공간적 근접성과 시간적 근접성을 이용한 효율적인 버퍼관리기법[1]

안재용° 민준기 정진완
한 국 과 학 기 술 원 전 자 전 산 학 과
{jyahn, jkmin, chungcw}@islab.kaist.ac.kr

An Efficient Buffer Management Strategy Using Spatial Locality and Temporal Locality

Jae-Yong Ahn° Joon-Ki Min Chin-Wan Chung
Dept. of Electrical Engineering & Computer Science, KAIST

요 약

데이터베이스 관리시스템에서 데이터를 디스크에서 읽어오는 작업은 많은 비용을 필요로 한다. 따라서 시스템의 성능을 향상시키기 위해서는 디스크 I/O 의 횟수를 최소화하는 것이 가장 중요한 문제이다. 특히 공간데이터베이스 환경에서는 클라이언트 버퍼의 크기가 데이터베이스의 크기에 비해 매우 작기 때문에 버퍼를 효율적으로 사용하여 디스크 I/O 횟수를 줄이는 문제는 더욱 중요하게 고려하여야 한다. 지금까지 디스크 I/O 횟수를 줄이기 위해서 많은 버퍼관리 기법들이 제안되었지만, 그 기법들은 시간적 근접성만을 고려해왔기 때문에 공간적 근접성도 존재하는 공간데이터베이스 환경에서는 최적의 성능을 보여주지 못했다.
본 논문에서는 공간 데이터베이스의 시간적 근접성과 공간적 근접성을 동시에 고려하는 새로운 버퍼관리기법인 *SLM-tree Buffer Management Strategy*를 제안한다. 제안한 버퍼관리기법은 공간 데이터베이스 환경에서 디스크 I/O의 횟수를 현저하게 줄임으로서 기존의 방법들에 비해 월등한 성능을 보여준다.

1. 서론

기존의 데이터베이스로는 간단한 구조의 정보만을 저장 할 수 있었기 때문에 복잡한 공간 객체를 효과적으로 처리하기 위해서 새로운 형태의 DBMS 가 필요하게 되었으며 이런 요구에 의해 등장하게 된 것이 공간 데이터베이스 관리 시스템(Spatial Database Management System: SDBMS)이다.
SDBMS은 대용량의 데이터를 관리하며 그에 따라 클라이언트 버퍼의 크기가 데이터의 크기에 비해 매우 작다. 따라서 디스크(disk)에서 데이터를 가져오는 횟수를 줄여서 효율적으로 버퍼를 관리하는 문제는 일반 데이터베이스 환경에서 보다 훨씬 중요한 문제가 된다.
지금까지 알려진 버퍼관리 방법들은 버퍼를 관리할 때에 시간적 근접성(temporal locality) 만을 고려해왔다. 그러나 공간 데이터베이스에는 공간적 근접성(spatial locality) 도 있기 때문에 기존의 버퍼 관리 방법들은 공간 데이터베이스에서는 최적의 성능을 보여주지 못한다. 따라서 기존의 버퍼 관리 방법들의 문제점을 해결하여 SDBMS 의 성능을 향상시킬 수 있는 버퍼 관리 방법이 필요하다.
본 논문에서는 시간적 근접성과 공간적 근접성을 동시에 고려하여 버퍼를 관리함으로써 SDBMS 의 성능을 향상시키는 버퍼관리 방법을 제안하고 있다. 공간 데이타의 특성을 잘 표현할 수 있는 것이 객체지향데이터베이스(OODBMS)이므로 이 논문은 기반시스템으로 OODBMS를 이용한다.
본 논문의 구성은 다음과 같다. 2 장은 기존의 버퍼관리방법과 SDBMS 에 대해 설명하고 3 장에서는 이 논문에서 제안하는 SLM-tree 버퍼 관리 방법에 대해 기술하고 있다. 4 장에서는 실험과 결과의 분석을 다루고 있다. 마지막으로 5 장에서는 결론에 대하여 논의한다.

2. 관련연구

2.1 버퍼관리방법과 필요성

데이터베이스에 존재하는 특정 데이터에 대한 요청을 논리적인 참조(logical reference)라고 한다[1]. 각각의 논리적인 참조에 대해서 버퍼 관리자는 그 객체가 버퍼에 있는지 없는지를 알아 본다. 만일 버퍼에 찾고자 하는 객체가 있다면, 그 객체를 이용할 수 있다. 그러나 데이터가 버퍼에 존재하지 않고 버퍼에 새로운 데이터를 적재할 공간이 없다면, 요구된 데이터가 적재될 수 있는 공간을 확보하기 위해서 버퍼의 객체들 중에서 희생자들(victims)을 선택해야 한다. 버퍼 교체 알고리즘을 통해 희생자들을 버퍼에서 제거한 후에 버퍼에 충분한 공간이 확보되면 객체를 디스크에서 읽어와서 버퍼에 적재한다. 이처럼 디스크에 있는 데이터를 참조하는 것을 물리적 참조(physical reference)[1] 라고 하며 이것은 DBMS 에서 가장 많은 비용을 요구하는 동작(operation)들 중의 하나이다. 따라서 버퍼 교체 알고리즘을 최적화하는 것은 DBMS 의 성능을 향상시키는 데에 매우 중요한 요소라고 할 수 있다.
데이터에 대한 논리적 참조들이 가진 가장 중요한 속성은 참조 행동에 존재하는 근접성이다. 시간적 근접성이란 최근에 참조한 데이터가 다시 참조될 확률이 다른 데이터들을 참조할 확률보다 높다는 것이다. 모든 교체 알고리즘들은 재참조 될 가능성이 가장 작은 데이터를 교체하기 위해서 과거의 참조 패턴을 이용하는데, 이때 시간적 근접성을 가정하고 이용한다.

2.2 지금까지의 버퍼관리방법들

가장 널리 알려진 LRU[1] 알고리즘은 가장 오랫동안 사용한 적이 없는 데이터를 교체한다. 또한 LRU-K[2] 알고리즘은 데이터에 대한 접근 빈도 정보도 고려하여 버퍼에 보관해야 할 데이터와 버려야 할 데이터를 더 잘 구분할 수 있게 하며 2Q[3] 알고리즘은 많은 비용이 드는 LRU-K 알고리즘의 문제를 해결 하면서 LRU-K 수준의 성능을 보여 줄 수 있는 알고리즘으로 제안되었다. 그 외에도 FIFO[1], LFU[1], CLOCK[1] 그리고 GCLOCK 등의 알고리즘들이 있다. 그러나 위의 알고리즘들은 공간 데이터베이스의 특징을 고려하지 않고 있기 때문에 공간 데이터베이스에서 좋은 성능을 보여주기 힘들다.

2.3 SDBMS의 특징

2.3.1 MBR

SDBMS에서 공간 선택(spatial selection) 을 효과적으로 하기 위한 공간 인덱스[8]는 복잡한 실제 공간 데이터 값보다 훨씬 간단한 키 값을 이용하여 객체들을 관리하기위해서 근사화(approximation)를 한다. 본 논문에서 제시하는 방법은 공간 인덱스에 사용하는 근사화 방법인 MBR[7] 을 이용하고 있다.

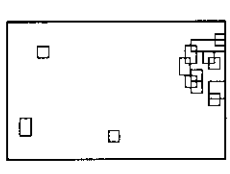

그림 1 –공간 데이터베이스에서 질의의 분포

2.3.2 공간적 근접성

공간 데이터베이스는 데이터를 참조하는 패턴에도 기존의 데이터베이스와 차이

[1] 본 연구는 한국과학재단의 목적기초연구(과제번호 99-2-315-001-3) 지원으로 수행됨

가 존재한다. 질의들은 지도 전체에 고르게 분포되어 있지 않고 지도에서 특정한 부분에 집중되기 때문에 특정 지역의 데이터가 자주 사용되는 공간적 근접성(Spatial Locality)[4] 이 나타난다. 그림1은 전체 지도상에서 질의 영역이 집중되는 예를 보여 준다. 작은 사각형들은 각각의 질의의 질의 영역이다. 공간 데이터베이스에서 시간적 공간적 근접성을 잘 고려하면 시스템의 성능을 향상시킬 수 있다.

3. SLM-tree 버퍼 관리 기법

공간 데이터베이스에는 공간적 근접성이 존재하기 때문에 버퍼에서 교체 대상을 선정할 때, 사용된 지는 오래되었지만 현재의 공간적 근접성 영역 안에 속하는 객체 보다는 사용된 지 얼마 되지 않았지만 공간적 근접성 영역과 멀리 떨어져 있어서 공간적으로 재 참조될 가능성이 적은 데이터를 선택하는 것이 좋다. 그러나 지금까지의 시간적 근접성만을 고려하는 방법들은 사용된 지는 오래 되었지만 공간적 근접성 영역에 존재해서 재 참조될 가능성이 큰 객체를 교체 대상으로 선정하는 게 일반적이었다. 물론 재 참조될 가능성이 큰 객체를 교체 대상으로 선택하기 때문에 성능이 나빠지게 된다.

SLM-tree 버퍼 관리 방법은 가장 최근에 사용된 객체들이 존재하는 위치가 대개의 경우 현재의 공간적 근접성 영역과 유사할 것이라고 가정하고, 추정된 공간적 근접성 영역 밖에 있는 객체들을 우선 교체 대상으로 삼는다. 공간 데이터베이스에서 재 참조될 가능성이 적은 객체를 교체 대상으로 선정하기 때문에 SLM-tree 버퍼 관리 방법은 기존의 방법들보다 뛰어난 성능을 보여줄 것으로 기대할 수 있다.

SLM-tree 버퍼 관리 모듈은 크게 공간적 근접성 영역을 추정하는 SLM-tree와 추정된 공간적 근접성 영역을 이용해서 교체대상을 선정하는 수정된 LRU 큐의 두 가지 구조체로 이루어져 있다.

3.1 SLM-tree

이진 트리 구조의 SLM-tree는 최근에 참조된 객체들의 MBR을 이용하여 그 MBR들을 모두 포함하는 가장 작은 MBR 을 만드는 것이 목적이다. 최근에 참조된 8 개의 객체를 고려하는 SLM-tree는 다음의 그림 2 과 같이 구성된다.

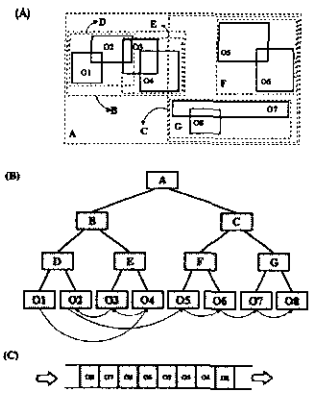

그림 2 - SLM-tree 의 구조와 구성

그림2에서 (A) 는 SLM-tree가 고려하고 있는 실제 지도상에 존재하는 객체들의 MBR 들이다. O1 에서 O8 까지는 8 개의 객체들의 MBR 들이며 그것들을 포함하는 A 에서 G 까지의 MBR 은 SLM-tree를 구성하는 내부 노드들의 MBR 들이다. (B)는 (A) 의 객체들이 SLM-tree를 이루고 있는 모습이다.

SLM-tree는 최근에 사용된 n 개 객체들의 MBR 들만을 고려하므로 새로운 SLM-tree가 새로 참조된 객체의 MBR 을 고려해야 할 때 SLM-tree에서 가장 오래된(Least recently referenced) 객체의 MBR 을 트리에서 제거해야 한다. 따라서 말단 노드들은 그림 3 의 (B) 처럼 순서를 가진 리스트 구조를 유지한다. 그림 2 의 (C) 는 B 의 말단 노드들 간에 존재하는 순서를 큐의 형태로 도식화한 것이다.

그림 2 의 (B) 그림에서 말단 노드들은 모두 최근에 참조된 O1 에서 O8 까지의 객체들이다. 각 노드의 부모 노드들은 두개의 자식 노드들을 모두 포함하는 MBR을 유지하게 되고 결과적으로 루트 노드 A 의 MBR 은 모든 말단 노드의 객체를 모두 포함하는 MBR 을 가지고 있게 된다. 그리고 이 루트 노드 A 는 최근에 참조된 객체들의 분포를 반영하고 있으므로 최근의 공간적 근접성을 반영하는 공간적 근접성 영역으로 가정하고 사용할 수 있다.

그림3은 그림2의 상태에서 새로운 객체인 O9 이 참조되었을 때 어떻게 SLM-tree에 반영되는 지를 보여주고 있다. SLM-tree에서는 항상 한 개의 객체가 나가고 하나의 객체가 들어오며 말단노드에서 객체들의 위치는 상관없으므로 SLM-tree에 새로운 객체를 넣을 때 삭제되는 객체의 위치에 새로운 객체를 대치하여 적은 비용으로 객체를 넣고 뺄 수 있다. 또한 이렇게 트리를 유지하면 정적인 트리 구조를 유지할 수 있다. 항상 최근에 참조된 8 개의 객체들만을 고려한다고 하였으므로 새로운 객체인 O9 을 트리에 반영하기 위해서 지금까지 고려했던 8 개의 객체들 중에 참조된 지 가장 오래된 객체인 O1 을 트리에서 제거해야 한다. 그림 3 의 큐는 고려대상중인 객체들의 큐에서 가장 오래된 객체 O1 이 나가고 새로운 객체 O9 이 추가됨을 보여준다.

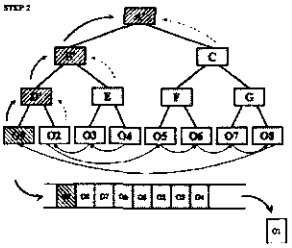

그림 3 - SLM-tree의 새로운 객체 참조

일단 O1 이 있던 자리에 O9 이 들어가게 되면, 변형된 데이터를 루트 노드 MBR에 반영해야 한다. O9 의 부모 노드인 D 는 자식 노드인 O9 과 O2 를 포함하는 MBR 을 가지는 D'으로 바뀌게 된다. 같은 방법으로 B 와 A 는 B'과 A'으로 바뀌게 된다. 적은 비용으로 루트 노드인 A' 노드에는 변형된 말단 노드들의 객체들을 반영하는 MBR 이 생기게 된다.

3.2 수정된 LRU 큐

수정된 LRU 큐는 처음 객체를 삽입할 때는 LRU 큐처럼 한 쪽 끝에서 삽입그러나, 객체를 빼낼 때에는 중간에 있는 객체를 빼낼 수 있으며 중간의 객체를 빼내기 위해 각각의 객체를 뒤에서부터 탐색해 갈 수 있다.

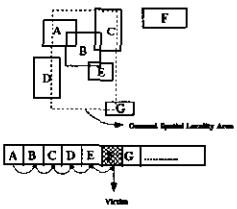

그림 4 - 수정된 LRU 큐의 교체 대상 선정

그림 4 에서 점선으로 된 사각 영역은 SLM-tree를 이용해서 공간적 근접성 영역으로 추정된 부분이고 A 에서 G 까지의 사각형은 각 객체의 MBR 들을 의미한다. 큐는 변형된 LRU 큐에 어떤 순서로 객체들이 쌓여있는 지를 보여 주고 있다. 교체 대상을 선정할 때 시간적 근접성에 근거하여 교체대상 객체는 큐의 뒤에서부터 찾게 된다. 처음에는 A 가 SLM-tree를 통해 추정된 공간적 근접성 영역과 겹치는 지를 확인한다. 그림에서 객체 A 의 MBR 은 추정된 공간적 근접성 영역과 겹치므로 공간적 근접성 영역에 속하는 객체로 생각되어 교체 대상으로 선정되지 않는다. 그리고 다음에 있는 객체 B 와 그 이후의 객체들에 대해서도 같은 방식으로 겹침 여부 테스트를 한다. 그런데 객체 F 의 경우 공간적 근접성 영역과 겹치지 않으므로 공간적 근접성 영역에 포함되지 않는 객체로 생각되어 교체 대상으로 선정된다. 이와 같은 방법으로 버퍼에서 앞으로 사용될 가능성이 적은 객체를 교체대상으로 선정한다.

4. 실험과 분석

본 절은 제안하고 있는 SLM-tree 버퍼 관리 방법의 성능 실험을 다루고 있다. 시스템의 성능은 적중률(hit ratio)과 응답 시간(response time)으로 측정하였다.

4.1 수정된 SLM-tree 버퍼관리기법

가끔씩 공간적 근접성 영역과 많이 떨어져 있는 객체를 참조할 경우가 발생한다. 이런 경우 SLM-tree는 멀리 떨어져 있는 객체도 포함하게 되어 공간적 근접성 영역을 너무 크게 추정한다. 이렇게 공간적 근접성 영역이 크게 추정되면 정확하지도 않을 뿐더러 클라이언트 버퍼에서 추정된 공간적 근접성 영역에 속하지 않는 객체를 찾기도 힘들어 진다.

수정된 SLM-tree 방법에서는 공간적 근접성 영역에 속하지 않는 객체를 찾을 때 버퍼의 뒤에서 75% 정도의 영역까지만 찾아보고 객체를 찾지 못하면 현재 공간적 근접성 영역이 너무 크게 추정 되었다고 가정하여 탐색을 중지한다.

4.2 실험환경

표 1 은 실험에 사용된 데이터들의 설정을 보여주고 있다.
전체 데이터에서 약 3% 가 참조되는 빈도가 높은 공간적 근접성 영역 안의 데이터이다. 시크 타임(Seek time) 은 9msec 으로 가정하였고, 평균 레이턴시 타임(latency time) 은 6msec 으로 가정하였으며 4KB를 전송하는 데 소요되는 전송 시간(transfer time)은 1msec 으로 가정하였다[5].
모든 실험은 SLM -tree 버퍼관리방법과 수정된 SLM -tree 버퍼관리방법의 LRU 버퍼관리방법에 대한 상대적인 성능으로 나타내었다.

Database size	100 M
Average object size	1KB
Number of Objects	1,000,000
Map Size	10,000 x 10,000
Spatial Locality Area Size	1,700 x 1,700

표 1

4.3 실험

그림 5 와 그림 6 은 전체 데이터베이스에 대한 상대적인 클라이언트의 버퍼 크기가 데이터 적중률과 응답 시간에 어떤 영향을 미치는 지를 알아보기 위한 실험 결과이다. 그림 5 에서 버퍼 크기가 1.5%~2% 로 버퍼에 대한 경쟁이 적당할 때 SLM -tree 버퍼관리방법의 상대적인 적중률이 최고조에 이르고 대부분의 경우에 좋은 적중률을 보여주는 것을 알 수 있다. 버퍼의 크기가 너무 커지게 되면 버퍼에 공간적 근접성 영역의 객체들이 적재될 공간이 충분하기 때문에 점차 LRU 방법과 비슷한 성능을 보이게 된다. 그림 6 을 통해 응답시간의 측면에서도 적절한 버퍼 크기인 경우 최고 15% 까지 좋은 성능을 보여줌을 알 수 있다.

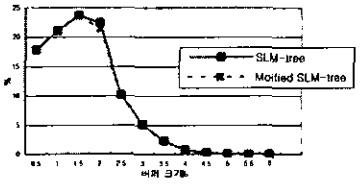

그림 5 - 버퍼크기에 따른 상대적 적중률

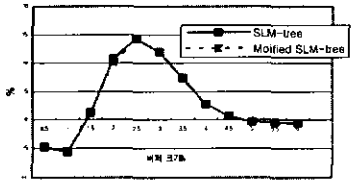

그림 6- 버퍼 크기에 따른 상대적 응답 시간

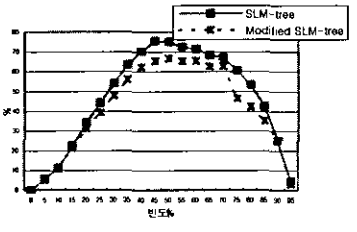

그림 7 - 빈도에 따른 상대적 히트비율

그림 7 은 공간적 근접성 영역 외의 객체를 참조하는 비율에 따라서 얼마나 성능차이가 나는가에 대한 실험 결과이다. 결과를 통해 빈도에 따라 최고 70% 까지 적중률이 향상됨을 알 수 있다. 이는 공간적 근접성 외의 객체를 참조하는 비율이 적절히 높을 경우에 버퍼에서 쉽게 교체대상을 선정할 수 있기 때문이다. 그러나 너무 빈도가 높을 경우에는 공간적 근접성 영역을 잘못 추정하는 경우가 많아서 적중률이 떨어지게 된다.
그림 8 은 SLM -tree가 고려하는 객체의 수에 따른 성능 실험이다. 환경에 따른 적절한 수의 객체들을 고려해야 가장 좋은 성능을 얻을 수 있다는 것을 보여준다. 고려하는 객체수가 너무 많게 되면 공간적 근접성 영역을 잘못 추정하는 경우가 많이 발생해서 성능이 조금씩 떨어지게 된다.

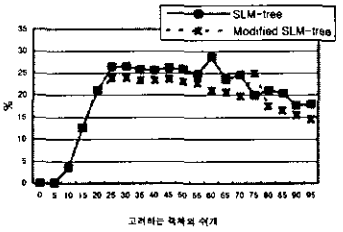

그림 8 - 고려하는 객체 수에 따른 상대적 성능

5. 결론

데이터베이스를 이용할 때 서버의 디스크에 저장되어 있는 데이터를 가져오는 것은 전체 시스템의 성능 저하를 가져오는 가장 큰 요인들 중의 하나이다. 따라서 시스템 성능을 개선하기 위해서는 버퍼 관리 모듈에서 디스크 I/O 를 최소화 해야 한다.
본 논문에서 제안하는 SLM -tree 버퍼 관리 방법은 공간 데이터베이스 환경에서 버퍼를 관리할 때 시간적 근접성과 공간적 근접성을 같이 고려하여 시스템의 성능을 향상시켰다.
LRU 버퍼 관리 방법과 비교한 실험을 통해 *SLM -tree* 버퍼 관리 방법이 적중률 측면에서 많은 향상을 보이며, 응답 시간 측면에서도 적지 않은 성능 향상을 보여 줌을 알 수 있다. 본 논문은 공간 데이터베이스에서 나타나는 공간적 근접성을 이용하여 버퍼를 관리하여 전체 시스템 성능이 향상될 수 있다는 것을 보여주었다는 것에 그 가치가 있다고 할 수 있다.
앞에서 살펴보았듯이 SLM -tree는 공간적 근접성 영역 외의 객체에 의해 영향을 받아 부정확한 근접성 영역을 추정하게 되는 경우가 종종 발생한다. 따라서 공간적 근접성 영역을 추정하는 과정에서 공간적으로 멀리 떨어진 객체들을 제외하기 위한 연구가 좀 더 진행 되어야 할 것이다. 또한 본 논문은 한 순간에 하나의 공간적 근접성만 존재한다는 가정을 하고 있으므로 동시에 두 개 이상의 공간적 근접성이 나타날 때 어떻게 공간적 근접성을 측정하고 그 값을 이용할 것인지에 대한 연구가 더 진행되어야 할 것이다.

참 고 문 헌

[1] W. Effelsberg, T. Harder "Principles of Database Buffer Management ," ACM Transactions on Database Systems, Vol. 9, No. 4, December 1984, pp.560-595.
[2] E. J. O'Neil, P. E. O 'Neil, G. Weikum "The LRU-K Page Replacement Algorithm For Database Disk Buffering, " ACM SIGMOD Conference, May, 1993, pp. 297-306.
[3] T. Johnson and D. Shasha "2Q: A Low Overhead High Performance *Buffer Management Replacement Algorithm. " Proceedings of the 20* th VLDB Conference, 1994, pp.439-450.
[4] L. Ki -Joune and L. Robert "The Spatial Locality and a Spatial Indexing Method by Dynamic Clustering in Hypermap System, " Advances in Spatial Databases, 1990, pp.207-223.
[5] T. Brinkhoff and H. Kriegel "The Impact of Global Clustering on Spatial Database System," Proceedings of the 20 th VLDB Conference, 1994, pp.168-179.
[6] A. Guttman "R-TREES: A Dynamic Index Structure for Spatial Searching," Proceedings of the ACM SIGMOD Conference, June 1984, pp.47-57.
[7] *R. Guting "An Introduction to Spatial Database Systems, "* the VLDB Journal, Vol. 3, No. 4, October 1994, pp. 357-399.