공유 디스크 기반의 다중 서버 DBMS를 위한 효율적인 버퍼 일관성 관리

(Efficient Buffer Coherency Management for a Shared-Disk based Multiple-Server DBMS)

고 현 선 † 김 이 른 †† 이 민 재 ††† 황 규 영 ††††
(Hyun-Sun Ko) (Yi-Reun Kim) (Min-Jae Lee) (Kyu-Young Whang)

요 약 공유디스크 모델을 사용하는 다중 서버 DBMS에서는 서버 프로세스들이 서로 독립된 메모리를 가지므로, 특정 서버 프로세스가 데이터베이스를 수정하더라도 다른 서버 프로세스들의 버퍼에는 수정된 내용이 반영되지 않는다. 따라서, 다른 서버 프로세스들이 수정되기 전 내용에 대하여 데이터 처리 요청을 수행하면 문제가 발생한다. 본 논문에서는 큰 단위의 로크(여기서는 볼륨 로크)를 사용하는 DBMS에서 이러한 문제를 해결하기 위한 새로운 방법을 제안한다. 이 방법에서는 서버 프로세스가 트랜잭션을 커밋할 때 수정한 페이지의 식별자와 타임스탬프를 일관성 볼륨에 저장하고, 이 정보를 통하여 다른 서버 프로세스가 로크를 획득하는 시점에 일관성 볼륨에서 다른 프로세스가 먼저 수정하였는지 여부를 확인하여 해당 페이지를 버퍼에서 무효화시켜 디스크에서 최신의 버전을 새로 읽어 들인다. 이 방법은 매우 작은 크기의 일관성 볼륨만을 사용하고, 액세스하는 데이터의 양이 적어서 성능이 매우 빠르다.

키워드 : 공유디스크 모델, 다중 서버 DBMS, 버퍼 일관성 문제, 일관성 볼륨

Abstract In a multiple-server DBMS using the share-disk model, when a server process updates data, the updated ones are not immediately reflected to the buffers of the other server processes. Thus, the other server processes may read invalid data. In this paper, we propose a novel method to solve this problem. In this method the server process stores the identifiers and timestamps of the pages that have been updated during a transaction into the coherency volume when the transaction commits. Then, the server process invalidates its buffers of the pages updated by the other server processes by accessing the coherency volume when the lock is acquired, and, subsequently, read the up-to-date versions of the pages from disk. This method needs only a very small coherency volume and shows a good performance because the amount of data that need to be accessed is very small.

Key words : shared-disk model, multiple-server DBMS, buffer coherency problem, coherency volume

· 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 국가지정연구실(NRL) 과제를 통하여 한국과학재단의 지원을 받아 수행된 연구임(R0A-2007-000-20101-0)

† 비 회 원 : KAIST 전산학과 박사과정
hsko@mozart.kaist.ac.kr
†† 학생회원 : KAIST 전산학과 박사과정
yrkim_s@mozart.kaist.ac.kr
††† 비 회 원 : KAIST 전산학과
mjlee@mozart.kaist.ac.kr
†††† 종신회원 : KAIST 전산학과 특훈 교수
kywhang@mozart.kaist.ac.kr

논문접수 : 2009년 5월 15일
심사완료 : 2009년 8월 24일

정보과학회논문지: 데이타베이스 제36권 제5호(2009.10)

1. 서 론

오늘날 대부분의 기업들은 인터넷을 통해 데이터베이스를 액세스할 수 있도록 하여 다양한 서비스를 온라인으로 제공하고 있다. 이러한 서비스를 제공하기 위하여 웹 서버와 데이터베이스 관리 시스템(DBMS)을 연동하여 사용한다[1]. 이때, 동시에 들어오는 수많은 사용자의 요청을 빠르게 처리하기 위해서는 공유디스크(shared-disk) 모델[2]을 사용하는 다중 서버 DBMS를 사용해야 할 필요성이 있다[3].

공유디스크 모델을 사용하는 다중 서버 DBMS는 서버 프로세스마다 독립된 메모리를 가지며 디스크를 서로 공유한다. 이때, 서버 프로세스들이 메모리를 서로 공유하지 않으므로 하나의 서버 프로세스에서 발생한 오류가 다른 서버 프로세스들에 영향을 미치지 않기 때문에 안

정성이 높다[2]. 그러나, 공유디스크 모델을 사용하는 DBMS에서는 버퍼 일관성 문제가 발생할 수 있다[4].

버퍼 일관성 문제는 특정 서버 프로세스가 데이터베이스를 수정하더라도 다른 서버 프로세스들의 버퍼에는 수정된 내용이 반영되지 않기 때문에, 다른 서버 프로세스들이 질의를 수행할 때 수정되기 전의 내용을 바탕으로 질의를 수행하는 문제이다. 이 문제는 데이터베이스의 일관성을 깨뜨릴 수 있으므로 반드시 해결되어야 한다.

버퍼 일관성 문제를 해결하기 위한 간단한 방법은 트랜잭션 시작 시에 서버 프로세스의 모든 버퍼 페이지들을 무효화시키고(invalidate), 페이지들을 디스크로부터 새로이 읽어 들이는 방법이다. 그러나, 이 방법은 수정되지 않은 페이지들까지도 버퍼에서 무효화시키고 새로 읽어 들이기 때문에, DBMS의 성능을 크게 저하시킨다. 따라서 효율적인 버퍼 일관성 관리를 하기 위해서는 다른 서버 프로세스에서 수정된 페이지들만을 버퍼에서 찾아 내어 무효화시켜야 한다.

본 논문에서는 공유디스크 모델을 사용하는 다중 서버 DBMS에서 효율적으로 버퍼 일관성을 관리하는 방법을 제안한다. 이때, DBMS는 동시성 제어 방법으로서 로크 기반 방법을 사용하는 것만을 대상으로 한다. 또한 설명의 편의를 위하여 로크의 단위가 볼륨임을 가정하나, 제안한 방법을 쉽게 확장하여 다른 로크의 단위를 갖는 시스템에도 적용시킬 수 있다. 여기서 볼륨은 하나 이상의 파일로 구성된 논리적인 저장 단위이다.

본 논문의 구성은 다음과 같다. 2장에서는 논문에서 제안하는 버퍼 일관성 관리 방법의 아키텍처를 설명하고, 3장과 4장에서는 버퍼 일관성 관리 방법을 설명한다. 5장에서 성능을 평가하고, 6장에서 관련 연구를 소개하며, 7장에서 결론을 내린다.

2. 버퍼 일관성 관리 방법의 아키텍처

버퍼 일관성 관리를 위해서는 서버 프로세스가 다른 서버 프로세스에서 수정된 페이지들을 버퍼에서 무효화시켜, 디스크로부터 최신의 페이지들을 새로이 버퍼로 읽어 들이도록 해야 한다. 이때, 서버 프로세스가 다른 서버 프로세스에서 수정된 페이지들을 버퍼에서 무효화시키는 것을 '버퍼 내용을 데이터베이스와 일치시킨다'고 표현한다.

본 논문에서는 다른 서버 프로세스에서 수정된 페이지들만을 버퍼에서 찾아내기 위하여 일관성 볼륨(coherency volume)을 사용한다. 각 서버 프로세스는 수정한 페이지들에 대한 정보를 일관성 볼륨에 저장하고, 다른 서버 프로세스에서 수정된 페이지들에 대한 정보를 일관성 볼륨으로부터 읽어서 무효화시킬 버퍼 페이지들을 찾아낸다. 그림 1은 이와 같이 동작하는 버퍼 일관성 관리

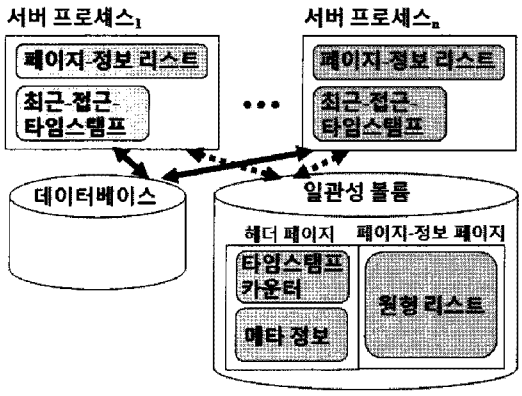

그림 1 버퍼 일관성 관리 방법의 아키텍처

방법의 아키텍처를 나타낸 것이다.

모든 서버 프로세스들이 공유하는 일관성 볼륨은 '헤더 페이지'와 '페이지-정보 페이지'로 구성된다. 헤더 페이지는 페이지-정보 페이지에 대한 메타 정보와 '타임스탬프 카운터'를 가진다. 타임스탬프 카운터는 각 트랜잭션에 타임스탬프를 할당하기 위하여 사용되는 카운터이다. 페이지-정보 페이지는 서버 프로세스들에 의해 수정된 페이지들의 정보를 가진다. 수정된 페이지의 정보는 <페이지 식별자, 타임스탬프>의 쌍인 '페이지-정보'로 나타낸다. 페이지 식별자는 수정된 페이지의 식별자이고, 타임스탬프는 해당 페이지를 수정한 트랜잭션의 타임스탬프이다. 서버 프로세스들이 수정한 페이지의 페이지-정보들은 모두 저장하려면 저장 공간이 많이 필요할 뿐만 아니라 이를 저장하고 읽기 위한 오버헤드가 크기 때문에 페이지-정보 페이지는 제한된 개수(N)의 페이지-정보들만을 원형 리스트로 저장한다. 이때, 제한된 개수의 페이지-정보만을 저장하면서 버퍼 일관성 관리를 할 수 있는 방법은 4장에서 설명한다.

페이지-정보가 타임스탬프를 포함하고 있는 이유는 서버 프로세스가 마지막으로 데이터베이스를 액세스한 이후에 기록된 페이지-정보들을 찾아서 그 페이지들만을 버퍼에서 무효화시키기 위해서이다. 타임스탬프로는 시스템 시간과 같은 물리적인 시간을 사용할 수도 있으나, 본 논문에서는 논리적인 시간을 나타내는 정수형의 카운터를 사용함으로써 타임스탬프를 업데이트하고 저장하는 오버헤드를 줄인다.

각 서버 프로세스는 '페이지-정보 리스트'와 '최근-접근-타임스탬프'를 메모리상에 유지한다. 페이지-정보 리스트는 현재 수행 중인 트랜잭션에서 수정한 페이지들의 페이지-정보를 저장한다. 최근-접근-타임스탬프는 서버 프로세스가 마지막으로 데이터베이스를 액세스한 시점을 저장하는 타임스탬프이다. 이것은 서버 프로세스가 마지막으로 데이터베이스를 액세스한 이후에 기록된 페이지-정보들을 찾기 위하여 사용된다.

3. 버퍼 일치 과정

버퍼 일관성 문제는 특정 서버 프로세스가 볼륨의 내용을 수정하고 트랜잭션을 커밋한 이후, 다른 서버 프로세스가 동일한 볼륨을 액세스하는 순간에 발생한다. 따라서, 버퍼 내용을 데이터베이스와 일치시키는 시점은 서버 프로세스가 특정 볼륨에 대한 트랜잭션을 커밋한 이후, 다른 서버 프로세스가 해당 볼륨을 액세스하기 이전이 되어야 한다. 따라서, 트랜잭션 커밋 시에는 일관성 볼륨에 페이지-정보를 기록하고, 볼륨 로크를 획득하는 시점에 일관성 볼륨을 읽어서 버퍼 내용을 데이터베이스와 일치시켜야 한다.

먼저, 트랜잭션 커밋 시 볼륨 로크를 해제하기 이전에는 페이지-정보 리스트의 페이지-정보들을 일관성 볼륨에 저장한다. 이를 위하여, 최근-접근-타임스탬프의 값을 타임스탬프 카운터 값으로 업데이트하고, 다음 트랜잭션을 위하여 타임스탬프 카운터의 값을 1만큼 증가시킨다. 그리고 페이지-정보 리스트의 모든 페이지-정보들의 타임스탬프에 최근-접근-타임스탬프 값을 기록하고 이들을 일관성 볼륨의 원형 리스트에 저장한다. 이때, 페이지-정보들을 원형 리스트의 뒷부분에 추가하는데, 만약 원형 리스트가 가득 차 있으면 오래된 페이지-정보부터 덮어 쓴다. 이렇게 하면, 페이지-정보들이 타임스탬프 순서로 저장된다.

볼륨 로크를 획득할 때에는 일관성 볼륨을 액세스하여 버퍼 내용을 데이터베이스와 일치시킨다. 이를 위하여, 일관성 볼륨의 원형 리스트를 뒤에서부터 스캔 하며 서버 프로세스의 최근-접근-타임스탬프 값보다 타임스탬프 값이 큰 페이지-정보들을 찾아서 이에 해당하는 페이지를 버퍼에서 무효화시킨다. 이때, 원형 리스트를 뒤에서부터 스캔 하는 이유는 스캔 범위를 줄이기 위해서이다. 페이지-정보들이 타임스탬프 순서로 저장되어 있으므로 최근-접근-타임스탬프 값보다 타임스탬프 값이 큰 페이지-정보까지만 스캔 하면 된다.

여러 서버 프로세스들이 일관성 볼륨을 액세스하므로 동시성 제어를 위하여 일관성 볼륨에 대한 쓰기 로크와 읽기 로크를 사용한다. 일관성 볼륨에 대한 파손 회복은 고려하지 않아도 된다. 왜냐하면, 시스템 파손 후에는 시스템 재시동 시 모든 버퍼 페이지를 디스크로부터 새로이 읽어 들이면 되기 때문이다.

4. 제한된 크기의 자료 구조를 사용하여 발생하는 문제와 해결 방법

본 논문에서 제안하는 버퍼 일관성 관리 방법에서는 일관성 볼륨에 제한된 개수(N)의 페이지-정보들만을 저장하고 제한된 크기의 정수형 타임스탬프를 사용한다. 본 장에서는 이 때문에 발생하는 문제들인 원형 리스트의 랩 어라운드(wrap around)로 인한 문제, 타임스탬프 값의 오버플로우로 인한 문제, 타임스탬프 값의 오버플로우로 인해 원형 리스트의 랩 어라운드를 발견하지 못하는 문제를 설명하고, 이러한 문제들의 해결 방법에 대하여 설명한다. 먼저, 표 1에서 이러한 문제들을 설명하기 위하여 사용되는 표기법들을 정의한다.

표 1 설명을 위하여 사용되는 표기법

기호	정의/의미
t_{head} /t_{tail}	일관성 볼륨 내에서 원형 리스트의 머리/꼬리가 가리키는 페이지-정보의 타임스탬프 값
t_{max}	타임스탬프로 사용되는 정수형의 최대값
N	원형 리스트에 저장되는 페이지-정보의 최대 개수

4.1 원형 리스트의 랩 어라운드로 인한 문제

일관성 볼륨의 원형 리스트에서 일부 서버 프로세스에서 아직 데이터베이스와 일치시키지 못한 페이지-정보가 새로 저장된 페이지-정보에 의해서 덮어 쓰여질 수 있다. 이 경우 일부 서버 프로세스는 덮어 쓰여진 페이지-정보에 해당하는 버퍼 페이지를 데이터베이스와 일치시켜야 함에도 불구하고 일치시키지 못하는 문제가 발생한다. 이러한 문제를 '원형 리스트의 랩 어라운드로 인한 문제'라고 정의한다. 이 문제는 페이지-정보들을 제한된 크기의 원형 리스트에 저장하기 때문에 발생한다.

서버 프로세스의 최근-접근-타임스탬프가 원형 리스트의 t_{head}보다 작은 경우 이러한 문제가 발생하였을 가능성이 있다. 이때, 최근-접근-타임스탬프 이후 최소 N개의 페이지가 수정된 것이므로 많은 수정에 의해 대부분의 버퍼 페이지가 수정된 것으로 간주하고 서버 프로세스의 버퍼에 있는 모든 페이지들을 데이터베이스와 일치시킨다.

4.2 타임스탬프 값의 오버플로우로 인한 문제

페이지-정보의 타임스탬프 값이 오버플로우되면 일관성 볼륨의 원형 리스트의 페이지-정보들이 타임스탬프 순서로 정렬되지 않게 된다. 이 경우는 일관성 볼륨의 원형 리스트 내에 페이지-정보들이 타임스탬프 순서로 정렬되어 있다는 가정에 어긋나기 때문에 볼륨 로크 획득 시 버퍼 내용과 데이터베이스를 일치시키는 작업이 정상적으로 수행되지 않는 문제가 발생한다. 이러한 문제를 '타임스탬프 값의 오버플로우로 인한 문제'라고 정의한다. 이 문제는 타임스탬프로 사용되는 정수형이 나타낼 수 있는 값의 최대 크기(t_{max})가 존재하기 때문에 발생한다.

타임스탬프 값의 오버플로우 여부를 판별하는 방법은

다음과 같다. 일관성 볼륨의 원형 리스트를 뒤에서부터 스캔 하면서 각 페이지-정보의 타임스탬프 값을 원형 리스트의 t_{head}와 비교한다. 만약 페이지-정보의 타임스탬프 값이 t_{head}보다 작으면 해당 타임스탬프가 오버플로우되었다고 판단한다.

이 문제를 해결하기 위하여 타임스탬프의 최대값으로 t_{max}를 사용하지 않고, 그보다 작은 값 $t_{timestamp\_max}$ ($=\left\lfloor \frac{t_{max}}{2} \right\rfloor -1$)를 사용한다. 그리고, 오버플로우된 타임스탬프들에는 일정한 값 t_{add}($=\left\lfloor \frac{t_{max}}{2} \right\rfloor$)를 더하여 최근-접근-타임스탬프와 비교한다. 이때, $t_{timestamp_max}$와 t_{add}는 다음 세 가지 조건을 만족시키는 값이다. 첫째, 타임스탬프의 최대값에 t_{add}를 더한 값이 오버플로우되지 않아야 한다. 이를 위해서는 $t_{timestamp_max} + t_{add}$가 t_{max}보다 작아야 한다. 둘째 타임스탬프의 최소값(=0)에 t_{add}를 더한 값이 어떠한 타임스탬프 값보다 커야 한다. 셋째, 타임스탬프 값이 자주 오버플로우되는 것을 방지하기 위해서 $t_{timestamp_max}$를 최대화시켜야 한다.

4.3 타임스탬프 값의 오버플로우로 인해 원형 리스트의 랩 어라운드를 발견하지 못하는 문제

타임스탬프 값이 오버플로우되면 원형 리스트의 랩 어라운드로 인한 문제가 발생하였음에도 불구하고 이를 발견하지 못하는 문제가 발생할 수 있다. 이러한 문제를 '타임스탬프 값의 오버플로우로 인해 원형 리스트의 랩 어라운드를 발견하지 못하는 문제'라고 정의한다. 이 문제는 특정 서버 프로세스의 최근-접근-타임스탬프가 오버플로우되지 않은 값을 가지는 상태에서 페이지-정보들의 타임스탬프에 오버플로우된 타임스탬프가 부여되고, 그 후 다른 서버 프로세스들에 의해 N보다 많은 수의 페이지-정보들이 일관성 볼륨의 원형 리스트에 쓰여진 경우에 발생한다. 따라서, 서버 프로세스의 최근-접근-타임스탬프는 매우 큰 값을 가지는 반면, 원형 리스트 내의 페이지-정보들의 타임스탬프는 매우 작은 값을 가진다. 앞에서 최근-접근-타임스탬프가 t_{head}보다 작은 경우에 원형 리스트의 랩 어라운드로 인한 문제가 발생한 것으로 판단한다고 하였다. 그러나 이 문제가 발생한 경우에는 최근-접근-타임스탬프가 원형 리스트의 어떠한 페이지-정보의 타임스탬프 값보다 크기 때문에 원형 리스트의 랩 어라운드로 인한 문제가 발생한 것을 발견하지 못한다.

서버 프로세스의 최근-접근-타임스탬프가 일관성 볼륨의 원형 리스트의 모든 페이지-정보들의 타임스탬프 값보다 큰 경우 이러한 문제가 발생한 것이다. 이러한 문제가 발생한 경우에는 원형 리스트의 랩 어라운드로 인한 문제에서와 마찬가지로 모든 버퍼 페이지들을 데이터베이스와 일치시킨다.

그러나, 이러한 방법은 오버플로우된 타임스탬프 값이 오버플로우되지 않은 최근-접근-타임스탬프만큼 커지는 경우를 처리하지 못한다. 그러나 이러한 경우는 실제 응용 프로그램에서는 일어날 가능성이 없다. 타임스탬프로서 4byte의 정수형을 사용할 경우, 초당 6개의 트랜잭션이 수행되는 환경에서, 이러한 문제가 발생하기 위해서는 특정 서버 프로세스가 약 11.3년 동안 트랜잭션을 수행하지 않다가 다시 트랜잭션을 수행해야 한다. 참고로 초당 6개의 트랜잭션이 수행되면 1년에 약 1억 9천만 개의 트랜잭션이 수행되는데, 이는 2006년 한국증권선물거래소의 거래 건수인 1억 7천만 건 보다 많다. 따라서, 이 문제는 실제 환경에서는 일어나지 않는 것으로 간주한다.

5. 성능 평가

본 장에서는 공유디스크 모델을 사용하는 다중 서버 DBMS를 사용하여 논문에서 제시한 버퍼 일관성 관리 방법(coherency)과 단순한 방법(naïve)의 성능을 비교한다. 여기서 단순한 방법은 볼륨 로크를 획득할 때마다 해당 볼륨 내에 있는 페이지들을 버퍼에서 무효화시키는 방법이다. DBMS로는 본 연구실에서 개발 중인 오디세우스 DBMS[5]를 사용하고, 성능 평가의 척도로는 트랜잭션당 평균 I/O 연산 횟수를 사용한다. 데이터로서는 TPC-C[6]에 정의된 데이터베이스와 트랜잭션을 사용한다.

본 실험에서는 데이터베이스의 크기가 790MB이고 버퍼의 크기가 80MB일 때, 프로세스의 개수를 2~32개로 증가시키며 이에 따른 성능 변화를 측정한다. 그림 2는 실험 결과로 논문에서 제시한 방법이 단순한 방법에 비해 1.2~2.9배 정도 성능이 향상되었음을 알 수 있다. 이때, 서버 프로세스 개수가 증가함에 따라 I/O 연산의 횟수가 증가하는 이유는 다른 서버 프로세스에서 수정된 페이지의 개수가 증가하므로 버퍼 일관성 관리를 위하여 버퍼에서 무효화되는 페이지 개수가 증가하기 때문이다. 참고로 무효화되는 페이지의 개수를 그림 2에 점선으로 표시하였다.

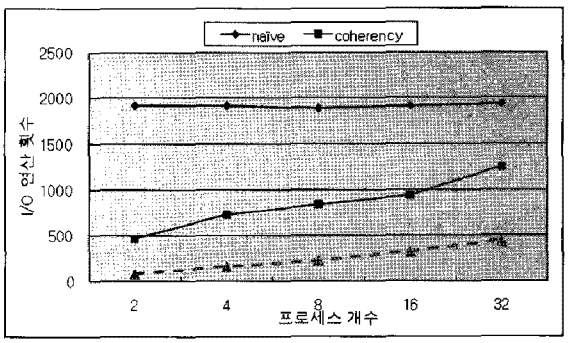

그림 2 서버 프로세스 개수에 따른 I/O 연산 횟수

6. 관련 연구

버퍼 일관성 관리에 대한 기존 연구들은 '통보, 전파, 탐색'이라 불리는 세 가지 어프로치들로 분류된다[4]. 통보는 트랜잭션 커밋 시 수정된 페이지들의 식별자들을 다른 서버 프로세스들에게 전달하는 방법이며, 전파는 수정된 페이지들을 전달하는 방법이다. 탐색은 버퍼 페이지를 액세스하기 이전에 그 버퍼 페이지가 유효한지를 검사하는 방법이다.

본 논문의 방법은 탐색에 속하지만, 탐색에 속하는 기존 방법들[4,7,8]과는 큰 차이가 있다. 첫째, 타임스탬프를 사용하여 제한된 개수의 페이지들에 대한 정보만을 유지하기 때문에 각 서버 프로세스 내의 버퍼 페이지들에 대하여 그 페이지가 유효한지 여부를 모두 유지하는 기존 방법들에 비해 자료 구조가 더 간결하다. 둘째, 큰 단위 로크를 사용하는 경우 별도의 프로세스를 사용하지 않고도 일관성 볼륨을 직접 액세스함으로써 일관성을 관리할 수 있기 때문에 별도의 프로세스를 사용하는 기존 방법들에 비해 안정성이 더 높다. 이때 일관성 볼륨의 디스크 접근 비용은 분산(amortize)되어 성능에 큰 영향을 미치지 않는다. 또한, 이 방법은 별도의 프로세스를 사용하여 일관성 볼륨의 자료 구조를 관리함으로써 페이지와 같은 작은 단위 로크를 사용하는 경우에도 적용될 수 있으며, 이 경우에도 간결한 자료 구조로 인하여 기존 방법들보다 효율적인 관리가 가능하다.

7. 결 론

본 논문에서는 공유디스크 모델을 이용한 다중 서버 DBMS에서 발생되는 버퍼 일관성 문제를 해결하는 방법을 제안하였다. 제안한 방법은 트랜잭션 커밋 시에 서버 프로세스들이 수정한 페이지들의 정보를 일관성 볼륨에 저장하고, 로크를 획득할 때 이 볼륨에서 다른 프로세스에서 수정된 페이지에 대한 정보를 읽은 후 해당 페이지를 버퍼에서 무효화시킴으로써 버퍼 일관성 문제를 해결한다. 이 방법은 매우 작은 크기의 일관성 볼륨만을 사용하고, 액세스하는 데이터의 양이 적어서 성능이 매우 빠르다.

참 고 문 헌

[1] Kim, T., Design and Implementation of a Buffer-Sharing Mechanism for Extended Single-User DBMSs Using Coarse-Granularity Locking, Master's Thesis, Department of Computer Science, KAIST, June 2005.

[2] Stonebraker, M., "The Case for Shared Nothing," *A Quarterly Bulletin of the IEEE Computer Society Technical Committee on Database Engineering*, vol.9, no.1, pp.4-9, Mar. 1986.

[3] Bamford, R., Ahad, R., and Pruscino, A., "A Scalable and Highly Available Networked Database Architecture," In *Proc. 25th Int'l Conf. on Very Large Data Bases (VLDB)*, pp.199-201, Sept. 1999.

[4] Dan, A. and Yu, P., "Performance Analysis of Buffer Coherency Policies in a Multisystem Data Sharing Environment," *IEEE Trans. on Parallel and Distributed Systems*, vol.4, no.3, pp.289-305, Mar. 1993.

[5] Whang, K. Lee, M., Lee, J., Kim, M., and Han, W., "Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features," In *Proc. 21st Int'l Conf. on Data Engineering (ICDE)*, pp.1104-1105, Demonstration Paper, Apr. 2005.

[6] Transaction Processing Performance Council (TPC), TPC BenchmarkTM C Standard Specification, Revision 5.9, June 2007.

[7] Cho, H., "Cache Coherency and Concurrency Control in a Multisystem Data Sharing Environment," *IEICE Trans. on Information and Systems*, vol.E82-D, no.6, pp.1042-1050, 1999.

[8] Mohan, C. and Narang, I., "Efficient Locking and Caching of Data in the Multi-system Shared Disks Transaction Environment," In *Proc. Int'l Conf. Extending Database Technology (EDBT)*, Mar. 1992.

고 현 선

2006년 KAIST 응용수학과 학사. 2008년 KAIST 전산학과 석사. 2008년~현재 KAIST 전산학과 박사과정. 관심분야는 저장 시스템, 색인 구조

김 이 른

1999년 KAIST 전산학과 학사. 2001년 KAIST 전산학과 석사. 2001년~현재 KAIST 전산학과 박사과정. 관심분야는 저장시스템, 임베디드 시스템

이 민 재

1995년 KAIST 전산학과 학사. 1997년 KAIST 전산학과 석사. 2004년 KAIST 전산학과 박사. 관심분야는 공간 데이터베이스, 공간 색인 구조, 데이터베이스 관리 시스템, 질의 처리 시스템, 정보 검색

황 규 영

1973년 서울대학교 전자공학과 졸업(학사). 1975년 KAIST 전기 및 전자학과 졸업(석사). 1982년 Stanford University EE/CSL(석사). 1984년 Stanford University EE/CSL(박사). 1975년~1978년 국방과학연구소 선임연구원. 1983년~1990년 IBM T.J. Watson Research Center, Research Staff Member. 1992년~1994년 한국정보과학회 데이터베이스 연구회(SIGDB) 운영위원장. 2007년 한국정보과학회 회장. 2007년~현재 Fellow, IEEE. 2007년~2009년 Coordinating Editor-in-Chief, The VLDB Journal (2003년~2009년 Editor-in-Chief). 2002년~2005년 Associate Editor, IEEE Trans. on Knowledge and Data Engineering. 1990년~2003년 Associate Editor, The VLDB Journal. 1990년~1993년 Associate Editor, The IEEE Data Engineering Bulletin. 1996년~2004년 Trustee, The VLDB Endowment. 2007년~2009년 Chair, Steering Committee, DASFAA. 1990년~현재 KAIST 전산학과 교수. 2008년~현재 KAIST 특훈교수. 관심분야는 데이타베이스 시스템, 멀티미디어, 검색엔진, GIS