디스크를 공유하는 다중 시스템 상에서 캐쉬 일관성 유지를 위한 동적 PCA 할당

김신희
대경대학 컴퓨터정보과
류명춘
경운대학교 컴퓨터공학과
박정량
영남대학교 컴퓨터공학과

초 록

데이타베이스 공유 시스템에서는 동일한 페이지가 여러 처리노드에 의해 동시에 캐싱될 수 있으므로, 각 처리노드가 항상 최신의 내용을 참조하기 위해서는 캐싱된 데이타의 일관성이 유지되어야 한다. 본 논문에서는 로킹 오버헤드를 줄이기 위해 주사본 권한을 이용하여 데이타베이스를 논리적으로 분할한 데이타베이스 공유 시스템 환경에서 필요한 캐쉬 일관성 기법들을 제안한다. 제안한 기법들인 DPCA_P와 DPCA_U는 PCA를 동적으로 할당하여 캐쉬 일관성을 위해 소요되는 메시지 전송량과 디스크 입출력 오버헤드를 줄임으로써 성능을 향상시키며, 데이타베이스 부하가 동적으로 변하는 경우에도 효율적으로 동작한다는 장점을 갖는다.

I. 서론

데이타베이스 공유 시스템(database sharing system)은 고성능의 온라인 트랜잭션 처리가 필요한 응용분야를 효율적으로 지원하기 위하여 제안된 분산 트랜잭션 처리 시스템이다. 별도의 메모리와 운영체제, 그리고 DBMS를 가지는 소결합된 여러 개의 처리노드로 구성되며 각각의 처리노드는 디스크 레벨에서 하나의 데이타베이스를 공유한다. 뿐만 아니라, 처리노드들은 물리적으로 인접한 위치에 존재하며, 고속의 통신 시스템을 이용하여 메시지 교환을 통해 교신한다. 데이타베이스 공유 시스템은 시스템 내의 모든 처리노드가 하나의 데이타베이스를 공유하므로 임의의 처리노드에서 전체 데이타베이스에 바로 접근할 수 있다(Rahm, 1992; Rahm, 1993; Yu, 1994). 또한 각각의 처리노드는 자신의 버퍼에 최근에 입출력한 페이지들을 캐싱한다. 처리노드가 항상 최신의 데이타를 사용할 수 있기 위해서는 버퍼에 캐싱된 데이타의 일관성이 유지되어야 한다(Dan, 1993; Dan, 1992; Mohan, 1991). 이를 위해 버퍼 관리자는 캐쉬 일관성 기법을 지원하여야 하며, 이러한 캐쉬 일관성 기법은 동시성 제어 기법과 밀접한 관계를 갖는다.

데이타베이스 공유 시스템에서는 동시성 제어를 위해 분산형 2-단계 로킹(two-phase locking: 2PL) 기법이 많이 이용된다. 분산 기법은 데이타베이스를 논리적으로 분할하여 각 분할에 대한 로킹 정보를 데이타베이스 공유 시스템을 구성하는 처리노드들에 나누어 저장한다(Rahm, 1986; Rahm, 1993). 이때 각 처리노드에 할당된 데이타베이스 분할에 관한 권한을 주사본 권한 (Primary Copy Authority: PCA)이라 부른다(Reuter, 1986). 처리노드가 참조하는 데이타는 자신이 PCA를 가지고 있는 지역 데이타와 다른 처리노드에게 PCA가 할당된 원격 데이타로 나눌 수 있다. 중앙집중형 기법에 비해 분산 기법은 지역 데이타를 참조하는 경우 로킹에 관련된 통신이 필요없으므로 성능이 향상되고, 로킹 정보가 여러 처리노드에 나누어 저장되므로 신뢰성을 향상시킬 수 있다.

분산 기법이 최적의 성능을 나타내기 위해서는 각 데이타의 최신 버전이 그 데이타에 대한 PCA를 가지고 있는 처리노드의 버퍼에

캐싱되어 있을 확률이 커야 한다. 이를 위해 Rahm은 다음과 같은 두가지 해결책을 제시하였다(Rahm, 1986). 첫째로, 페이지를 갱신한 트랜잭션이 완료할 때 해당 페이지의 최신 내용을 공유 디스크에 저장하는 방법이다. 그러나, 이 방법은 빈번한 공유 디스크 입출력으로 인해 성능이 더욱 저하될 수 있다. 실제로, Mohan(1991)과 Rahm(1993)은 네트워크를 통해 페이지를 전송하는 것이 공유 디스크를 이용하는 것보다 약 50배 정도의 좋은 성능을 보인다고 주장하고 있으며, 이러한 관점은 클라이언트-서버 DBMS 분야에서도 제안된 바 있다(Franklin, 1997).

또 다른 방법은 페이지를 갱신한 트랜잭션이 완료할 때 최신 내용을 PCA 처리노드에게 전송한 후, PCA 처리노드가 자신의 버퍼에 이 내용을 캐싱하는 것이다. 이후 해당 페이지에 대한 모든 최신 버전의 요청은 PCA 처리노드에서 직접 처리할 수 있다. 그러나, 이 방법은 PCA 처리노드로 최신 페이지를 전송하기 위한 메시지 전송량이 많아질 수 있다. 또한 PCA 처리노드에서 최신 페이지를 캐싱하기 위한 페이지 교체가 빈번히 발생할 수 있다. 특히 처리노드의 버퍼 용량이 적거나 빈번한 데이타 갱신이 발생할 경우, PCA 처리노드는 자신이 참조하는 페이지와 다른 처리노드에서 전송된 페이지들을 동시에 캐싱할 수 없게 되어 결국 과도한 페이지 부재로 인한 스래싱(thrashing) 현상이 발생할 수 있다.

이러한 디스크 입출력 오버헤드와 각 처리노드 간의 메시지 전송량을 줄이기 위해서는 각 분할에 대한 PCA를 동적으로 관리할 필요가 있다(Kim, 1998). 본 논문에서는 데이타베이스를 공유하는 환경에서 PCA를 이용하여 분산 동시성 제어 기법을 지원하는 캐쉬 일관성 기법들을 제안한다. 제안된 기법들은 PCA를 동적으로 관리함으로써 갱신 트랜잭션 완료시에 PCA 처리노드로 갱신된 페이지를 전송하기 위한 메시지 오버헤드를 줄인다. 뿐만 아니라, 새로운 처리노드가 시스템에 포함되거나 기존 처리노드의 고장 발생시, 혹은 데이타베이스 참조 형식이 변할 때와 같은 시스템 환경의 변화에 대한 적응성을 제공하며, 처리노드들 간에 부하 불균형 문제를 해결함으로써 고성능의 트랜잭션 처리를 가능하게 한다.

논문의 구성은 다음과 같다. 2장에서는 본 논문에서 구성한 시스템의 제반 환경에 대해 기술한다. 3장에서 제안한 캐쉬 일관성 기법에 대해 기술하고, 각 기법들의 오버헤드를 비교하고 분석한다. 4장에서 각 캐쉬 일관성 기법의 성능 평가를 위한 모의실험 환경과 모의실험 결과에 대해 기술한다. 5장에서 본 논문의 연구 결과에 대한 결론을 맺고, 앞으로의 연구 방향에 대해 기술한다.

II. 시스템 구성

이 장에서는 본 논문에서 구성한 시스템의 제반 환경에 대해 기술한다.

2.1 PCA 관리

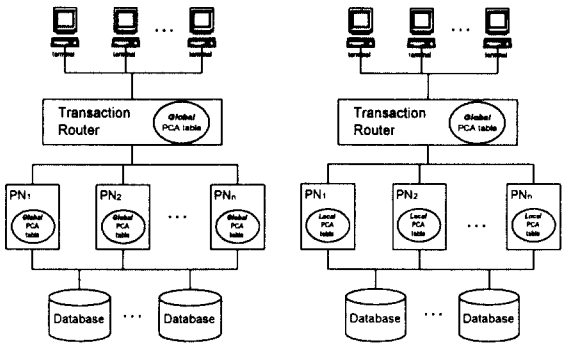
PCA 관리 방식은 PCA의 변화를 쉽게 지원하는가 혹은 그렇지 않은가에 따라 PCA 중복 관리와 PCA 분할 관리로 나누어질 수 있다.

PCA 중복 관리는 시스템 전체의 PCA 할당에 관한 정보를 카탈로그(catalog) 형식으로 구성하여 모든 처리노드에 중복시키는 방식이다(Rahm, 1986). 이 방식에서는 시스템 구성 초기에 일단 카탈로그가 만들어지면 시스템 구성에 대한 물리적인 변동사항이 발생하지 않는 한 카탈로그 정보를 갱신하지 않는다. 즉, PCA는 정적으로 유지된다. 각 처리노드는 트랜잭션의 로크 요청이 발생하면 카탈로그를 참조하여 지역 데이타에 대한 요청이면 바로 처리하고, 그렇지 않으면 해당 PCA 처리노드로 로크 요청 메시지를 전송하여 처리한다. 이 방식은 기존의 PCA를 이용한 로킹 기법에서 사용된다.

PCA 분할 관리 방식은 전체 PCA 할당 정보에 대한 카탈로그를 분할하여 각 처리노드로 하여금 분할된 지역 카탈로그만을 관리하도록 함으로써 PCA를 동적으로 할당할 수 있게 한다. 즉, 각 처리노드는 지역 PCA 정보만을 가지므로 시스템 운영 도중에 임의의 처리노드에서 다른 처리노드로 PCA를 재할당하더라도 해당하는 두 처리노드의 지역 카탈로그만 갱신하면 되므로 나머지 처리노드에는 아무런 영향을 미치지 않는다. 본 논문에서 제안한 캐쉬 일관성 기법들은 이 방식을 이용한다.

<그림 1>은 PCA 관리 방식에 따라 구성된 시스템 예이다. <그림 1(a)>는 PCA 중복 관리 방식으로 분산 동시성 제어 기법을 구현할 경우의 시스템 구성도로서, 전체 시스템의 PCA 정보를 저장한 전역 PCA 테이블이 모든 처리노드에 의해 중복되어 정적으로 유지된다. <그림 1(b)>는 PCA를 동적으로 할당하는

PCA 분할 관리 방식을 위해 구성한 시스템이다. PCA 중복 관리 방식과 달리, 전역 PCA 테이블은 트랜잭션 라우터(Transaction Router: TR) 처리노드에만 유일하게 존재하며, 각 처리노드는 자신에게 할당된 지역 PCA 정보만을 유지한다.



(a) PCA 중복 관리 (b) PCA 분할 관리

<그림 1> PCA 관리 방식에 따른 시스템 구성

2.2 전역 동시성 제어

전통적인 중앙집중형 데이타베이스 시스템에서와 같이 데이타베이스 공유 시스템에서 전역 동시성 제어 기법은 트랜잭션의 직렬화 가능성(serializability)을 보장하여야 하고, 처리노드 간의 메시지 전송을 최소화함으로써 고성능의 트랜잭션 처리를 가능하도록 해야 한다. 본 절에서는 데이타베이스 공유 환경에서 동시에 실행되는 트랜잭션들에 대해 데이타의 일관성을 유지하기 위한 전역 동시성 제어 방식에 대해 설명한다. 본 연구에서 고려하는 PCA를 이용한 로킹 기법은 기본적으로 2-단계 로킹 방식을 사용하여 동시에 실행되는 트랜잭션들의 직렬화 가능성을 보장한다.

트랜잭션은 액세스하는 페이지와 액세스 유형으로 구성된 연산 리스트 형태로 터미널에서 발생하며 트랜잭션 경로 배정기를 거쳐 적절한 노드로 배정된다. 노드 내의 트랜잭션 스케줄러는 전역 로크 테이블과 지역 로크 테이블을 관리한다. 전역 로크 테이블은 전역 로크 요청을 처리하기 위해 자신이 PCA를 가진 페이지에 대한 시스템 전체의 로킹 정보를 포함하고, 지역 로크 테이블은 지역 로크 요청을 처리하기 위해 필요한 로킹 정보를 포함한다. 로킹은 페이지 단위로 이루어진다고 가정한다. 트랜잭션은 연산 리스트의 순서로 실행되는데 만약 액세스할 페이지에 대해 자신이 PCA 노드이면 지역적으로 바로 처리하고, 그렇지 않으면 전역 로크 요청 메시지를 TR로 전송한다. TR 내의 PCA 테이블 관리자는 전역 로크 요청 메시지에 대한 PCA 노드를 식별하여 해당 노드로 메시지를 전송하여 로크 요청을 처리할 수 있게 한다.

전역 로크 요청 메시지는 <노드 식별자, 트랜잭션 식별자, 페이지 식별자, 로크 모드, 지역 버퍼내에 캐싱하고 있는 페이지의 page_LSN> 으로 구성된다. 이때 로크 모드는 판독 연산일 경우 "S"로 설정되며, 갱신 연산일 경우 "X"로 설정된다. page_LSN은 그 페이지를 최종적으로 갱신한 연산에 대한 로그의 일련 번호를 저장하는 필드이다(Mohan, 1992).

본 논문에서는 판독 로크에 대한 로크 캐싱을 수행한다. 즉, 각 처리노드는 트랜잭션 완료시에 판독 로크에 대해서는 바로 해제하지 않고 PCA 처리노드로부터 로크 해제 요청 메시지가 들어올 때까지 계속 유지한다. 일반적으로 판독 연산이 갱신 연산에 비해 자주 실행되며 판독 로크는 공유될 수 있으므로, 판독 로크만 캐싱함으로써 로크 캐싱에 따른 오버헤드를 최소화할 수 있으며 성능을 최적화할 수 있다. 동일한 주장이 클라이언트-서버 DBMS 분야에서도 제안된 바 있다(Fraklin, 1997).

III. 캐쉬 일관성 기법

본 장에서는 데이타베이스 공유 환경에서 기존의 캐쉬 일관성 기법과 PCA 분할 관리 방식을 이용하여 디스크 입출력 오버헤드와 처리노드 간의 메시지 전송량을 줄이기 위한 기법들을 제안한다.

3.1 정적 캐쉬 일관성 기법 (SPCA)

SPCA는 PCA가 정적으로 유지될 때 사용되는 기법으로, (Rahm, 1986)에서 제안되었다. 기본 개념은 임의의 처리노드에서 갱신 트랜잭션이 완료될 때마다 PCA 처리노드로 갱신된 페이지를 전송한다는 것이다. PCA 처리노드는 전송받은 페이지를 자신의 버퍼에 캐싱하고, 이 페이지에 대한 이후의 접근 요청은 PCA 처리노드에서 바로 처리할 수 있다. 그러나, SPCA는 다음과 같은 몇 가지 문제점을 가지고 있다. 먼저 PCA 처리노드로 전송되는 로크 해제 메시지에 갱신된 페이지를 포함해야 하기 때문에 통신 오버헤드가 증가한다. 뿐만 아니라, 로크 해제 메시지를 전송하는 처리

노드에서 추가적인 디스크 입출력이 발생할 수도 있다. 마지막으로, 빈번한 갱신이 발생할 경우 PCA 처리노드는 자신이 참조하는 페이지와 다른 처리노드로부터 전송된 모든 페이지를 동시에 캐싱할 수 없게 된다. 따라서, 과도한 페이지 부재로 인한 스래싱 현상이 발생할 수 있고, 디스크 입출력의 빈도수가 급격히 증가할 수 있다.

3.2 제안한 캐쉬 일관성 기법

SPCA의 문제점을 해결하기 위하여 본 절에서는 PCA의 동적 할당을 지원하는 캐쉬 일관성 기법들을 제안한다. 제안한 캐쉬 일관성 기법들은 PCA를 재할당하는 조건에 따라 나누어진다.

3.2.1 페이지 교체시 PCA 재할당 (DPCA_P)

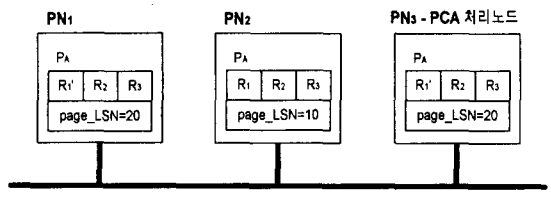
DPCA_P는 SPCA에서 발생할 수 있는 스래싱 현상을 피하기 위해 제안된 기법이다. DPCA_P의 기본 개념은 PCA 처리노드에서 페이지 교체시에 해당 페이지에 대한 PCA를 재할당함으로써 디스크 입출력 횟수를 줄인다는 것이다. 즉, 버퍼 관리자가 새로운 페이지를 버퍼에 캐싱하기 위해 기존의 페이지를 교체할 때, 자신이 교체될 페이지에 대한 PCA 처리노드라면 최신 페이지를 캐싱하고 있는 다른 처리노드로 PCA를 재할당한다. 이로써, 교체전에 굳이 디스크에 기록할 필요가 없고, PCA는 항상 최신 페이지를 가진 처리노드에 할당되므로 이후의 페이지 참조시에도 디스크로부터 읽어 들일 필요가 없다. 물론 PCA 재할당을 위한 메시지 오버헤드가 있지만 디스크 입출력으로 인한 오버헤드보다는 크지 않다. PCA 처리노드에서 페이지가 교체됨으로 인한 디스크 입출력은 PCA를 재할당할 처리노드가 존재하지 않는 경우에만 일어나며, 이때 PCA 재할당은 고려되지 않는다.

3.2.2 갱신 요청시 PCA 재할당 (DPCA_U)

DPCA_U의 기본 개념은 갱신 트랜잭션의 완료시에 갱신된 페이지의 전송을 하지 않도록 하는 것이다. 이를 위해 PCA 처리노드로 X 로크 요청이 들어오면 요청한 처리노드에 로크 허용과 동시에 PCA를 재할당한다. 따라서 SPCA나 DPCA_P와는 달리 갱신 트랜잭션이 완료될 때 페이지를 전송할 필요가 없기 때문에 메시지 전송 오버헤드가 적고, PCA 처리노드에서의 페이지 교체율도 줄일 수 있다. 예 1은 DPCA_U에서 PCA가 재할당되는 경우를 보여주고 있다.

예 1: (DPCA_U에서 PCA 재할당)

<그림 2>에서 처리노드 PN_2가 P_A를 갱신할 경우를 생각해보자. PN_2는 P_A의 PCA 처리노드인 PN_3에게 P_A에 대한 X 로크를 요청하게 되고, 로크가 허용될 경우 PN_3는 PN_2에게 P_A의 최신 버전을 전송할 것이다. 이때 SPCA와 DPCA_P에서는 PN_3가 P_A의 PCA를 계속



<그림 2> 세 개의 처리노드로 구성된 데이타베이스 공유 시스템 예

유지하며, PN_2에서 P_A를 갱신한 트랜잭션이 완료할 경우 P_A의 최신 버전이 PN_3로 다시 전송되게 된다. 그러나, DPCA_U에서는 P_A에 대한 X 로크가 허용될 때 P_A의 PCA 처리노드가 PN_2로 변경되므로, 이후 P_A의 최신 버전이 PN_3로 전송될 필요가 없다.

IV. 성능 평가

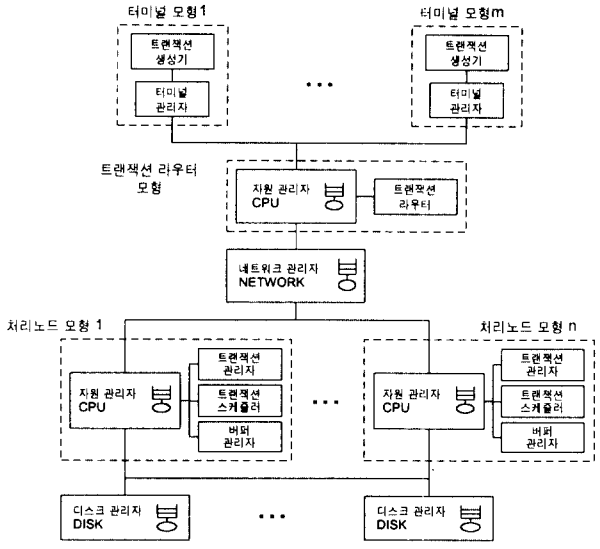
본 장에서는 각 캐쉬 일관성 기법에 대한 성능 평가를 위한 모의실험 환경에 대해 설명하고 실험 결과를 분석한다.

4.1 모의실험 환경

제안한 캐쉬 일관성 기법들의 성능을 평가하기 위한 모의실험 환경은 미국의 MCC에서 개발한 이산 사건 모의실험 패키지인 CSIM 언어(Schwetman, 1992)를 이용하여 구현되었고, 실험은 SunOS Release 5.5를 운영체제로 가지는 SUN-HYPER SPARC에서 수행되었다.

모의실험을 위한 공유 데이타베이스 환경이 <그림 3>에 나타나 있다. 본 논문에서 사용한 입력 매개 변수는 (Carey, 1994)를 참조하였다. 이러한 모의실험 환경은 (Kim, 1998)에서의 환경을 그대로 적용하였다.

본 논문에서는 성능 평가 지수로서 트랜잭션 처리율을 사용한다. 트랜잭션 처리율은 초



<그림 3> 모의실험 모형

당 완료되는 트랜잭션 수를 의미한다. 또한 보조 성능 평가 지수로 트랜잭션당 평균 디스크 입출력 횟수를 사용하였다. 그 이유는 트랜잭션의 응답 시간이 대략적으로 로크 요청을 처리함에 따른 지연시간, 페이지 전송에 따른 부담시간, 그리고 디스크 입출력 시간으로 구성된다고 보았을 때, (Mohan, 1991)에서 지적한 바와 같이 디스크 입출력 시간이 트랜잭션 처리율에 미치는 영향이 가장 크기 때문이다.

본 논문에서 각 기법들에 대한 모의실험은 보다 정확한 결과를 얻기 위해 완료된 트랜잭션 수가 2,000개가 될 때까지 수행하며, 초기 200개가 완료될 때까지의 결과들은 무시하였다. 또한, 신뢰성있는 모의실험 결과를 얻기 위해 배치 평균 기법(batch mean method)을 이용하였다. 본 논문에서 나타난 실험 결과들은 30개의 다른 seed를 이용하여 산출된 결과들의 평균값이다. 이러한 기법을 이용하여 산출된 결과들은 90 퍼센트의 신뢰 수준을 만족하였다.

4.2 결과 분석 및 검토

본 절에서는 모의실험 모형을 이용하여 캐쉬 일관성 기법들에 대해 수행한 실험 결과를 분석하고 성능을 평가한다. 다양한 작업 환경에서 캐쉬 일관성 기법들의 성능을 분석하기 위하여, 데이타베이스 접근 유형을 분할 접근 환경과 높은 충돌 환경으로 나누어 실험을 하였다. 각 환경에서의 모의실험은 터미널 수를 100, 갱신 연산 비율을 20%로 고정하고 처리 노드 수를 변화시켜 가면서 수행하였다.

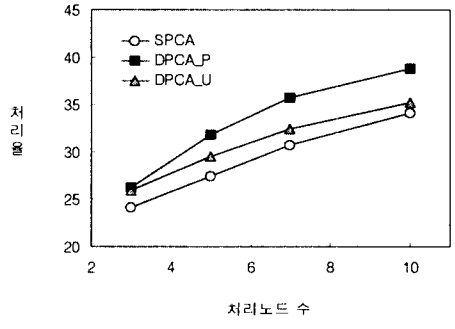
4.2.1 분할 접근 환경

분할 접근 환경은 트랜잭션들이 접근 유형에 따라 몇 개의 그룹으로 나누어지며, 각 그룹에 속한 트랜잭션들은 특정 데이타를 빈번하게 참조하는 경우이다. 본 논문에서는 각 그룹에 할당된 데이타를 참조하는 비율에 따라 분할 접근 환경을 낮은 참조 지역성 환경과 높은 참조 지역성 환경으로 세분화하여 실험을 하였다.

4.2.1.1 낮은 참조 지역성

이 환경에서는 트랜잭션이 실행하는 연산의 20%만 지역적으로 처리되고, 나머지 80%는 전역적으로 처리되며, 트랜잭션이 공유 데이타베이스에 무작위로 접근하는 경우에 해당한다. 따라서 전역 로크 처리가 많아지게 되고 PCA 처리노드로의 페이지 접근 요청이 빈번히 발생한다. 그 결과로, 각 처리노드의 지역 버퍼의 페이지 교체율도 증가한다.

<그림 4>는 낮은 참조 지역성 환경에서의 트랜잭션 처리율을 나타내며, 이때의 트랜잭션당 평균 디스크 입출력 횟수가 <그림 5>에 나타난다. 처리노드의 수가 증가할수록 트랜잭

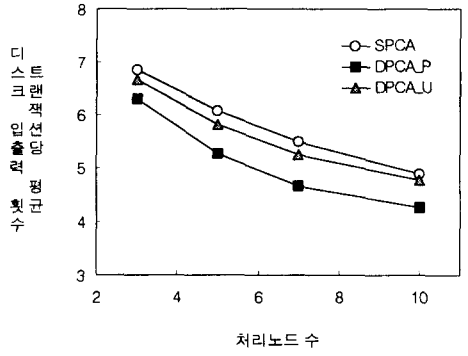


<그림 4> 낮은 참조 지역성 환경에서 트랜잭션 처리율

션 처리율은 증가하였다. 그 이유는 시스템 전체의 관점에서 보았을 때 처리노드의 수가 증가할수록 사용가능한 버퍼 크기가 증가하므로 참조하고자 하는 페이지를 지역 버퍼에서 발견할 가능성이 커지기 때문이다. 뿐만 아니라, 처리노드 수만큼 공유 데이타베이스를 논리적으로 분할하여 처리노드에게 페이지에 대한 PCA를 할당하므로 처리노드 수가 많을수록 각 처리노드에게 할당되는 PCA 수가 적다. 따라서 PCA 처리노드의 버퍼에서 최신 페이지를 캐싱하고 있을 확률이 높기 때문에 디스

크 입출력 횟수를 줄일 수 있다.

PCA를 정적으로 유지하는 SPCA에 비해 DPCA_P가 약 20%, DPCA_U가 약 10% 정도 성능이 향상되었다. SPCA에서 각 처리노드는 별도의 PCA 할당 테이블을 가지므로 전역 로



<그림 5> 낮은 참조 지역성 환경에서
트랜잭션당 평균 디스크 입출력 횟수

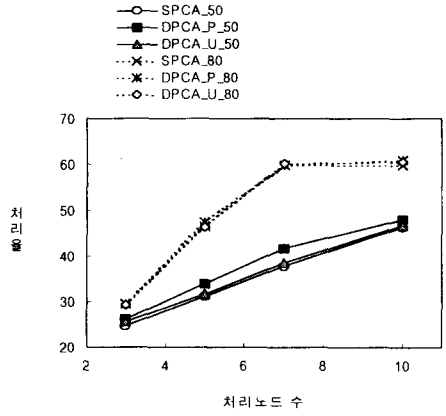
크 요청 및 해제시에 TR 처리노드를 거칠 필요없이 바로 해당 PCA 처리노드로 메시지를 전송가능하므로 로크 요청 및 해제와 관련된 통신 오버헤드는 DPCA_P와 DPCA_U에 비해 적다. 그러나 <그림 6>에 나타나듯이, SPCA에 비해 DPCA_P와 DPCA_U의 트랜잭션당 평균 디스크 입출력 횟수가 적다. 디스크 입출력에 대한 오버헤드가 메시지 전송 오버헤드보다 훨씬 크기 때문에, 디스크 입출력 횟수가 많은 SPCA의 성능이 다른 기법들에 비해 낮게 나타났다.

<그림 4>에서 보면 SPCA와 DPCA_P에 비해 DPCA_U는 처리노드 수가 증가함에 따라 성능 향상 정도가 비교적 완만하다. 왜냐하면 처리 노드가 많아질수록 DPCA_U는 PCA 재할당에 따른 부담이 증가하기 때문이다. 이에 반해 SPCA와 DPCA_P는 각 처리노드에 할당된 PCA 수가 적어 PCA 처리노드가 관리해야 될 페이지 수가 감소하므로 갱신된 페이지의 전송으로 인한 페이지 교체의 발생 가능성이 낮아진다.

4.2.1.2 높은 참조 지역성

이 환경은 트랜잭션의 접근 유형이 높은 참조 지역성을 가짐에 따라 하나의 PCA 처리노드에서 지역적으로 처리될 가능성이 높은 경우이다. 이를 위해 트랜잭션이 50%의 참조 지역성을 갖는 경우(트랜잭션이 참조하는 페이지의 50%가 한 처리노드에서 지역적으로 처리되고 나머지 50%에 대해서는 다른 처리노드에 의해 전역적으로 처리되도록 하는 경우)와 80%의 참조 지역성을 갖는 경우에 대해 각각 실험을 수행하였다.

<그림 6>은 높은 참조 지역성 환경에서의 트랜잭션 처리율을 나타내는데, 실선은 참조



<그림 6> 높은 참조 지역성 환경에서
트랜잭션 처리율

지역성이 50%일 경우이며 점선은 참조 지역성이 80%일 경우이다.

참조 지역성이 낮을 때에 비해 모든 캐쉬 일관성 기법들의 성능이 향상되었다. 그 이유는 참조 지역성이 높아지면 대부분의 로크가 PCA 처리노드에 의해 지역적으로 처리되므로 메시지 오버헤드가 감소하기 때문이다. 뿐만 아니라, 각 처리노드에서 실행되는 트랜잭션들이 공통된 페이지를 참조할 가능성이 높아지기 때문에 최신 페이지가 PCA 처리노드의 버퍼에 캐싱되어 있을 확률도 높아지게 된다.

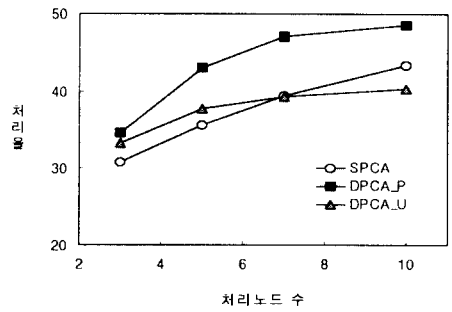
참조 지역성이 커질수록 캐쉬 일관성 기법들간의 차이가 줄어들었으며, 참조 지역성이 80%인 경우 캐쉬 일관성 기법들의 성능이 거의 동일하게 나타났다. SPCA와 본 논문에서 제안한 기법들간의 차이는 버퍼 공간의 부족으로 인해 페이지 교체가 빈번하게 발생할 경우(DPCA_P), 혹은 PCA 처리노드가 아닌 처리노드에서 갱신이 발생할 경우(DPCA_U)에서 나타난다. 그러나 참조 지역성이 높아지면, 각각의 경우들이 발생할 가능성이 낮아지므로 그만큼 캐쉬 일관성 기법들간의 차이도 줄어들게 된다.

4.2.2 높은 충돌 환경

높은 충돌 환경은 각 처리노드에서 실행되는 트랜잭션들이 데이타베이스의 특정 부분을

높은 확률로 참조하는 경우를 모델링한 것이다. 이 환경에서 각 트랜잭션이 실행하는 연산의 60%는 전체 데이타베이스의 20%에 해당하는 특정부분을 참조하며, 연산의 40%는 나머지 데이타베이스를 무작위로 참조한다.

<그림 7>은 높은 충돌 환경에서 처리노드 수의 변화에 따른 트랜잭션 처리율을 나타내



<그림 7> 높은 충돌 환경에서 트랜잭션 처리율

고 있다. 분할 접근 환경에서 참조 지역성이 20%와 50%일 때에 비해 처리율은 더 높게 나타났다. 특히 처리노드 수가 3일 때의 트랜잭션 처리율은 높은 참조 지역성(80%)의 경우보다도 성능이 우수하였다. 그 이유는 데이타베이스의 특정 부분이 집중적으로 참조되므로 PCA 처리노드의 버퍼에 캐싱되어 있을 확률이 높기 때문이다. 또한 시스템 전체의 사용가능한 버퍼 크기는 처리노드 수에 비례하는데, 높은 충돌 환경의 경우 처리노드 수가 적을 때에도 빈번히 참조되는 페이지가 PCA 처리노드의 버퍼에 캐싱되어 있을 확률이 높다. 높은 충돌 환경에서도 DPCA_P는 SPCA에 비해 최신 페이지를 캐싱할 확률이 높으므로 성능이 우수하다.

V. 결론

본 논문에서는 데이타베이스 공유 환경에서 PCA를 동적으로 할당하는 개념을 이용하여 디스크 입출력 횟수를 줄일 수 있는 새로운 캐쉬 일관성 기법인 DPCA_P와 DPCA_U를 제안하였다. PCA를 정적으로 유지하는 환경에서 제안된 기존의 캐쉬 일관성 기법(SPCA)은 PCA 처리노드에서 최신 페이지를 제공할 수 있도록 하기 위해 갱신 트랜잭션이 완료될 때마다 갱신된 페이지를 PCA 처리노드로 전송하는데, 이로 인해 디스크 입출력을 수반하는 페이지 교체가 발생하고, 페이지 전송에 따른 오버헤드도 커진다. 이에 대해 DPCA_P는 PCA 처리노드에서 페이지 교체가 발생할 경우, 교체될 페이지의 최신 버전을 캐싱하고 있는 다른 처리노드로 PCA를 재할당한다. 따라서 PCA 처리노드의 버퍼에 최신 페이지를 캐싱할 확률을 높임으로써 디스크 입출력 횟수를 줄일 수 있다. DPCA_U는 갱신 로크 요청에 응답할 때 해당 페이지를 요청한 처리노드로 PCA를 재할당하기 때문에 갱신 트랜잭션이 완료될 때 갱신된 페이지를 전송함에 따른 페이지 교체를 고려할 필요가 없고, 페이지 전송에 따른 메시지 오버헤드도 줄일 수 있다.

제안된 기법들의 성능을 분석하기 위하여 데이타베이스 공유 환경을 위한 모의실험 모형을 개발하였고, 다양한 데이타 접근 환경에서 실험을 하였다. 트랜잭션의 데이타 접근 유형이 낮은 참조 지역성을 가질 경우, SPCA에 비해 DPCA_P가 약 20%, DPCA_U가 약 10% 정도 성능이 향상된다. 그러나 참조 지역성이 증가할수록 캐쉬 일관성 기법들 간의 성능 격차는 점차 줄어든다. 대부분의 데이타베이스 공유 시스템의 응용 분야에서 트랜잭션의 참조 지역성은 30% 내외(Rahm, 1993)라는 사실을 감안할 때, 낮은 참조 지역성 환경에서 좋은 성능을 보이는 것은 매우 중요하다. 높은 충돌 환경에서는 분할 접근 환경에서 참조 지역성이 20%일 때와 50%일 때보다 최신 페이지를 캐싱할 확률이 높아 성능이 우수하다. 특히 처리노드 수가 적을 때 높은 충돌 환경에서의 성능은 분할 접근 환경에서보다 우수하다.

향후 혼성 PCA 관리 방식에 대해 연구하고 여러 환경에서의 모의실험을 통해 PCA를 정적 또는 동적으로 관리하는 방식의 성능과 비교하고 분석하고자 한다.

참 고 문 헌

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

Dan, A. and Yu, P., "Performance Analysis of Buffer Coherency Policies in a Multisystem Data Sharing Environments," *IEEE Trans. on Parallel and Distributed Systems*, Vol.4, No.3, pp.289-305, 1993.

Dan, A. and Yu, P., "Performance Analysis

of Coherency Control Policies through Lock Retention," *Proc. of ACM SIGMOD,* pp.114-123, 1992.

Franklin, M., Carey, M., and Livny, M., "Transactional Client-Server Cache Consistency: Alternatives and Performance," *ACM Trans. on Database Systems,* 1997.

Kim, S., Cho, H. and Kang, B., "Distributed Cache Management in a Multisystem Data Sharing Environment with Dynamic Transaction Routing," to appear in *Int. Conf. on ITC-CSCC '98*

Mohan, C. and Narang, I., "Recovery and Coherency Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disks Transaction Environment," *Proc. of 17th VLDB Conf.,* pp.193-207, 1991.

Mohan, C. et al., "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," *ACM Trans. on Database Systems,* Vol.17, No.1, pp.94-162, 1992.

Rahm, E., "Primary Copy Synchronization for DB-Sharing," *Information Systems,* Vol.11, No.4, pp.275-286, 1986.

Rahm, E., "A Framework for Workload Allocation in Distributed Transaction Systems," *J. System Software,* Vol.18, No.3, pp.171-190, 1992.

Rahm, E., "Empirical Performance Evaluation of Concurrency and Coherency Control Protocols for Database Sharing Systems," *ACM Trans. on Database Systems,* Vol.18, No.2, pp.333-337, 1993.

Reuter, A., "Load Control and Load Balancing in a Shared Database Management System," *Proc. of 2nd IEEE Int'l. Conf. on Data Engineering,* pp.188-197, 1986.

Schwetman, H., *CSIM Users Guide for use with CSIM Revision 16,* MCC, 1992.

Yu, P. and Dan, A., "Performance Evaluation of Transaction Processing Coupling Architectures for Handling System Dynamics," *IEEE Trans. on Parallel and Distributed Systems,* Vol.5, No.2, pp.139-153, 1994.