공유 디스크 클러스터에서 버퍼 교체 알고리즘의 성능 평가

(Performance Evaluation of Disk Replacement Algorithms in a Shared Cluster)

조 행 래 †

(Haengrae Cho)

요 약 공유 디스크(Shared Disk: SD) 클러스터는 온라인 트랜잭션 처리를 위해 다수 개의 처리 노드들을 연동하는 방식으로, 모든 노드는 디스크 계층에서 데이타베이스를 공유한다. 빈번한 디스크 액세스를 피하기 위하여 각 노드는 자신의 메모리 버퍼에 최근에 액세스한 페이지들을 캐싱한다. 이때 동일한 페이지가 여러 노드의 메모리 버퍼에 동시에 캐싱될 수 있으므로 각 노드가 최신의 내용을 액세스하기 위해서는 캐싱된 페이지의 일관성이 유지되어야 한다. SD 클러스터에서 기존에 제안된 대부분의 캐쉬 일관성 기법들은 버퍼 교체 알고리즘으로 LRU를 가정하였다. 이와는 달리 본 논문에서는 SD 클러스터의 특징을 고려한 네 가지의 버퍼 교체 알고리즘들을 제안하고 성능을 평가한다. 클러스터 구성과 데이타베이스 부하를 다양하게 변경하면서 실험을 수행하였고, 제안한 알고리즘은 LRU에 비해 최대 5배까지 성능이 향상됨을 확인할 수 있었다.

키워드 : 트랜잭션 처리, 클러스터, 공유 디스크, 버퍼 교체, 캐쉬 일관성

Abstract A shared disk (SD) cluster couples multiple nodes for high performance transaction processing, and all the coupled nodes share a common database at the disk level. To reduce the number of disk accesses, each node caches database pages in its memory buffer. Since a particular page may be cached simultaneously in different nodes, cache consistency should be maintained to ensure that nodes can always access the most recent version of database pages. Most cache consistency schemes proposed in the SD cluster adopted LRU as a buffer replacement algorithm. In this paper, we first present four buffer replacement algorithms that consider the characteristics of the SD cluster. Then we compare the performance of the buffer replacement algorithms. We perform the experiments on a variety of cluster configurations and database workloads. The experiment results show that the proposed algorithms achieve performance improvement up to 5 times of LRU algorithm.

Key words : transaction processing, cluster, shared disks, buffer replacement, cache consistency

1. 서 론

클러스터는 다수 개의 연결된 노드들이 트랜잭션 처리를 위해 하나의 노드처럼 협력하는 시스템으로, 온라인 트랜잭션 처리와 전자 상거래, 그리고 병렬 데이타베이스 시스템 등 다양한 분야에 적용되고 있다[1]. 클러스터의 구성방식은 데이타를 액세스하는 방법에 따라 모든 노드에서 디스크를 공유하는 방식(Shared Disk: SD)과 메모리나 디스크를 공유하지 않는 방식(Shared Nothing: SN)으로 분류할 수 있다. SD 클러스터는 각 노드가 디스크 계층에서 전체 데이타베이스를 공유함으로써, 동적 부하 분산과 확장성이 용이하다는 장점을 가진다. 뿐만 아니라, SAN(storage area networks) 기술의 발전으로 높은 가용성과 데이타 액세스의 융통성을 지원하는 SD 클러스터의 장점이 한층 부각되고 있다. 이런 관점에서 IBM과 Oracle에서는 SD 클러스터를 위

· 이 연구는 2008학년도 영남대학교 학술연구조성비에 의한 것임

† 종신회원 : 영남대학교 전자정보공학부 교수
hrcho@yu.ac.kr

논문접수 : 2008년 8월 20일
심사완료 : 2008년 10월 13일

정보과학회논문지: 데이타베이스 제35권 제6호(2008.12)

한 병렬 데이타베이스 시스템으로 IBM DB2 Parallel Edition[2]과 Oracle Real Application Cluster[3] 등의 제품을 출시하고 있다.

SD 클러스터를 구성하는 각 노드는 자신의 메모리 버퍼에 최근에 액세스한 데이타 페이지들을 캐싱한다. 효율적인 캐싱은 디스크 액세스 수나 노드들 간 데이타 전송량을 줄임으로써 성능을 크게 향상시킬 수 있다. 노드들이 최신의 페이지를 항상 사용할 수 있기 위해서는 캐싱된 페이지의 일관성이 유지되어야 한다. 이를 위해 SD 클러스터에서는 캐쉬 일관성 기법을 지원하여야 하는데, 기본 개념은 한 노드에서 특정 페이지를 갱신할 경우 그 페이지의 이전 버전을 캐싱하고 있는 다른 노드들의 버퍼에서 무효화를 시킨다는 것이다[4,5].

그림 1은 본 논문에서 가정하고 있는 SD 클러스터의 구조이다. 트랜잭션 라우터는 사용자가 요구한 트랜잭션을 실행할 노드를 선택한 후, 해당 노드에게 트랜잭션의 실행을 요청한다. 본 논문에서는 친화도 기반의 트랜잭션 라우팅 정책[6,7]이 지원된다고 가정한다. 즉, 트랜잭션들을 데이타베이스 액세스 유형에 따른 친화도 클러스터(Affinity Cluster: AC)들로 분류하고, 각 AC마다 특정 노드를 할당한다. 동일한 AC에 속하는 트랜잭션들은 데이타베이스의 특정 부분을 빈번히 액세스하므로, 라우팅의 결과 각 노드의 지역 버퍼 히트율이 높아지며 버퍼 무효화와 로크 동기화에 의한 노드들 사이의 간섭을 줄일 수 있다.

각 노드의 클러스터 연동기능(Coupling Facility: CF)은 전역 로크 관리자(Global Lock Manager: GLM)와 함께 전역 동시성 제어 기법과 캐쉬 일관성 기법을 지원한다. 본 논문에서 가정하고 있는 전역 동시성 제어 기법과 캐쉬 일관성 기법은 2절에서 자세히 설명하도록 한다.

본 논문의 주제는 SD 클러스터에서 각 노드의 메모리 버퍼에 대한 버퍼 교체 알고리즘의 성능을 평가하는 것이다. 버퍼 교체 알고리즘은 컴퓨터공학에서 오랫동안 연구되어 온 주제이며, 데이타베이스 분야에서도 질의 처리의 성능 향상을 위해 활발히 연구되고 있다[8]. SD 클러스터에서의 버퍼 교체 알고리즘은 일반적인 버퍼 교체 알고리즘과 다음과 같은 차이점이 존재한다.

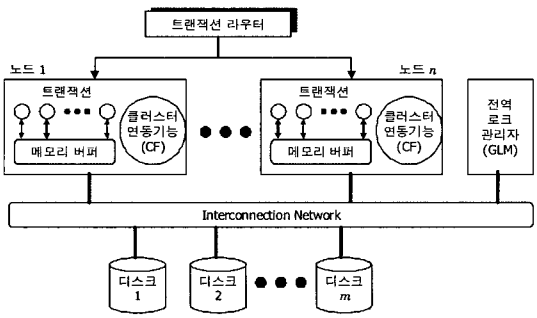

그림 1 SD 클러스터 구조

- SD 클러스터에서는 각 노드마다 별도의 메모리 버퍼를 갖는다. 이때 각 노드별로 자신의 액세스 패턴을 고려한 독자적인 버퍼 교체 알고리즘을 구현할 경우, 전체 시스템 관점에서 버퍼 효용이 감소할 수 있다. 예를 들면 동일한 데이타 페이지를 여러 노드에서 동시에 캐싱하고 있을 경우 메모리 버퍼의 총 용량이 낭비될 수 있다.
- SD 클러스터에서는 트랜잭션 라우팅 정책에 따라 각 노드별로 유사한 액세스 패턴을 갖는 트랜잭션들이 할당된다. 그 결과 노드별로 상이한 참조 지역성이 존재하며, 버퍼 교체 알고리즘은 이를 고려하여야 한다.
- SD 클러스터의 버퍼 교체 알고리즘은 캐쉬 일관성 기법과 강한 상관관계가 존재한다. 따라서 캐쉬 일관성 기법의 특성을 고려한 버퍼 교체 알고리즘이 지원되어야 한다.

첫 번째 차이점의 경우, NOW(Network of Workstation)를 위한 분산 캐싱[9-11]이나, 클라이언트-서버 시스템의 다단계 캐싱[12,13]에서 제안된 개념들을 일부 활용할 수 있으나, 두 번째와 세 번째 차이점을 고려한 버퍼 교체 알고리즘은 SD 클러스터만의 고유한 문제라고 할 수 있다. SD 클러스터에서 기존에 제안된 대부분의 캐쉬 일관성 기법들은 노드의 버퍼가 전통적인 LRU 알고리즘에 따라 동작한다고 가정하고 있으며, 버퍼 교체 알고리즘과 캐쉬 일관성 기법간의 성능 관계에 대해서는 연구된 바 없다.

이런 관점에서 본 논문에서는 SD 클러스터의 특징을 고려한 버퍼 교체 알고리즘들을 제안하고, 상이한 버퍼 교체 알고리즘에 대한 캐쉬 일관성 기법의 성능을 평가한다. 구체적으로 본 논문에서는 다음과 같은 내용들을 기술한다.

- SD 클러스터에서 각 노드의 데이타 액세스 패턴을 고려하여 버퍼 히트율을 향상시킬 수 있는 버퍼 교체 알고리즘들을 제안한다. 제안한 알고리즘들은 갱신 페이지에 우선순위를 둠으로써 디스크 기록 연산의 빈도수를 줄이거나, 혹은 각 노드가 자주 액세스하는 지역 페이지에 우선순위를 둠으로써 노드간 메시지 전송 빈도수를 줄인다.
- SD 클러스터에서 여러 노드의 메모리 버퍼에 동일한 페이지가 캐싱될 가능성을 줄여 버퍼 용량을 보

다 효과적으로 사용하는 단일 버전 유지 알고리즘을 제안하고, 이를 이용한 두 가지 버퍼 교체 알고리즘을 제안한다.

- SD 클러스터를 위한 실험 모형을 개발하였고, SD 클러스터의 구성과 데이타베이스 부하 환경을 다양하게 변화시키면서 제안한 버퍼 교체 알고리즘들의 성능을 비교한다.

본 논문의 나머지 구성은 다음과 같다. 2절에서는 본 논문에서 가정하고 있는 캐쉬 일관성 기법들에 대해 설명하고, 3절에서는 SD 클러스터의 특징을 고려한 버퍼 교체 알고리즘들을 제안한다. 4절에서는 성능평가를 위한 모의실험 모형을 설명하고, 5절에서 실험결과들을 분석한다. 끝으로 6절에서 결론을 맺는다.

2. 캐쉬 일관성 기법

본 절에서는 SD 클러스터의 가장 대표적인 캐쉬 일관성 기법인 ARIES/SD[4]와 ARIES/SD의 성능을 개선한 RO-ARIES(Read Optimized ARIES/SD)[7,14]를 소개한다. 캐쉬 일관성 기법은 3절에서 제안하는 버퍼 교체 알고리즘들과 연동하여 동작한다.

2.1 ARIES/SD

ARIES/SD는 캐쉬 페이지에 대한 물리적 일관성을 유지하기 위하여 페이지의 소유자 노드를 나타내는 물리적 로크 개념을 이용한다[4]. 노드 N이 페이지 P를 갱신하기 위해서는 먼저 P의 소유자가 되어야 하며 이를 위해 P에 대한 배타적(eXclusive: X) 모드의 물리적 로크를 획득하여야 한다. 기존의 소유자 노드는 P에 대한 갱신 연산을 실행한 후 소유권을 N에게 넘겨주며, 이때 P의 최신 내용도 같이 전송된다. 물리적 로크를 이용함으로써 한 순간에 하나의 노드만이 소유자 노드가 될 수 있고, 여러 개의 갱신 연산들이 P에 동시에 실행될 수 없다. ARIES/SD에서는 페이지가 갱신될 때마다 로그가 생성되며, 각 로그에 대해 일련 번호(Log Sequence Number: LSN)가 할당된다. 뿐만 아니라, 데이타베이스의 페이지마다 그 페이지를 최종적으로 갱신한 연산에 대한 로그의 LSN을 저장하는 PageLSN 필드가 유지된다.

전역 로크 관리자(GLM)는 트랜잭션이 액세스한 데이타 레코드에 대한 논리적 로크 테이블과 페이지에 대한 물리적 로크 테이블을 관리한다. 물리적 로크 테이블은 <페이지 식별자, 소유자 노드, 최신 버전 페이지의 PageLSN> 정보 등을 저장한다. 노드 N에서 실행중인 트랜잭션 T가 페이지 P에 저장된 레코드 R을 액세스할 경우, N은 GLM에게 먼저 R에 대한 T의 논리적 로크를 요청한다. 로크가 허용될 경우 GLM은 로크 응답 메시지에 R을 포함하는 P의 최신 버전에 대한 PageLSN을 함께 전송한다. N은 P를 캐싱하고 있지 않거나 혹은 자신이 캐싱하고 있는 P의 PageLSN이 전송받은 P의 PageLSN보다 작을 경우 GLM에게 P의 최신 버전을 요청한다. P의 요청 메시지에는 P의 물리적 로크 모드(갱신 연산: X 모드, 판독 연산: S 모드)도 함께 포함된다. GLM은 물리적 로크 테이블을 참조한 후 P의 소유자 노드가 존재하면 N의 요청 메시지를 전달한다. 소유자 노드는 S 모드로 P가 요청될 경우 단순히 P를 N에게 전달한다. 이와는 달리, N이 X 모드로 P를 요청할 경우 소유자 노드는 P에 대한 갱신 로그를 디스크에 저장한 후, 자신의 버퍼에서 P에 대한 갱신 비트를 해제한다. 그리고 P를 N에게 전송한 후, P의 소유자가 N으로 변경되었음을 GLM에게 통보한다. 소유권이 변경될 경우, GLM은 이를 물리적 로크 테이블에 반영한다. 소유자 노드가 존재하지 않을 경우에는 디스크에서 P의 최신 버전을 액세스하도록 GLM이 N에게 통보한다.

2.2 RO-ARIES

ARIES/SD의 경우 소유자 노드를 통해서만 노드간 페이지 전송이 이루어지며, 소유자 노드는 페이지에 대한 갱신 연산을 실행한 노드로 국한된다. 그러나 소유자 노드가 없는 경우라도 최신 버전의 페이지를 캐싱하고 있는 노드가 존재할 수 있다. 예를 들면 페이지 P를 캐싱하고 있는 노드가 없는 상태에서 P의 최신 버전은 디스크에 저장되어 있다. 이때 노드 N이 P에 대한 판독 연산을 위해 디스크에서 P를 액세스한 후 이를 캐싱한 경우, N은 P의 소유자 노드는 아니지만 최신 버전을 캐싱하고 있다. 다른 노드가 P를 요청한 경우, N에 캐싱된 P를 전송함으로써 불필요한 디스크 연산을 줄일 수 있다는 것이 RO-ARIES의 기본 개념이다.

RO-ARIES의 경우 GLM은 물리적 로크 테이블에 페이지의 최신 버전을 캐싱하고 있는 노드들의 집합인 복사 리스트(copy list)를 추가로 유지한다. 이를 위하여 소유자가 아닌 노드가 페이지의 최신 버전을 캐싱한 경우 GLM에게 이를 알리는 메시지가 추가로 전송되어야 한다. 페이지에 대한 전송이 필요한 경우, 소유자가 존재하면 그 노드에게 페이지 전송을 요청한다. 소유자 노드가 존재하지 않을 경우 복사 리스트의 임의 노드에게 전송 메시지를 전달한다. 복사 리스크가 공백인 경우에는 디스크에서 페이지를 액세스한다.

3. 버퍼 교체 알고리즘

본 절에서는 먼저 SD 클러스터에서 기존에 제안된 대부분의 캐쉬 일관성 기법들이 가정하고 있는 버퍼 교체 알고리즘인 LRU의 문제점에 대해 설명한다. 그리고 LRU의 문제점을 해결할 수 있는 버퍼 교체 알고리즘들

을 제안한다.

3.1 LRU의 문제점

버퍼 교체 알고리즘이란 메모리 버퍼가 가득 찬 경우 추가적인 페이지를 저장할 공간을 마련하기 위하여 버퍼에서 제거할 페이지를 선택하는 정책이다. LRU는 대표적인 버퍼 교체 알고리즘으로 가장 최근까지 액세스하지 않았던 페이지를 희생자로 선택하여 버퍼에서 제거한다. LRU는 구현의 용이성 및 성능의 상대적 우수성으로 인해 대부분의 상용 시스템에서 사용되지만, 다양한 액세스 패턴이 존재하는 데이타베이스 분야에서는 비효율적인 경우가 존재하는 것으로 알려져 있다[8]. 뿐만 아니라, SD 클러스터에서 각 노드마다 자신의 메모리 버퍼를 관리하기 위하여 LRU 알고리즘을 독자적으로 사용할 경우 다음과 같은 추가적인 문제가 발생한다.

- 친화도 기반 트랜잭션 라우팅을 지원하는 SD 클러스터의 경우, 각 노드마다 빈번히 액세스되는 페이지 집합이 존재하고 이를 그 노드의 "지역 페이지"(local page)라 정의한다. 지역 페이지가 아닌 페이지들은 "원격 페이지"(remote page)라 정의한다. 지역 페이지는 앞으로 액세스될 가능성이 상대적으로 높지만, LRU 알고리즘에 의해 방금 액세스한 원격 페이지보다 교체될 가능성이 높다.
- 동일한 페이지를 여러 노드에서 동시에 캐싱할 수 있으며, 그 결과 전체 시스템 관점에서 메모리 버퍼를 비효율적으로 사용할 수 있다.

SD 클러스터에서 각 노드는 자신의 버퍼(지역 버퍼)나 다른 노드의 버퍼(원격 버퍼), 그리고 디스크에 저장된 데이타 페이지들을 액세스한다. 이 경우 지역 버퍼에 저장된 페이지를 액세스하는 속도가 가장 빠르며, 다음으로 원격 버퍼에 저장된 페이지, 그리고 디스크에 저장된 페이지를 액세스하는 속도가 가장 늦다. 따라서 버퍼 교체 알고리즘은 각 노드가 원하는 페이지를 지역 버퍼나 원격 버퍼에서 찾을 수 있는 확률을 높여야 한다. 지역 페이지에 대한 고려가 없으며 메모리 버퍼를 비효율적으로 사용한다는 관점에서 LRU는 SD 클러스터에 부적합하다고 할 수 있다.

LRU의 문제점을 해결하기 위한 많은 연구들이 진행되어 왔다. 기존에 제안된 대부분의 알고리즘들은 최근에 액세스된 페이지와 자주 액세스되는 페이지를 구분함으로써 순차 스캔이나 반복 액세스, 혹은 인덱스와 데이타 페이지가 함께 캐싱될 경우 발생하는 LRU의 문제점을 해결하려고 노력하였다[8]. 예를 들면 LRU-K[15]나 2Q[16] 알고리즘의 경우, 버퍼 페이지에 대해 보다 많은 액세스 이력 정보를 유지하고 이를 이용하여 버퍼에서 교체할 페이지를 선택한다. LIRS[17] 알고리즘은 버퍼 페이지의 연속된 액세스 시점 사이에 액세스된 버퍼 페이지의 횟수를 유지하고, 그 횟수가 적은 페이지를 교체한다. 그러나 이러한 알고리즘들은 SD 클러스터에서 발생하는 동일한 페이지의 중복 캐싱이나 상이한 노드별 데이타 액세스 패턴을 고려하지 않고 있다. 뿐만 아니라 LRU의 문제점을 해결하기 위하여 보다 많은 액세스 정보들을 유지하여야 하므로 구현의 복잡성이 증가하고, 그 결과 실제 시스템에 적용되는 사례도 드물다. 이런 관점에서 본 논문에서는 가장 널리 사용되고 있는 LRU를 SD 클러스터의 특성에 맞게 확장하는 방안들을 제안하도록 한다.

3.2 갱신 페이지 고려(Dirty Page Last: DPL) 알고리즘

DPL 알고리즘은 갱신 페이지에 우선권을 둔다. 구체적으로, 버퍼에 존재하는 페이지들을 갱신 비트(dirty bit)가 설정된 갱신 그룹과 갱신 비트가 설정되지 않은 판독 그룹으로 분할하고, 각 그룹에 대해 별도로 LRU 리스트를 유지한다. 판독 그룹에서 LRU 알고리즘에 따라 희생자 페이지를 우선적으로 선택하고, 판독 그룹이 공집합일 경우에만 갱신 그룹에서 LRU 순서로 희생자 페이지를 선택한다. 갱신 페이지의 교체 확률을 낮춤으로써 다음과 같은 장점을 기대할 수 있다.

- ARIES/SD나 RO-ARIES의 경우 각 페이지에 대해 소유자 노드만이 갱신 페이지를 캐싱한다. 갱신 페이지의 교체 확률을 낮춤으로써 소유자 노드의 버퍼에서 갱신 페이지가 캐싱되는 기간이 길어지고, 그 결과 소유자 노드를 통한 페이지 전송 빈도수가 증가하여 디스크 연산을 줄일 수 있다.
- 각 페이지에 대해 기껏해야 하나의 소유자 노드만 존재하므로 서로 다른 노드가 동일한 페이지를 캐싱할 확률이 낮아진다. 따라서 보다 많은 페이지들을 캐싱할 수 있다.
- 갱신 페이지가 희생자로 선택될 경우 갱신된 내용을 디스크에 먼저 기록하고 교체하여야 한다. 갱신 페이지의 교체 확률을 낮춤으로써 디스크 기록 연산을 줄일 수 있다.

3.3 지역 페이지 고려(Local Page Last: LPL) 알고리즘

LPL 알고리즘은 버퍼에 캐싱된 페이지들을 지역 페이지와 원격 페이지로 분류하고, 지역 페이지에 우선순위를 두는 정책이다. 구체적으로 지역 페이지와 원격 페이지에 대해 별도로 LRU 리스트를 유지하며, 원격 페이지들을 우선적으로 교체한다. 그리고 원격 페이지가 없을 경우에만 지역 페이지들을 교체한다. 이 알고리즘의 장점은 다음과 같다.

- 지역 페이지에 우선순위를 둠으로써 원하는 페이지를 지역 버퍼에서 찾을 가능성이 높아진다. 그 결과 원격 페이지를 액세스하기 위한 네트워크 트래픽을 줄일 수 있다.
- 각 노드의 지역 페이지가 균등히 분할될 경우 동일한 페이지가 여러 노드에 캐싱될 확률이 낮아진다.

3.4 단일 버전 유지(1-COPY) 알고리즘

1-COPY 알고리즘은 여러 노드의 메모리 버퍼에 동일한 페이지가 캐싱될 가능성을 줄여 버퍼 용량을 보다 효과적으로 사용하는 것을 목적으로 한다. 그림 2는 두 개의 노드가 동일한 페이지를 캐싱하는 예를 나타낸다.

ARIES/SD의 경우 노드 N1은 페이지 P의 소유자 노드이며, RO-ARIES의 경우에는 N1이 P의 소유자 노드이거나 혹은 복사 리스트에 속한다. N2가 P를 전송받은 후 P는 N1과 N2의 버퍼에 모두 캐싱되게 되고, 그 결과 전체 시스템 관점에서 버퍼에 캐싱된 페이지 수가 감소하는 문제점이 발생한다. 1-COPY 알고리즘은 P가 전송된 후, 노드 N1이나 N2에서 P가 즉시 교체될 수 있도록 함으로써 중복 캐싱 문제를 해결한다. 구체적으로 각 노드의 버퍼는 LRU 리스트로 구성되며, P가 전송된 후 아래 두 가지 유형 중에 하나를 선택한다.

- **유형 1:** N1이 N2에게 P를 전송한 후, N1의 버퍼에서 P를 LRU 리스트의 맨 끝으로 이동하여 즉시 교체되도록 한다. N2의 버퍼에서 P는 LRU 리스트의 맨 앞에 저장된다.
- **유형 2:** N2는 P를 전송받아 연산을 실행한 후 자신의 버퍼에서 P를 LRU 리스트의 맨 끝으로 이동하여 즉시 교체되도록 한다. N1은 N2에게 P를 전송한 후, 자신의 버퍼에서 P를 LRU 리스트의 맨 앞으로 이동한다.

페이지 전송이 일어날 때마다 유형 1과 유형 2 중에 한 가지 정책을 수행함으로써 동일한 페이지가 여러 노드에 동시에 캐싱되는 기간을 최소화 한다. 두 가지 유형에 대한 선택은 P가 지역 페이지인지에 대한 여부와 P에 대한 연산의 종류에 따라 결정되는데, 표 1에 선택

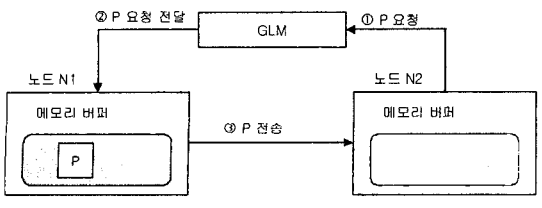

그림 2 두 개의 노드가 동일한 페이지를 캐싱하는 경우

표 1 1-COPY 알고리즘의 유형 선택 기준

페이지 보유 노드(N1) / 페이지 요청 노드(N2)		지역 페이지		원격 페이지	
		X	S	X	S
지역 페이지	X	유형 1	유형 1	유형 1	유형 1
	S	유형 2	유형 2	유형 2	유형 1
원격 페이지	X	유형 1	유형 1	유형 1	유형 1
	S	유형 2	유형 2	유형 2	유형 2

기준을 나타낸다.

N2가 갱신을 위해 X 모드로 페이지 P를 요구할 경우 N1에서 캐싱된 P는 즉시 교체될 수 있도록 유형 1을 선택한다. 이때 N1에서 X 모드로 P를 캐싱하고 있을 경우, ARIES/SD의 소유권 이전 방식에 따라 N2에게 P를 전송하기 전에 P에 대한 갱신 비트를 해제하므로 N1에서 P가 교체되더라도 디스크 기록 연산은 발생하지 않는다.

N2가 판독을 위해 S 모드로 P를 요구할 경우 N1이 P에 대해 X 모드의 로크를 보유하고 있는 소유자 노드라면 유형 2의 정책에 따라 N2에서 P가 즉시 교체될 수 있도록 한다. 그 이유는 3.2절의 DPL 알고리즘에서 설명했듯이 소유자 노드를 통한 페이지 전송 빈도수를 증가시키고 갱신 페이지가 교체될 경우 발생할 수 있는 디스크 기록 연산을 줄이기 위해서이다.

RO-ARIES에서는 N1과 N2가 모두 S 모드로 P를 액세스할 때 P의 전송이 발생할 수 있다. 이 경우 N1과 N2 중에서 P를 지역 페이지로 하는 노드가 있으면 그 노드가 P를 캐싱하도록 한다. 즉, N1이 P를 지역 페이지로 할 경우에는 유형 2를 선택하며, N2의 지역 페이지일 경우에는 유형 1을 선택한다. 만약 두 노드가 모두 P를 지역 페이지로 하거나[1] 두 노드에서 모두 지역 페이지가 아닌 경우에는 기존에 P를 캐싱하던 노드 N1이 P를 계속 캐싱하도록 유형 2를 선택한다. 그 이유는 N1이나 N2 모두 P를 액세스할 확률이 동일하며, GLM의 물리적 로크 테이블에 N1이 P의 복사 리스트로 이미 등록되어 있기 때문이다.

1-COPY 알고리즘에서 버퍼 교체를 위하여 LRU나 DPL, 혹은 LPL 알고리즘을 사용하는 것이 모두 가능하다. 본 논문에서는 1-COPY 알고리즘에 LRU를 적용한 1-LRU 알고리즘과 DPL을 적용한 1-DPL 알고리즘을 각각 구현하여 성능을 평가한다. 1-COPY 알고리즘은 LPL과 유사하게 지역 페이지를 우선적으로 고려하므로 LPL 알고리즘을 채택할 경우 1-LRU에 비해 성

1) 특정 클래스에 속한 트랜잭션들이 많이 입력될 경우, 트랜잭션 라우터는 부하 분산을 위하여 여러 노드에 하나의 AC를 할당할 수 있고 그 결과 노드간 지역 페이지가 공유된다[7].

능상 차이가 크지 않을 것으로 예상되어 실험에서 제외한다.

4. 실험 모형

본 절에서는 버퍼 교체 알고리즘들의 성능을 평가하기 위한 모의실험 모형을 설명한다. SD 클러스터의 성능 평가 모형은 그림 3에 나타나며, 이산-사건 실험 프로그램인 CSIM 언어[18]를 이용하여 구현하였다.

SD 클러스터는 하나의 GLM과 네트워크를 통해 연결된 여러 개의 노드로 구성되고, 디스크는 모든 노드에 의해 공유된다. 트랜잭션 생성기는 데이타베이스 부하 환경에 따라 다양한 액세스 패턴을 갖는 사용자 트랜잭션을 생성하여 각 노드에게 할당한다. 노드는 트랜잭션 관리자와 버퍼 관리자, 그리고 자원 관리자 등으로 구성된다. 트랜잭션 관리자는 할당된 트랜잭션에 대해 GLM에 로크 요청 메시지와 트랜잭션 완료 메시지를 전송한다. 버퍼 관리자는 3절에서 설명한 다섯 가지의 버퍼 교체 알고리즘(LRU, DPL, LPL, 1-LRU, 1-DPL)을 각각 구현한다.

GLM은 트랜잭션 스케줄러와 캐쉬 관리자를 이용하여 노드에서 요청한 로킹과 캐쉬 일관성 기법을 수행한다. 트랜잭션 스케줄러는 레코드 단위의 2 단계 로킹 기법을 지원한다. 캐쉬 관리자는 버퍼의 전역 일관성을 유지하기 위하여 ARIES/SD와 RO-ARIES 알고리즘을 각각 구현한다. 본 논문에서는 GLM이 전체 시스템에 하나만 존재한다고 가정하며, GLM이 시스템의 병목이 되는 것을 방지하기 위하여 GLM의 CPU는 일반 노드의 CPU에 비해 성능이 우수하다고 가정한다.

실험에서 사용한 시스템 구성 변수와 트랜잭션 변수들을 표 2에서 요약한다. 각 매개 변수의 구체적인 값들은 [7]과 [14]에서 주로 참조하였다.

네트워크 관리자는 1Gbps의 대역폭을 갖는 FIFO 서버로 구현하였다. 네트워크를 통해 메시지를 전송하는 과정을 표현하기 위해 노드의 CPU 및 GLM의 CPU는

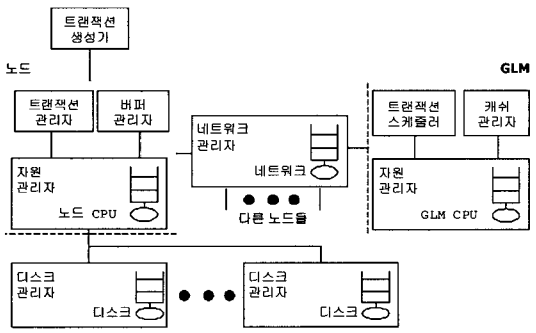

그림 3 SD 클러스터의 성능 평가 모형

표 2 입력 매개 변수

시스템 구성 변수		
LCPUSpeed	노드 CPU의 속도	500 MIPS
GCPUSpeed	GLM CPU의 속도	2 GIPS
NetBandwidth	네트워크의 데이타 전송 속도	1 Gbps
NumNode	노드의 수	4, 8, 16, 40
NumDisk	공유 디스크의 수	16
MinDiskTime	최소 디스크 액세스 시간	0.005 초
MaxDiskTime	최대 디스크 액세스 시간	0.02 초
PageSize	페이지 크기	4096 bytes
RecPerPage	페이지당 레코드 수	200
DBSize	데이타베이스의 페이지 수	8192
HotSet	Hot set의 비율	0.2
BufSize	노드의 버퍼 크기(페이지 수)	256, 512, 1024
MsgInst	메시지당 명령 수	22000
LockInst	로크 획득/해제를 위한 명령 수	3000
PerIOInst	디스크 I/O를 위한 명령 수	25000
PerObjInst	레코드 처리를 위한 명령 수	25000
트랜잭션 변수		
TrxSize	트랜잭션이 액세스하는 레코드 수	10
SizeDev	트랜잭션 크기 편차	0.1
UpdateProb	레코드 갱신 비율	0.0 - 0.5
NumTrans	사용자 트랜잭션의 수	400
ACNum	AC의 수	8
ACLocality	지역 클러스터를 액세스할 확률	0.3, 0.8
HotProb	Hot set을 액세스할 확률	0.2, 0.8

메시지마다 고정된 명령 수(MsgInst)를 실행한다. 디스크 수는 16개이며 각 디스크는 FIFO 큐를 이용하여 I/O 요청을 들어온 순서대로 처리한다. 디스크 액세스 시간은 0.005초와 0.02초 사이의 일양 분포를 따른다고 가정한다.

버퍼 교체 알고리즘들의 차이점이 명확하게 드러나도록 하기 위하여 8192개의 페이지로 구성된 소규모 데이타베이스를 가정하였다. 노드의 버퍼 크기(BufSize)가 512이고 노드의 수(NumNode)가 8일 경우 전체 시스템의 버퍼 용량의 합은 데이타베이스 크기의 1/2에 해당한다. 이를 기본적인 실험 환경으로 설정하였으며, BufSize를 1/2과 2배로 변경하고 NumNode를 4와 16, 그리고 40으로 각각 변경하면서 다양한 시스템 구성에 따른 버퍼 교체 알고리즘들의 성능 변화를 관찰하였다.

SD 클러스터에서는 두 가지 형태의 참조 지역성이 존재한다. 먼저 데이타베이스 액세스 패턴에 따라 트랜잭션들을 여러 개의 친화도 클러스터(AC)로 분류할 수 있고, 동일한 AC에 속하는 트랜잭션들은 데이타베이스의 특정 영역을 빈번하게 액세스하므로 이들을 하나의 노드에 할당함으로써 참조 지역성을 증가시킬 수 있다[6,7]. 본 논문에서는 AC의 수(ACNum)를 8로 고정하였다. 사용자 트랜잭션의 수(NumTrans)가 400이므로 50개의 트랜잭션이 하나의 AC를 구성한다. AC마다 전체 데이타베이스 크기(DBSize)의 1/8(=1024)에 해당하

는 영역을 할당하고, 서로 다른 AC에 할당된 데이타베이스 영역들은 겹치지 않는다고 가정한다. 각 노드마다 해당 AC의 영역에 속하는 페이지들을 지역 페이지라고 정의하며, 그렇지 않은 페이지들을 원격 페이지라고 정의한다. 각 트랜잭션이 지역 페이지를 액세스할 확률은 ACLocality 변수에 의해 표현된다.

두 번째 참조 지역성으로 각 AC에 할당된 데이타베이스 영역의 페이지들 중에서 특정 페이지들이 상대적으로 빈번하게 액세스될 수 있다. 이러한 페이지들의 집합을 "hot set"이라 정의한다. 각 AC에 할당된 데이타베이스 영역 중에서 20%의 페이지를 hot set으로 정의하고(HotSet = 0.2), 이에 대한 액세스 확률을 HotProb로 표현한다. HotProb의 값이 클수록 데이타베이스 영역 내에서 hot set을 액세스할 확률이 높아진다.

각 트랜잭션에 의해 액세스되는 레코드의 수는 TrxSize ± TrxSize × SizeDev 사이의 일양 분포를 따른다. UpdateProb는 레코드의 갱신 비율을 의미한다. 과도한 로크 충돌을 피하기 위하여 각 페이지마다 200개의 레코드를 저장한다고 가정하였으며 레코드 단위의 로킹을 구현하였다. 각 레코드를 처리하는 데는 25000개의 명령어(PerObjInst)가 실행된다고 가정하였다.

모의실험에서 사용한 주요 성능 평가 지수는 트랜잭션 처리율이다. 트랜잭션 처리율은 단위 시간동안 완료한 트랜잭션의 수에 의해 측정된다. 그리고 보조 성능 지수로 버퍼 히트율과 갱신 페이지 선택율을 사용한다. 버퍼 히트율은 지역 버퍼 히트율과 원격 버퍼 히트율의 합으로 정의되는데, 지역 버퍼 히트율은 필요한 페이지를 트랜잭션이 실행되는 노드의 버퍼에서 발견할 확률이다. 원격 버퍼 히트율은 트랜잭션의 실행에 필요한 페이지를 다른 노드의 버퍼에서 발견할 확률이다. 갱신 페이지 선택율이란 버퍼 교체 알고리즘의 결과로 갱신 페이지가 선택될 확률이다.

5. 실험 결과

본 절에서는 SD 클러스터에서 버퍼 교체 알고리즘들의 성능 평가 결과를 분석한다. 비교한 알고리즘들은 LRU, DPL, LPL, 1-LRU, 1-DPL이며, ARIES/SD와 RO-ARIES 캐쉬 일관성 알고리즘을 대상으로 한다. 참조 지역성의 정도와 버퍼 크기, 그리고 노드 수 등을 변화시키면서 차례대로 실험을 수행한다.

5.1 참조 지역성이 높은 환경

참조 지역성이 높은 환경을 생성하기 위하여 ACLocality = 0.8, HotProb = 0.8로 각각 설정한 후 실험을 수행하였다. ACLocality가 높게 설정되어 노드마다 지역 페이지에 대한 액세스 확률이 높고 노드간 페이지 무효화 현상을 줄일 수 있으므로 SD 클러스터의 성능이 극대화될 수 있는 경우이다. NumNode = 8, BufSize = 512로 두었을 때, 레코드 갱신 비율의 변화에 따른 ARIES/SD에 대한 실험 결과가 그림 4에서 7까지 나타난다.

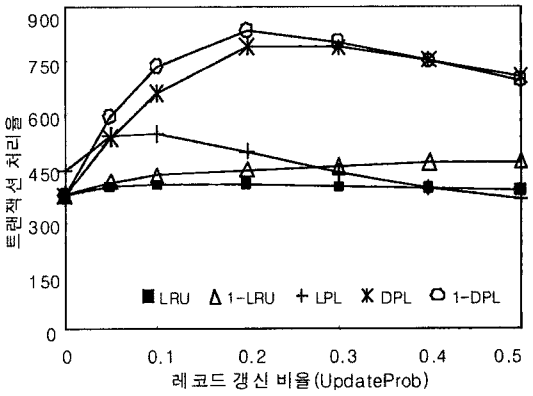

그림 4 ARIES/SD: 트랜잭션 처리율
(ACLocality = 0.8, HotProb = 0.8)

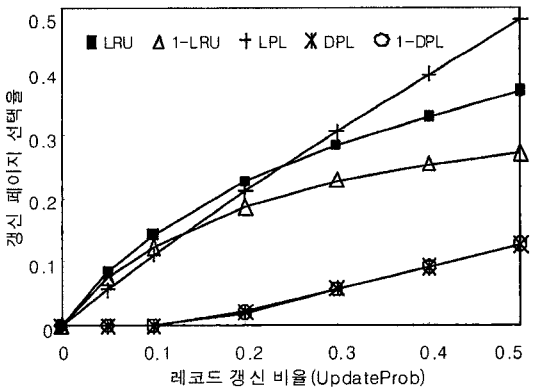

그림 5 ARIES/SD: 갱신 페이지 선택율
(ACLocality = 0.8, HotProb = 0.8)

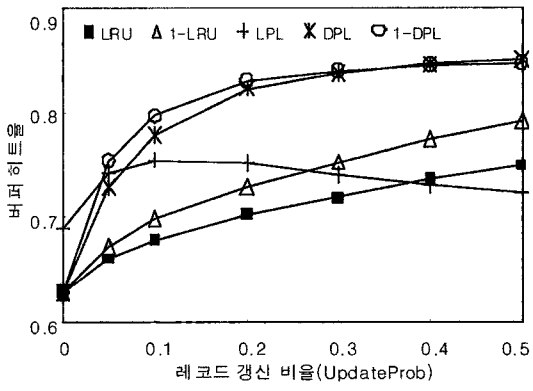

그림 6 ARIES/SD: 버퍼 히트율
(ACLocality = 0.8, HotProb = 0.8)

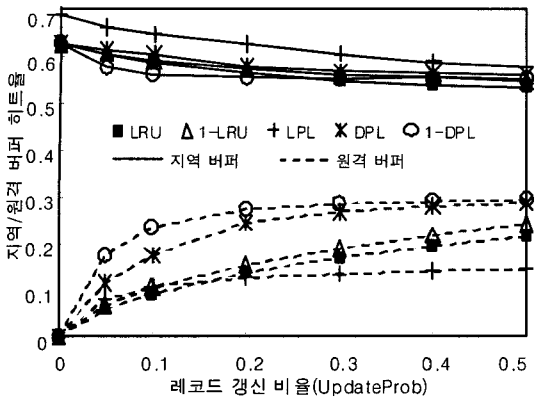

그림 7 ARIES/SD: 지역/원격 버퍼 히트율
(ACLocality = 0.8, HotProb = 0.8)

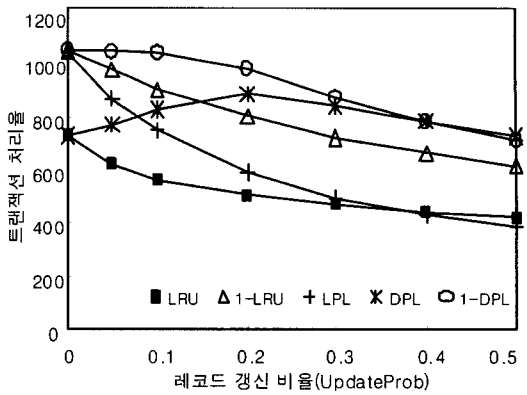

그림 8 RO-ARIES: 트랜잭션 처리율
(ACLocality = 0.8, HotProb = 0.8)

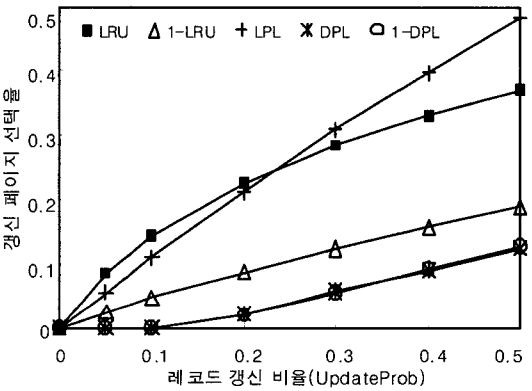

그림 9 RO-ARIES: 갱신 페이지 선택율
(ACLocality = 0.8, HotProb = 0.8)

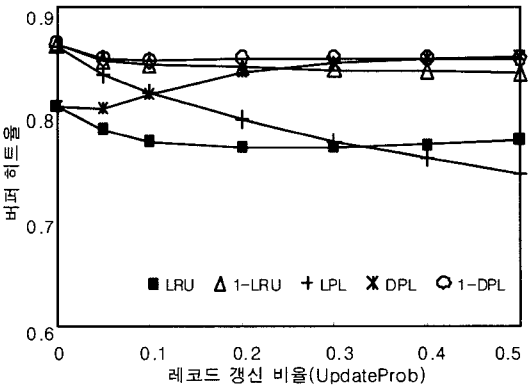

그림 10 RO-ARIES: 버퍼 히트율
(ACLocality = 0.8, HotProb = 0.8)

DPL과 1-DPL의 성능이 가장 우수한 것으로 나타났다. 그 이유는 두 가지로 설명할 수 있는데, 먼저 그림 5에서 나타나듯이 두 기법의 갱신 페이지 선택율이 가장 낮았다. 버퍼 교체 알고리즘에서 갱신 페이지 P가 선택될 경우, P가 점유하고 있는 버퍼 공간을 사용하기 위해서는 먼저 P를 디스크에 기록하여야 한다. 따라서 갱신 페이지 선택율이 높을수록 디스크 기록 연산이 많아지며, 버퍼 공간을 사용하기 위한 대기 시간도 증가한다. DPL과 1-DPL의 성능이 우수한 두 번째 이유는 그림 6에 나타나듯이 두 기법의 높은 버퍼 히트율 때문이다. ARIES/SD의 경우, 갱신 페이지를 캐싱하고 있는 소유자 노드를 통한 페이지 전송만 지원한다. 따라서 갱신 페이지가 버퍼에 존재할 가능성이 높은 DPL과 1-DPL 기법에서 페이지 전송이 발생할 가능성이 높고, 그 결과 그림 7에 나타나듯이 원격 버퍼 히트율이 다른 기법들에 비해 월등히 높은 것으로 나타났다. LPL은 지역 페이지들의 교체 확률을 낮춤으로써 지역 버퍼 히트율이 다른 기법들보다 높은 것으로 나타났다. 그러나 갱신 페이지 선택율이 가장 높아 버퍼 히트율로 얻은 이득을 상쇄하였으며, 원격 버퍼 히트율도 가장 낮아 갱신 비율이 높을 때 LRU보다도 성능이 떨어졌다.

동일한 환경(ACLocality = 0.8, HotProb = 0.8, NumNode = 8, BufSize = 512)에서 갱신 연산의 비율 변화에 따른 RO-ARIES의 실험 결과가 그림 8에서 11까지 나타난다.

ARIES/SD에 비해 RO-ARIES의 성능이 전반적으로 개선되었는데, 소유자 노드가 아닌 노드로부터 갱신되지 않은 페이지의 전송을 허용함으로써 그림 10에 나타나는 것과 같이 버퍼 히트율이 증가하기 때문이다. 이러한 현상은 LPL과 1-LRU, 그리고 1-DPL 기법에서 레코드 갱신 비율이 낮을 때 현저하게 나타났으며, ARIES/SD와 비교할 때 약 3배 정도 성능이 향상 되었다. 그 이유는 LPL이나 1-COPY 기반의 버퍼 교체 알고리즘들은 각 노드마다 자신의 지역 페이지를 우선적으로 캐싱하

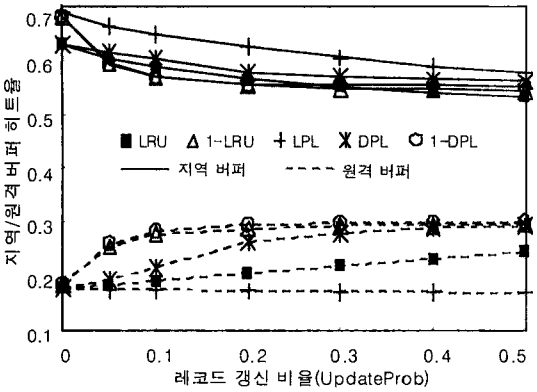

그림 11 RO-ARIES: 지역/원격 버퍼 히트율
(ACLocality = 0.8, HotProb = 0.8)

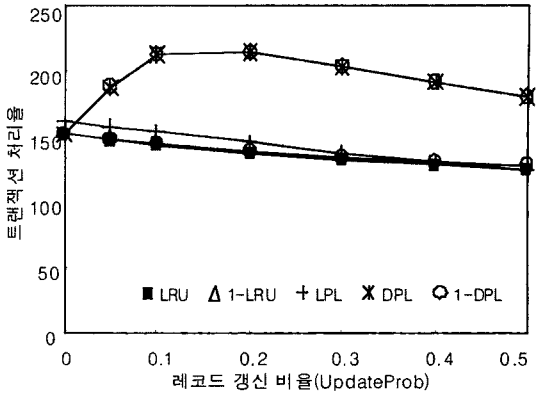

그림 12 ARIES/SD: 트랜잭션 처리율
(ACLocality = 0.3, HotProb = 0.2)

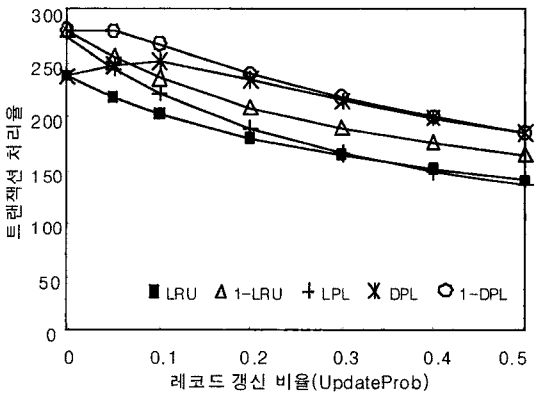

그림 13 RO-ARIES: 트랜잭션 처리율
(ACLocality = 0.3, HotProb = 0.2)

므로 노드간에 중복된 페이지가 캐싱될 확률이 낮고, 그 결과 그림 11에 나타나듯이 지역/원격 버퍼 히트율을 높일 수 있기 때문이다. 이와는 달리 LRU는 동일한 페이지가 여러 노드에 오랫동안 캐싱되어 있을 수 있으므로, 버퍼 공간을 비효율적으로 사용한다. DPL의 경우에도 레코드 갱신 비율이 0일 경우에는 LRU와 동일하게 동작하므로 성능이 낮게 나타났다. 레크드 갱신 비율이 증가할수록 LPL과 1-LRU의 성능이 급격하게 떨어졌는데, 이는 갱신 페이지 선택율의 증가 및 이에 따른 버퍼 히트율의 저하 때문이다. 1-DPL은 레코드 갱신 비율이 낮을 때에는 지역 페이지 우선 정책으로, 레코드 갱신 비율이 높을 때에는 갱신 페이지 우선 정책의 장점으로 인하여 성능이 가장 우수하였고, LRU와 비교할 때 최대 2배 정도 성능이 향상되었다.

5.2 참조 지역성이 낮은 환경

본 절에서는 ACLocality = 0.3, HotProb = 0.2로 변경하여 실험을 수행하였다. ACLocality가 낮아짐으로 인해 트랜잭션들이 지역 페이지를 액세스할 확률이 낮으며, 낮은 HotProb로 인해 데이타베이스 영역내의 모든 페이지를 동일한 확률로 액세스하므로 참조 지역성이 매우 떨어진다. 5.1절과 동일하게 NumNode = 8, BufSize = 512로 설정하였다. 갱신 연산의 비율 변화에 따른 ARIES/SD와 RO-ARIES의 실험 결과가 그림 12와 13에 나타난다.

버퍼 교체 알고리즘들의 성능 변화 패턴은 5.1절과 유사하게 나타났지만 양적으로는 최대 70% 정도 성능이 떨어지는 것으로 나타났다. 트랜잭션들이 전체 데이타베이스의 페이지들을 무작위로 빈번히 액세스함으로 인해 지역 노드나 원격 노드의 버퍼에 캐싱된 페이지를 다시 액세스할 확률이 낮아졌다. 따라서 모든 버퍼 교체 알고리즘들에 대해 버퍼 히트율이 감소하였고 버퍼 교체 알고리즘들 간의 차이점도 명확해지지 않았다. 이러한 결과는 NumNode 및 BufSize, 그리고 DBSize 값의 설정과도 관계가 있다. 구체적으로 DBSize가 8192개의 페이지로 구성되지만, 전체 시스템은 최대 절반의 페이지(NumNode * BufSize = 4096)만 캐싱할 수 있다. 모든 페이지를 캐싱할 수 없으므로 참조 지역성이 낮을 경우 빈번한 디스크 액세스로 인해 성능이 떨어질 수밖에 없다.

대부분의 경우에서 RO-ARIES의 성능이 ARIES/SD의 성능보다 우수하였고, RO-ARIES에서 버퍼 교체 기법들의 성능 특성이 다양하게 나타나므로 본 논문의 나머지 실험에서는 RO-ARIES를 대상으로 버퍼 교체 기법들의 성능을 분석하도록 한다.

5.3 버퍼 크기의 변화

노드의 버퍼 크기(BufSize)를 256, 512, 1024로 변화하면서 버퍼 교체 알고리즘들의 성능을 비교하였다. DBSize가 8192이므로 NumNode를 8로 고정했을 때,

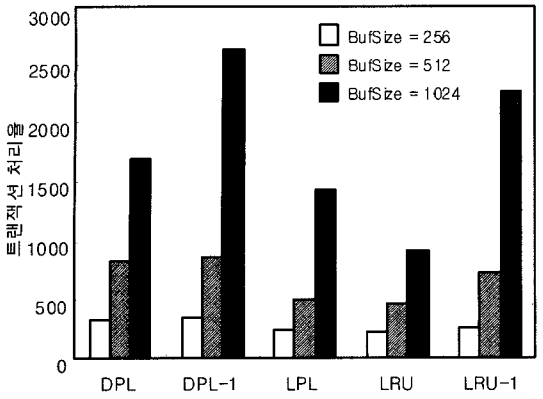

그림 14 버퍼 크기 변화에 따른 트랜잭션 처리율
(ACLocality = 0.8, HotProb = 0.8)

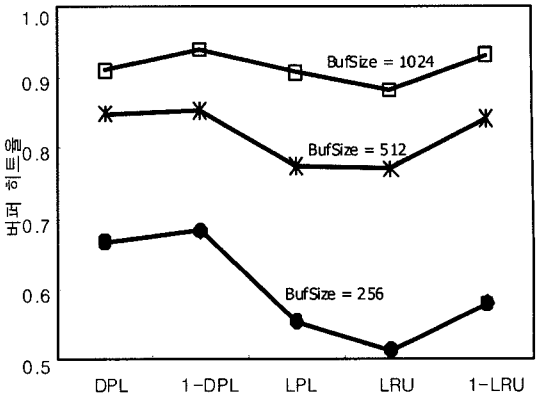

그림 16 버퍼 크기 변화에 따른 버퍼 히트율
(ACLocality = 0.8, HotProb = 0.8)

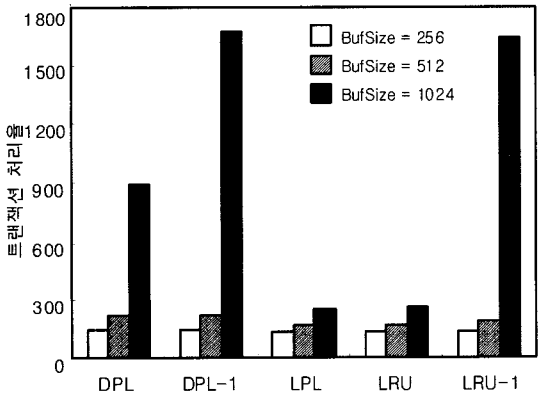

그림 15 버퍼 크기 변화에 따른 트랜잭션 처리율
(ACLocality = 0.3, HotProb = 0.2)

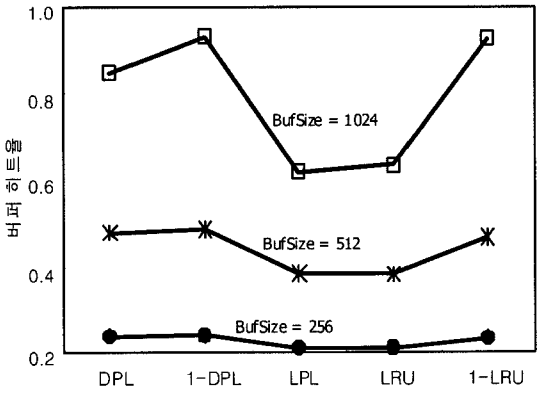

그림 17 버퍼 크기 변화에 따른 버퍼 히트율
(ACLocality = 0.3, HotProb = 0.2)

BufSize의 각 설정은 데이타베이스 페이지 중에서 최대 25%, 50%, 100%를 캐싱할 수 있는 용량을 나타낸다. 참조 지역성이 높은 환경(ACLocality = 0.8, HotProb = 0.8)과 낮은 환경(ACLocality = 0.3, HotProb = 0.2)으로 나누어 실험을 수행하였으며, NumNode = 8, UpdateProb = 0.3으로 고정하였다. 실험 결과는 그림 14에서 17까지 나타난다.

BufSize가 증가할수록 버퍼 히트율이 증가하며 모든 알고리즘들의 성능이 향상되었다. 그림 14의 참조 지역성이 높은 환경에서 BufSize가 1024일 때 1-DPL의 성능이 가장 우수하였다. 1-DPL은 중복된 페이지의 버전들이 여러 노드에 동시에 캐싱되는 것을 최소화하므로 거의 모든 데이타베이스 페이지들이 버퍼에 캐싱되게 되고, 그 결과 그림 16에 나타나듯이 버퍼 히트율이 95%까지 올라갔다. 1-LRU도 비슷한 버퍼 히트율을 나타내지만, 갱신 페이지가 버퍼 교체 대상으로 선택될 가능성이 높기 때문에 1-DPL보다 성능이 떨어졌다. LPL이나 DPL, 그리고 LRU는 동일한 페이지를 여러 노드에서 동시에 캐싱할 확률이 높아 버퍼 히트율이 상대적으로 낮았으며, BufSize의 증가에 따른 성능 향상의 정도가 제한적이었다.

그림 15의 참조 지역성이 낮은 환경에서 BufSize가 1024일 때 1-LRU와 1-DPL의 성능 향상 정도가 가장 명확하였으며, LRU나 LPL에 비해 5배 정도 성능이 우수하였다. 참조 지역성이 낮으므로 특정 페이지가 빈번하게 액세스될 확률이 낮지만, BufSize가 1024일 경우 1-LRU와 1-DPL은 모든 페이지들을 버퍼에 캐싱할 수 있다. 뿐만 아니라, 두 알고리즘에서 갱신 페이지는 다른 노드로 전송될 때마다 액세스 시점이 현재로 변경되므로 LRU에 의해 교체될 가능성이 상대적으로 낮다. 그 결과 1-LRU의 성능은 갱신 페이지에 대한 교체 확률을 낮추는 1-DPL과 비슷한 성능을 보였다. LPL과 LRU는 낮은 버퍼 히트율과 높은 갱신 페이지 선택율로 인해 성능이 가장 낮았다.

5.4 노드 수의 변화

마지막 실험으로 SD 클러스터를 구성하는 노드 수(NumNode)를 4, 8, 16, 40으로 변화하면서 버퍼 교체 알고리즘들의 성능을 비교하였다. AC의 수를 8로 고정하였으므로, NumNode가 4일 경우에는 한 노드에 두 개의 AC들을 할당하여 노드마다 100개의 트랜잭션을 실행하였다. NumNode가 16일 경우에는 한 AC에 속하는 50개의 트랜잭션들을 두 개의 노드에 나누어 실행하며, NumNode가 40일 경우에는 다섯 개의 노드에 나누어 실행한다. 참조 지역성이 높은 환경(ACLocality = 0.8, HotProb = 0.8)과 낮은 환경(ACLocality = 0.3, HotProb = 0.2)으로 나누어 실험을 수행하였으며, BufSize = 512, UpdateProb = 0.3으로 고정하였다. 실험 결과는 그림 18과 19에 나타난다.

NumNode가 증가할수록 SD 클러스터 내 메모리 버퍼의 총 용량이 증가하므로 모든 버퍼 교체 알고리즘들의 성능이 증가하였다. 실제로 NumNode가 16일 경우,

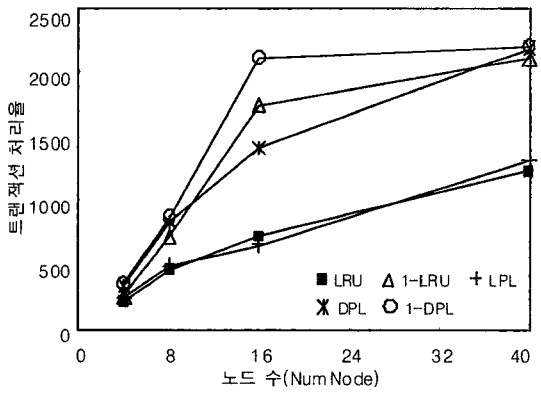

그림 18 노드 수의 변화에 따른 트랜잭션 처리율 (ACLocality = 0.8, HotProb = 0.8)

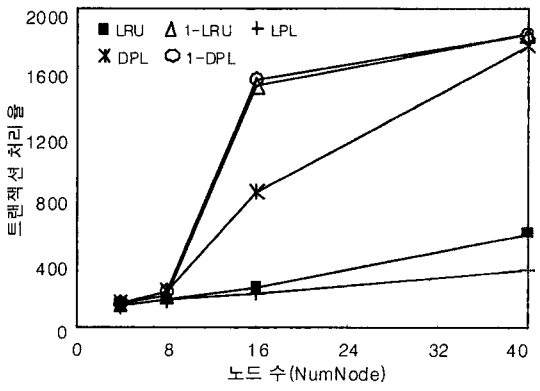

그림 19 노드 수의 변화에 따른 트랜잭션 처리율 (ACLocality = 0.3, HotProb = 0.2)

각 알고리즘들의 성능은 5.3절에서 설명한 BufSize가 1024인 경우와 기본적으로 동일하다. NumNode를 40으로 증가할 경우, 1-DPL이나 1-LRU의 성능 향상 정도가 다른 알고리즘들에 비해 다소 적게 나타났다. 그 이유는 두 가지로 설명할 수 있는데, 먼저 5.3절에서 설명했듯이 NumNode가 16일 때 두 알고리즘은 대부분의 데이타베이스 페이지들을 캐싱할 수 있으므로 NumNode를 40으로 증가하더라도 버퍼 히트율의 증가폭이 크지 않다. 이와는 달리 페이지의 중복 캐싱이 많은 DPL이나 LRU의 경우에는 NumNode를 40으로 증가할 경우 버퍼 히트율이 상대적으로 많이 증가하였다. 두 번째 이유로 NumNode가 16이상일 경우 하나의 AC에 속하는 트랜잭션들이 여러 노드에 나누어져 실행된다. 1-DPL과 1-LRU는 하나의 페이지에 대해 캐싱된 버전의 수를 하나로 제한하기 때문에 버퍼 용량의 추가적인 증가가 성능에 크기 도움이 되지 않는다. 예를 들어, AC1의 트랜잭션들이 노드 N1과 N2에 나누어져 실행되며, AC1의 hot set에 페이지 P가 포함되어 있다고 가정하자. 그리고 노드 N2가 P의 소유자 노드라고 가정하자. 노드 N1의 트랜잭션이 P를 판독할 경우 N2로부터 P를 전송받은 후 N1의 버퍼에 캐싱한다. 1-DPL이나 1-LRU의 경우 P는 N1의 지역 페이지이지만 N1은 P의 소유자 노드가 아니므로 N1의 버퍼에서 즉시 교체된다. 따라서 N1이 P를 판독할 때마다 N2로부터 P를 전송받아야 하고, 그 결과 지역 버퍼 히트율이 낮아지는 원인이 된다. DPL이나 LRU의 경우에는 전송받은 P를 N1이 계속 캐싱할 수 있으므로 지역 버퍼 히트율을 높일 수 있다. 하나의 AC를 실행하는 노드 수가 증가할수록 이러한 현상이 빈번히 발생하므로, 1-LRU와 1-DPL의 성능 향상 정도가 제한적이 된다.

6. 결 론

SD 클러스터는 독자적인 메모리 버퍼를 갖는 여러 개의 노드들로 구성되며, 트랜잭션 라우팅 정책에 따라 각 노드별로 유사한 액세스 패턴을 갖는 트랜잭션들이 할당된다. 그 결과 노드별로 상이한 참조 지역성이 존재한다. 본 논문에서는 먼저 SD 클러스터의 구성 및 액세스 패턴을 고려한 네 가지 버퍼 교체 알고리즘(LPL, DPL, 1-LRU, 1-DPL)을 제안하였다. 그리고 다양한 SD 클러스터 구성과 데이타베이스 부하 환경에서 기존의 LRU 기반 버퍼 교체 알고리즘과 성능을 비교하였다. 주요 실험 결과를 요약하면 다음과 같다.

- 갱신 페이지에 대해 높은 우선순위를 부여하는 DPL 알고리즘은 대부분의 경우에서 높은 성능을 보였다. 그러나 갱신 연산의 비율이 낮을 경우에는 DPL의

성능이 매우 낮게 나타났으며 기존 LRU 알고리즘과 큰 차이가 없었다.

- 지역 페이지에 대한 우선순위를 높게 고려하는 LPL은 갱신 페이지에 대한 디스크 기록 오버헤드로 인하여 대부분의 경우 1-DPL보다 성능이 낮게 나타났으며, 갱신 연산의 비율이 매우 높을 경우나 참조 지역성이 낮을 경우에는 LRU 알고리즘보다도 성능이 낮았다.
- 1-LRU와 1-DPL은 페이지의 중복 캐싱을 최소화하므로 버퍼 히트율을 높일 수 있었고, 갱신 연산에 대해 높은 우선순위를 부여한 1-DPL은 모든 실험에서 우수한 성능을 보였다. 특히 1-DPL은 버퍼 크기를 증가하거나 노드 수를 증가한 경우 성능 향상 정도가 매우 크므로, 하드웨어 가격 하락 및 클러스터 규모 확대라는 현 추세에 적합하다는 장점을 갖는다.

본 논문의 향후 연구 계획으로는 SD 클러스터에서 트랜잭션의 우선순위를 고려한 버퍼 교체 알고리즘을 개발하고 그 성능을 평가하는 것이다. 차별화된 트랜잭션 서비스나 실시간 서비스를 요구하는 다양한 응용들이 존재하고 이를 지원하는 SD 클러스터 기반 데이타베이스 시스템을 개발할 경우 트랜잭션의 우선순위에 대한 고려는 필수적이다. 이를 위하여 트랜잭션의 우선순위를 고려한 실시간 트랜잭션 라우팅 기술과 실시간 전역 캐쉬 일관성 기법을 현재 연구 중에 있으며, 이에 기반을 둔 버퍼 교체 알고리즘들을 개발할 예정이다.

참 고 문 헌

[1] M. Yousif, "Shared-Storage Clusters," *Cluster Computing*, Vol.2, No.4, pp.249-257, 1999.

[2] *DB2 Version 9.1 for z/OS - Data Sharing: Planning and Administration*, IBM SC18-9845-01, 2007.

[3] M. Vallath, *Oracle Real Application Clusters*, Elsevier Digital Press, 2004.

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

[5] A. Dan and P. Yu, "Performance Analysis of Buffer Coherency Policies in a Multisystem Data Sharing Environment," *IEEE Trans. Parallel and Distributed Syst.*, Vol.4, No.5, pp.289-305, 1993.

[6] P. Yu and A. Dan, "Performance Analysis of Affinity Clustering on Transaction Processing Coupling Architecture," *IEEE Trans. Knowledge and Data Eng.*, Vol.6, No.5, pp.764-786, 1994.

[7] K. Ohn and H. Cho, "Dynamic Affinity Cluster Allocation in a Shared Disks Cluster," *J. Supercomputing*, Vol.37, No.1, pp.47-69, 2006.

[8] C. Goh, Y. Shu, Z. Huang and B. Ooi, "Dynamic Buffer Management with Extensible Replacement Policies," *VLDB J.*, Vol.15, No.2, pp.99-120, 2006.

[9] A. Leff, J. Wolf, and P. Yu, "Efficient LRU-Based Buffering in a LAN Remote Caching Atchitecture," *IEEE Trans. Parallel and Distributed Syst.*, Vol.7, No.2, pp.191-206, 1996.

[10] M. Sinnwell and G. Weikum, "A Cost-Model-Based Online Method for Distributed Caching," *Proc. ICDE*, pp.532-541, 1997.

[11] M. Vilayannur, A. Sivasubramaniam, M. Kandemir, R. Thakur, and R. Ross, "Discretionary Caching for I/O on Clusters," *Cluster Computing*, Vol.9, No.1, pp.29-44, 2006.

[12] T. Wong and J. Wilkes, "My Cache or Yours? Making Storage More Exclusive," *Proc. 2002 Ann. USENIX Technical Conf.*, 2002.

[13] S. Jiang, K. Davis, and X. Zhang, "Coordinated Multilevel Buffer Cache Management with Consistent Access Locality Quantification," *IEEE Trans. Computers*, Vol.56, No.1, pp.95-108, 2007.

[14] K. Ohn and H. Cho, "Path Conscious Caching of B^+ Tree Indexes in a Shared Disks Cluster," *J. Parallel and Distributed Computing*, Vol.67, No.3, pp.286-301, 2007.

[15] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," *Proc. ACM SIGMOD*, pp.297-306, 1993.

[16] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," *Proc. VLDB*, pp.439-450, 1994.

[17] S. Jiang and X. Zhang, "LIRS: An Efficient Low Inter-reference Recency Set Replacement Policy to Improve Buffer Cache Performance," *Proc. SIGMETRICS*, pp.31-42, 2002.

[18] H. Schwetman, *CSIM User's Guide for use with CSIM Revision 18*, MCC., 1996.

조 행 래

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