814	1140
BA	14	42	117	199	310

징 순서는 $M_1 \rightarrow M_2 \rightarrow M_3$이라고 가정하면 SER에서는 T_2, T_3가 철회된다. 메시지 M_1이 전송되면 N_2는 M_1의 갱신 데이타와 충돌이 발생한 T_2를 철회하고, 노드 N_3도 M_2가 전송되면 M_2의 갱신 데이타와 충돌이 발생한 T_3을 철회한다. 그러나 BA는 대기 그래프에 순환이 발생하고 교착상태 해결 기법에 따라 가장 큰 MSN을 갖는 T_3만 철회된다. MOERS는 갱신 요구 메시지 방송 전에 전역 직렬성 검사가 이루어지기 때문에 N_2는 M_2를 방송하지 않는다. 따라서 SER과 달리 T_2만 철회되고 T_3는 철회되지 않는다. 앞의 예에서 N_3를 제외한 N_1과 N_2에서 각각 T_1과 T_2를 동시에 실행한다고 가정하면 MOERS에서는 T_2가 철회되지만 BA에서는 $T_1 \rightarrow T_2$ 순으로 철회없이 실행된다. 즉, BA는 로크에 의해 대기하는 트랜잭션으로 인하여 SER과 MOERS에 비해 상대적으로 동시성 정도가 낮아 철회 트랜잭션 수가 적은 것으로 해석된다. 표 3은 중복 노드 수에 따른 각 기법의 로크 대기 시간을 나타낸다. 예상한 것과 같이 BA는 가장 긴 로크 대기 시간을 갖는다.

표 4 높은 충돌 환경에서 노드 수 변화에 따른 로크 대기 시간

중복 기법	중복 노드 수				
	5	10	20	30	40
SER	0.00626	0.03059	0.1341	0.26588	0.44943
MOERS	0.00002	0.00004	0.00016	0.00056	0.00219
BA	0.034482	0.142217	0.500381	0.912985	1.587065

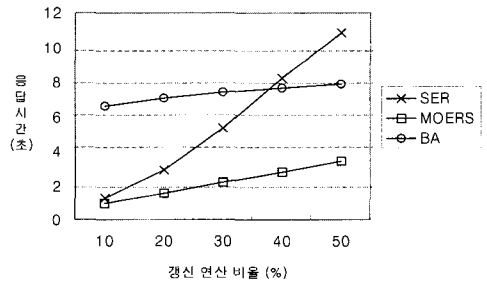

그림 3 높은 충돌 환경에서 갱신연산 비율 변화

그림 3은 중복 노드 수를 30으로 두었을 때, 갱신 연산의 비율(WriteOpPct)를 다양하게 변화하면서 실험한 결과를 나타낸다. WriteOpPct가 증가할수록 MOERS의 성능이 BA와 SER에 비해 우수해지며, WriteOpPct가 50%인 경우 SER에 비해 대략 4배, BA에 비해 2배 정도의 성능 향상을 보인다. 흥미로운 사실은 WriteOpPct가 40%보다 증가하면서 SER의 성능이 BA보다 저하된다는 것과 WriteOpPct가 커짐에 따라 SER은 8.6배, MOERS는 3.4배 성능 저하되지만, BA는 1.2배정도만 저하된다는 것이다. 그 이유는 다음과 같이 두 가지로 요약할 수 있다. 첫째, BA의 경우 WriteOpPct에 관계없이 중복 노드는 해당 트랜잭션의 판독 연산과 갱신 연산을 모두 실행하기 때문에 WriteOpPct의 증가에 따른 성능 저하 현상이 상대적으로 둔감하다. 그렇지만 갱신 연산만 방송하는 SER과 MOERS는 WriteOpPct가 커질수록 보다 많은 갱신 데이타를 방송하고 실행하므로 응답시간이 길어지게 된다. 둘째, WriteOpPct가 커질수록 SER의 철회 트랜잭션 실행 오버헤드가 증가한다. 데이타들이 동일한 확률로 액세스되며, 트랜잭션은 데이타를 액세스하기 전에 로크를 요청하며 완료할 때 모든 로크를 반납할 경우, 데이타 충돌 확률은 $(1-s^2)k^2N/D$라는 식으로 나타난다[20]. 이때, s는 판독 연산의 비율이고 k는 트랜잭션이 요청한 로크 수이다. 그리고 N은 다중 프로그래밍 정도(노드 수)이며 D는 데이타 수를 나타낸다. 따라서 WriteOpPct가 증가할수록 앞의 식에서 상대적으로 판독 연산 비율이 줄어들어 데이타 충돌 확률이 증가하고 그 결과 철회 트랜잭션 수가 증가하게 된다.

5.2 클러스터링 액세스 환경

클러스터링 액세스 환경은 각 노드마다 액세스하는 데이타들이 분할되어 있고, 노드는 자신에게 할당된 데이타를 주로 액세스하는 환경을 모델링한 것이다. 각 트랜잭션이 실행하는 연산의 20%는 시스템 카탈로그나 파일 디렉토리, 혹은 B-트리 인덱스의 루트 노드나 중간 노드와 같은 공유 데이타를 액세하며, 60%의 연산은 해당 노드에 할당된 데이타를 액세스한다. 나머지 20%의 연산은 데이타베이스의 나머지 부분을 무작위로 액세스한다. 데이타베이스에서 공유 데이타가 차지하는 부분은 10%(100 페이지)이며, 노드에 할당된 데이타는 데이타베이스의 약 4.5%(45페이지)이다. 그림 4는 노드 수를 다양하게 변화시킬 때, 각 중복 기법의 성능 변화를 나타낸다. 트랜잭션의 평균 갱신 비율은 30%로 두었다.

높은 충돌 환경보다 모든 기법의 성능이 향상되었다. 그리고 MOERS는 SER에 비해 1.2배 정도만 향상되었

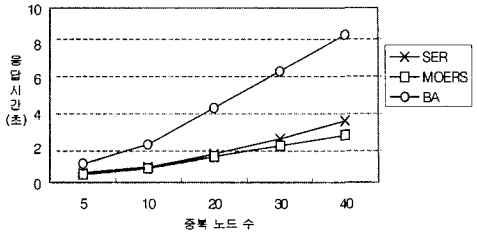

그림 4 클러스터링 액세스 환경에서 노드 수 변화

을 뿐이다. 그 이유는 클러스터링 환경에서는 상이한 트랜잭션이 동일한 데이타를 액세스할 확률이 줄고 그 결과 상대적으로 트랜잭션 철회 가능성도 줄어들어 SER의 문제점인 철회 트랜잭션으로 인한 오버헤드가 감소한 때문으로 해석된다. 그러나 BA는 동일한 데이타를 액세스할 빈도수는 줄었지만 여전히 판독 연산을 모든 노드에서 실행해야 하기 때문에, MOERS와 SER에 비해 데이타 충돌 확률이 높다.

5.3 균일 액세스 환경

균일 액세스 환경은 데이타베이스에 저장된 모든 데이타를 동일한 확률로 액세스하는 환경을 모델링한 것이다. 이 환경에서는 각 트랜잭션이 실행하는 연산의 20%는 공유 데이타를 액세스하며, 나머지 80%는 데이타베이스의 나머지 부분을 균일하게 액세스한다. 공유 데이타는 데이타베이스의 10%를 차지한다. 그림 5는 갱신 연산 비율을 30%로 두고 노드 수를 5에서 40까지 변화하면서 실험한 결과이다.

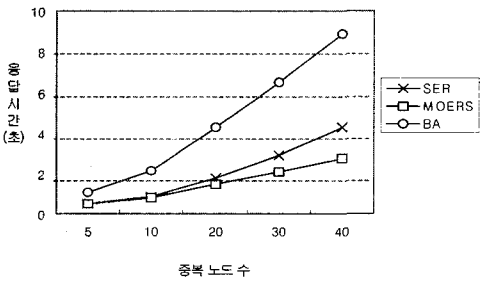

그림 5 균일 액세스 환경에서 노드 수 변화

클러스터링 액세스 환경과 마찬가지로 높은 충돌 환경에 비해 모든 중복기법의 트랜잭션 응답 시간이 줄어든다. 그러나, 클러스터링 액세스 환경과 달리 노드 수가 증가할수록 MOERS의 성능이 BA와 SER에 비해 우수해지며, 노드 수가 40인 경우 SER에 비해 1.5배, BA에 비해 3배 정도의 성능 향상을 보인다. 성능 향상

정도가 높은 충돌 환경보다 낮고 클러스터링 액세스 환경보다 높은 것은, 균일 액세스 환경에서 동일한 데이타를 액세스할 확률이 높은 충돌 환경에 비해서 낮고 클러스터링 액세스 환경에 비해 높기 때문이다. 즉, 데이타 충돌 확률이 높을수록 철회 트랜잭션 수가 증가하여 철회 트랜잭션 실행과 방송 오버헤드를 줄인 MOERS의 성능은 우수해지지만 SER의 성능은 저하된다. 가장 높은 데이타 충돌 확률을 갖는 높은 충돌 환경에서는 BA의 판독 연산으로 인한 오버헤드가 SER의 철회 트랜잭션 실행과 방송 오버헤드에 의해 상쇄되기 때문에 BA와 SER이 거의 유사한 성능을 보인 것으로 해석된다. 반면에 균일 액세스 환경은 높은 충돌 환경에 비해 상대적으로 데이타 충돌 확률이 낮기 때문에, BA보다 2배정도 우수한 성능을 보여준다.

6. 결 론

본 논문에서는 중복사본을 갖는 분산 시스템에서 트랜잭션 철회에 따른 실행과 방송 오버헤드를 줄이기 위한 중복 기법인 MOERS를 제안하였다. MOERS는 기존에 제안된 BA나 SER 기법과 같이 그룹 통신의 메시징 순서를 이용하여 즉시 갱신 기법의 빈번한 교착상태 발생 문제를 해결한다. 그리고 MSN 요청 시 판독 연산의 유효성 검증을 통한 전역 직렬성을 검사하고 그 결과에 따라 갱신 연산의 방송 여부를 결정한다. 판독 연산에서 액세스한 데이타가 이미 무효화된 경우, 송신 노드는 갱신 요구 메시지를 방송하지 않으므로 SER과 달리 모든 수신 노드에서 철회 트랜잭션을 실행할 필요가 없고 송신 노드는 철회 메시지를 방송하지 않아도 된다.

제안된 기법의 성능을 분석하기 위하여 모의 실험 모형을 개발하였고, 다양한 데이타 액세스 환경에서 실험을 하였다. 실험 결과를 정리하면 다음과 같다. 첫째, MOERS는 노드 수가 늘어남에 따라 데이타 액세스 환경에 관계없이 SER과 BA에 비해 성능이 향상됨을 보인다. 그 이유는 철회 트랜잭션의 실행 부담 및 철회 결과 메시지의 방송 오버헤드를 줄였으며, 전역 직렬성 검사 결과 메시지가 도착할 때까지 로크 해제를 지연하는 SER에 비해 로크 대기 시간이 줄어들기 때문이다. 둘째, 높은 충돌 환경에서 MOERS는 SER에 비해 최대 3.4배까지 성능이 향상되었다. 그 이유는 높은 충돌 환경에서 트랜잭션 철회율이 증가하고, 그 결과 SER의 철회 트랜잭션 실행 오버헤드가 증가하기 때문이다. 노드 수가 증가할수록 SER은 철회 트랜잭션의 실행 오버헤드가 가중되어 모든 노드에서 판독 연산을 실행하는 BA와 거의 성능 차이가 없다. 마지막으로, 클러스터링 액세스 환경에서 MOERS는 SER에 비해 약간의 성능 향상을 보인다. 클러스터링 액세스 환경에는 상이한 트랜잭션간의 데이타 충돌 확률이 거의 없기 때문에 철회 트랜잭션 발생 확률이 줄어들기 때문이다.

본 논문의 향후 연구 과제로는 데이타가 모든 노드에 완전 중복되지 않고 부분적으로 중복된 환경으로 MOERS를 확장하는 방법과 데이타베이스나 그룹 통신에 고장이 발생할 경우의 해결 방법, 그리고 실제 시스템에서 MOERS의 구현 및 성능 평가 등을 들 수 있다. 특히, MOERS의 구현을 위하여 그룹 통신 시스템과 분산 데이타베이스 시스템이 모두 필요하므로, 본 연구의 관련 연구로 수행되고 있는 그룹 통신 시스템의 시제품에 분산 데이타베이스 시스템의 기능 일부를 통합하는 작업이 현재 계획 중에 있다.

참 고 문 헌

[1] J. Gray, P. Helland, P. O'Neil and D. Shasha, "The Dangers of Replication and a Solution," *Proc. ACM SIGMOD*, pp. 173-182, 1997.

[2] T. Anderson, Y. Breitbart, H.F. Korth and A. Wool, "Replication, Consistency, and Practicality: are These Mutually Exclusive?" *Proc. ACM SIGMOD*, pp. 484-495, 1998.

[3] Y. Breitbart and H.F. Korth, "Replication and Consistency: being Lazy Helps Sometimes," *Proc. 16th ACM Symposium on Principles of Database Syst.*, pp. 173-184, 1997.

[4] P. Chundi, D.J. Rosenkrantz and S. Ravi, "Deferred Updates and Data Placement in Distributed Databases," *Proc. Int'l Conf. on Data Eng.*, pp. 469-476, 1996.

[5] E. Pacitti, P. Minet and E. Simon, "Fast Algorithms for Maintaining Replica Consistency in Lazy Master Replicated Databases," *Proc. Int'l Conf. on VLDB*, pp. 126-137, 1999.

[6] M. Butrico et al., "Gold Rush: Mobile Transaction Middleware with Java-Object Replication," *Proc. 3rd USENIX Conf. on Object-Oriented Technologies and Syst.*, pp. 91-101, 1997.

[7] J. Holliday, D. Agrawal and A. Abbadi, "The Performance of Database Replication with Group Multicast," *Proc. IEEE 29th Int'l Symposium on Fault Tolerant Computing*, pp. 158-165, 1999.

[8] J. Holliday, D. Agrawal and A. Abbadi, "Using Multicast Communication to Reduce Deadlock in Replicated Databases," *Proc. 19th IEEE Symposium on Reliable Distributed Syst.*, pp. 196-205, 2000.

[9] B. Kemme and G. Alonso, "A Suite of Database

Replication Protocols based on Group Communication Primitives Distributed Computing Systems," *Proc. 18th Int'l Conf. on Distributed Computing Syst.*, pp. 156-163, 1998.

[10] B. Kemme and G. Alonso, "Don't be Lazy, be Consistent: Postgres-R, A New Way to Implement Database Replication," *Proc. Int'l Conf. on VLDB*. pp. 134-143, 2000.

[11] B. Kemme and G. Alonso, "A New Approach to Developing and Implementing Eager Database Replication Protocols," *ACM Trans. Database Syst.*, Vol. 25, No. 3, pp. 333-379, 2000.

[12] B. Kemme, F. Pedone, G. Alonso and A. Schiper, "Processing Transactions over Optimistic Atomic Broadcast Protocols," *Proc. 19th IEEE Int'l Conf. on Distributed Computing Syst.*, pp. 424-431, 1999.

[13] M. Carey, M. Franklin and M. Zaharioudakis, "Fine-Grained Sharing in Page Server DBMS," *Proc. ACM SIGMOD*, pp. 359-370, 1994.

[14] S. Mishra and L. Wu, "An Evaluation of Flow Control in Group Communication," *IEEE/ACM Trans. Networking*, Vol. 6, No. 5, pp. 571-587, 1998.

[15] M.F. Kaashoek and A. Tanenbaum, "An Evaluation of the Amoeba Group Communication System," *Proc. 16th Int'l Conf. on Distributed Computing Syst.*, pp. 436-447, 1996.

[16] K.P. Birman, A. Schiper, and P. Stephenson, "Lightweight Casual and Atomic Group Multicast," *ACM Trans. Computer Syst.*, Vol. 9, No. 3, pp. 272-314, 1991.

[17] L. Moser et al., "Totem: A Fault-tolerant Multicat Group Communication System," *Comm. ACM*, Vol. 39, No. 4, pp. 54-63, 1996.

[18] H. Schwtman, *CSIM User Guide for use with CSIM Revision 16*, MCC, 1992.

[19] H. Cho, "Performance of Cache Coherency Schemes in a Shared Disks Transaction Environment," *Proc. 6th IEEE Workshop on Future Trends of Distributed Computing Syst,*, pp. 154 -161, 1997.

[20] Y.C. Tay, N. Goodman and R. Suri, "Locking Performance in Centralized Databases," *ACM Trans. Database Syst.*, Vol. 10, No. 4, pp. 415-462, 1085.

문 애 경

1992년 영남대학교 전산공학과 학사. 1997년 영남대학교 전산공학과 석사. 2000년 영남대학교 컴퓨터 공학과 박사. 2000년 ~ 현재 한국전자통신연구원 이동분산처리팀 선임연구원. 관심 분야는 분산/병렬 데이타베이스, 분산 처리, 메시징 기술 등

남 궁 한

1976년 고려대학교 이학사. 1982년 미국 Columbia 대학교 전산학 석사. 1982년 ~ 1985년 LG전자 컴퓨터 사업부. 1987년 ~ 현재 한국전자통신연구원 책임연구원. 관심분야는 분산시스템, fault-tolerant system, group communication 등

조 행 래

1988년 서울대학교 컴퓨터공학과 학사. 1990년 한국과학기술원 전산학과 석사. 1995년 한국과학기술원 전산학과 박사. 1995년 ~ 현재 영남대학교 전자정보공학부 부교수. 관심분야는 분산/병렬 데이타베이스, 트랜잭션 처리, DBMS 개발 등