그룹통신 환경에서 효율적 즉시 갱신 중복 기법

문애경 °, 남궁한
한국전자통신연구원 컴퓨터소프트웨어 연구소
e-mail:{akmoon,nghan}@etri.re.kr

An Efficient Eager Replication Scheme in Group Communication Environment

Aekyung Moon °, Han Namgoong
Computer Software Laboratory, ETRI

요약

기존에 제안된 대부분의 중복 기법은 원본 트랜잭션을 완료한 후 비동기로 중복 사본에 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 그룹통신을 이용한 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 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.

1. 서론

분산 시스템 환경에서 가용성, 신뢰성, 그리고 고성능 트랜잭션 처리를 위하여 각 노드들은 중복 사본을 갖는다. 중복 사본의 일관성을 유지하기 위한 중복 기법은 "즉시 갱신(eager update)"과 "지연 갱신(lazy update)"의 두 가지 기법으로 나눌 수 있다[3]. 즉시 갱신은 원본 트랜잭션과 다른 노드의 중복 사본에 대한 갱신 요구 트랜잭션이 하나의 트랜잭션으로 구성되는 것으로, Gray[3]가 빈번한 교착상태 발생을 즉시 갱신의 문제로 지적한 이후 대부분 원본 트랜잭션을 완료한 후 비동기로 중복 사본을 갱신하는 지연 갱신 기법을 가정하고 있다[2]. 그러나 지연 갱신은 즉시 갱신에서 발생하는 빈번한 교착상태 발생 문제를 해결할 수는 있지만 데이타 일관성 유지가 사용자 책임이라는 단점을 갖는다.

최근 그룹통신을 이용한 즉시 갱신 중복 기법들이 제안되고 있다[4,5,7,8]. 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 증가하는 문제점을 갖는다. 즉, 직렬성 위반 여부가 갱신 요구 메시지 방송 이후 결정되기 때문에 송신 노드는 철회될 트랜잭션에 대한 갱신 요구 방송 오버헤드, 수신 노드는 트랜잭션의 실행 오버헤드를 갖는다. 따라서 본 논문은 철회될 트랜잭션의 갱신 요구 메시지 방송 오버헤드와 실행 오버헤드를 줄이는 것을 기본 개념으로 하는 즉시 갱신 기법을 제안한다. 제안된 기법은 그룹통신의 메시징 순서 기능에 직렬성 검사 기능을 추가하여 송신 노드는 트랜잭션의 판독 연산을 실행한 후, 갱신 요구 메시지를 방송하기 전에 전역 직렬성을 검사할 수 있다. 그 결과, 메시지 전송 횟수와 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 즉시 갱신 중복 기법(Efficient Eager Replication Scheme in Group Communication: ERGC)을 설명한다. 4절에서는 정성적인 성능 비교를 한다. 끝으로 5절에서 결론을 맺는다.

2. 관련 연구

기존의 그룹통신을 이용한 즉시 갱신 중복기법인 SER(Replication with Serializability)[8]에서 발생하는 문제점은 다음과 같다.

[예 1] 노드 N_1과 N_2가 데이타 a와 b를 중복 저장한

다고 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, 트랜잭션 철회율이 훨씬 높기 때문에 철회될 트랜잭션의 갱신 요구 메시지 방송 오버헤드와 실행 오버헤드를 줄이는 것은 시스템의 성능에 매우 중요한 영향을 미친다.

본 논문의 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. 즉시 갱신 중복 기법

3.1 자료 구조 및 가정

ERGC에 필요한 기본적인 자료 구조 및 가정은 [1]과 [9]에서 채택된 내용들과 동일하다.

그룹통신 관리자(Group Communication Manager: GCM)는 메시징 순서와 전역 직렬성 검사 역할을 담당하는 것으로 Amoeba 그룹통신[6]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[6], 이로 인하여 ISIS[1]는 분산형 메시징 기능을 동적-집중형 구조로 변경한 경우가 있기 때문이다. Sequencer는 메시징 순서 기능을 제공하기 위하여 메시지 일련번호(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 ERGC 알고리즘

ERGC에서 각 노드는 트랜잭션의 판독 연산을 진행하고 갱신 요구를 방송하기 전에 전역 직렬성을 검사하며 4단계로 구성된다.

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. 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 갱신

된 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 중복 사본에 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 GCM에 추가된 전역 직렬성 검사 기능을 통한 ERGC의 수행과정을 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 저장되며 GCM은 먼저 U-TBL에서 M_1의 <RS_1={a}>와 충돌하는 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 ERGC 알고리즘의 오버헤드 분석

ERGC에서의 GCM은 다른 기법과 달리 전역 직렬성 검사도 담당하기 때문에 U-TBL과 추가적인 처리 과정을 갖는다. 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 커질 경우 문제가 발생할 수 있다. 또한 GCM은 ERGC 알고리즘의 단계 2 처리 과정으로 인한 추가적인 오버헤드를 갖는다.

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. 뿐만 아니라 단계 2의 전역 직렬성 검사 과정으로 인하여 GCM이 전체 시스템의 병목이 되는 것을 방지하기 위하여 해싱과 같은 빠른 검색 방법을 이용한다. 즉, 단계 2에서 발생하는 가장 큰 오버헤드는 U-TBL에 저장된 <d_r, MSN(d_r)>를 검색하는 것으로, d_r의 해싱 값을 키로 사용하여 빠르게 검색함으로써 GCM의 추가적인 처리 과정으로 인한 오버헤드를 줄일 수 있다.

고려해야 할 또다른 점은 ERGC를 분산형 구조에 적용하는 경우 다른 기법과 달리 수정이 필요하다는 것이다. 기존의 분산형 메시징 순서 기능에 대한 연구는 Totem과 같이 토큰을 이용하는 방식[10]과 특

정 수신 노드에서 일방적으로 MSN을 방송하는 방식[2]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[9] 자체적으로 U-TBL의 크기가 조절되어 ERGC의 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.

4. 성능 비교

본 절에서는 SER과 ERGC에서 액세스하는 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.

SER과 ERGC는 다른 노드에서 발생한 트랜잭션은 갱신 연산만 실행하면 되므로, $\#D(N_i)$는 다음과 같다. $\#D(N_i)$가 클 경우 액세스되는 데이타 수가 증가하고 데이타간 충돌 확률이 높아진다.

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

수식 (1)은 모든 트랜잭션이 철회없이 실행된다는 가정에서만 동작하며, 트랜잭션이 철회될 경우를 고려하면 ERGC와 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 $tr\_length \times wpct$ 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 $tr\_length \times wpct \times (\#N-1)$ 만큼의 데이타 액세스 오버헤드가 발생한다. 이와는 달리 ERGC는 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 ERGC는 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 ERGC는 SER보다 우수한 성능을 보일 수 있다.

5. 결론

중복 사본을 갖는 분산 시스템에서 그룹통신을 이용한 즉시 갱신 기법을 지원하는 경우, 교착상태 발생률은 줄였지만, 송신 노드는 갱신요구 메시지를 방송한 후 전역 직렬성을 검사하기 때문에, 동시성이 증가할수록 철회 트랜잭션의 방송 및 실행 부담이 증가로 인하여 성능이 저하될 수 있다. 본 논문에서 제안한 직렬성 기능을 추가한 그룹통신의 즉시 갱신기법이 갖는 장점은 다음과 같다. 첫째, 갱신 요구 메시지를 방송하기 전에 전역 직렬성 검사가 이루어지기 때문에 완료 트랜잭션은 한번의 메시지 방송으로 처리할 수 있다. 둘째, 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 향상시킬 수 있다. 본 논문의 향후과제는 제안한 기법을 분석모델을 통하여 기존 기법과 비교하는 것이다.

참고문헌

[1] K.P. Birman, A. Schiper, and P. Stephenson, "Lightweight Casual and Atomic Group Multicast," *ACM Trans. Computer Syst.*, 9(3), pp. 272-314, 1991.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] B. Kemme and G. Alonso, "A New Approach to Developing and Implementing Eager Database Replication Protocols," *ACM Trans. Database Syst.*, 25(3), pp. 333-379, 2000.

[9] S. Mishra and L. Wu, "An Evaluation of Flow Control in Group Communication," *IEEE/ACM Trans. Networking*, 6(5), pp. 571-587, 1998.

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