디클러스터된 공간 데이타베이스에서 다중 질의의 병렬 처리

(Parallel Processing of Multiple Queries in a Declustered Spatial Database)

서 영 덕 [†] 박 영 민 [††] 전 봉 기 [†] 홍 봉 희 [†††]
(Young Duk Seo) (Young Min Park) (Bong Gi Jeon) (Bong Hee Hong)

요 약 다중 공간 질의는 동시에 2개 이상 수행되는 영역 질의로 정의되며 인터넷 기반 지도 보기 응용의 주요 연산이 되므로, 질의 처리 속도의 개선을 위해서 병렬로 처리되어야 하고 질의 처리 비용 중 큰 비중을 차지하는 디스크 입출력 시간을 최대한 줄일 필요가 있다. 그런데 다중 CPU/다중 디스크 구조상에서 디클러스터링을 수행하더라도, 다중 공간 질의를 처리하는 도중 질의 간 임의 탐색이 발생하여 디스크 입출력 시간이 증가하는 문제점이 있다.

이 논문에서는 디클러스터된 공간 데이타베이스에서 다중 공간 질의를 처리할 때 발생하는 문제점인 질의 간 임의 탐색을 분석하고, 해결 방안으로 질의 스케줄링 기법을 제시하였다. 질의 스케줄링 기법은 질의 간 관련성을 이용해서 질의 우선 순위를 조정해서 질의 간 임의 탐색을 해결하는 방법으로써, 질의 간의 공간 및 시간 관련성 부여를 위해 질의 간 위치 관련성과 질의 대기 시간을 이용하였다. 실험 결과, 질의 스케줄링을 수행하면 디스크 캐쉬의 적중률이 최대 34% 향상되어 디스크 입출력 비용을 최대 6%까지 줄일 수 있어 다중 공간 질의 처리 시의 성능을 개선할 수 있는 것으로 나타났다.

키워드 : 영역 질의, 디클러스터링, 다중 공간 질의, 병렬 질의 처리, 질의 스케줄링

Abstract Multiple spatial queries are defined as two or more spatial range queries to be executed at the same time. The primary processing of internet-based map services is to simultaneously execute multiple spatial queries. To improve the throughput of multiple queries, the time of disk I/O in processing spatial queries significantly should be reduced. The declustering scheme of a spatial dataset of the MIMD architecture cannot decrease the disk I/O time because of random seeks for processing multiple queries.

This thesis presents query scheduling strategies to ease the problem of inter-query random seeks. Query scheduling is achieved by dynamically re-ordering the priority of the queued spatial queries. The re-ordering of multiple queries is based on the inter-query spatial relationship and the latency of query processing. The performance test shows that the time of multiple query processing with query scheduling can be significantly reduced by easing inter-query random seeks as a consequence of enhanced hit ratio of disk cache.

Key words : Range query, Declustering, Multiple spatial query, Parallel query processing, Query scheduling

· 본 연구는 한국과학재단 목적기초연구(2000-1-51200-004-2)지원으로 수행되었음
† 학생회원 : 부산대학교 컴퓨터공학과
ydseo@pusan.ac.kr
bgjun@hyowon.cc.pusan.ac.kr
†† 비 회 원 : 엔텔시스템 연구원
ymnpark@dreamwiz.com
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
부산대학교 컴퓨터및정보통신연구소 연구원
bhhong@pusan.ac.kr
논문접수 : 2001년 5월 11일
심사완료 : 2001년 9월 7일

1. 서 론

지리 정보 시스템에서 사용하는 공간 질의 중 가장 보편적인 영역 질의는 단일 주사 질의(single scan query)로써, 사용자가 정의한 질의 창(query window)과 교차하는 공간 객체들을 결과로 삼는 공간 질의이다. 영역 질의는 다중 주사 질의(multiple scan query)인 공간 조인과는 달리, 전체 질의 처리 시간에서 CPU 처리 시간보다 공간 데이타에 접근하기 위한 디스크 입출

력에 소요되는 시간이 큰 특징이 있다. 영역 질의는 최근 많이 사용되는 클라이언트-서버(client-server) 구조의 인터넷 기반 지도 보기 응용에서 중요성이 더욱 강조되고 있다.

지도 보기 응용은 다수의 사용자(클라이언트)가 단일 서버에 저장된 대용량의 기본도로부터 자신이 지정한 관심 영역을 전송 받은 후, 해당 지역을 중심으로 이동하거나 확대, 축소해서 보는 서비스를 제공하는 지리 정보 시스템의 한 응용 분야이다. 클라이언트는 사용자가 관심을 가지는 지역에 대해 영역 질의를 수행하고, 서버는 클라이언트가 수행한 영역 질의를 처리해서 그 결과를 전송한다. 현재 인터넷 기반 지도 보기 응용 서비스를 제공하는 웹 사이트가 여러 곳 존재하며 향후 그 서비스가 더욱 확대될 전망에 있으므로, 응용의 성능 향상을 위해 서버 측에서 클라이언트가 요청한 영역 질의를 빠른 시간 내에 처리할 수 있는 방법이 필요하게 된다.

그런데 이 때의 영역 질의는 순차적으로 처리되는 일반적인 영역 질의의 형태가 아니라 다수의 독립적인 클라이언트에서 동시에 수행되는 영역 질의의 형태를 가지게 된다. 이 논문에서는 이러한 형태의 질의를 다중 공간 질의(multiple spatial query)로 정의한다. 다중 공간 질의는 동시에 수행되는 2개 이상의 영역 질의의 집합으로써, 디스크 입출력 비용이 큰 영역 질의의 문제점을 그대로 가지고 있고, 다중 공간 질의의 고유한 문제점인 질의 간 임의 탐색(inter-query random seek)이 발생하여 질의 처리 시간이 증가하는 단점이 있다. 기존 연구에서는 영역 질의를 처리할 때 발생하는 디스크 입출력 비용을 줄이기 위한 기법으로, 공간 데이타를 다중 디스크 상에 논리적 관련성을 이용하여 배치하는 디클러스터링(declustering)을 이용하였다. 그런데 질의 간 임의 탐색은 디클러스터링으로 해결하기 어려운 문제점이 있다.

이 논문에서는 디클러스터링 기법을 사용한 공간 데이타베이스 상에서 다중 공간 질의를 처리할 때 발생하는 질의 간 임의 탐색을 줄이기 위해, 질의 간 위치 관련성 및 질의 대기 시간을 이용한 질의 스케줄링 알고리즘을 제안한다. 질의 스케줄링 알고리즘은 질의 처리 시의 우선 순위를 동적으로 조정하는 기법으로, 질의들의 공간 및 시간적 관련성을 이용하여 질의 우선 순위를 할당하게 된다. 질의 스케줄링을 수행하게 되면 관련성이 높은 질의들을 연속적으로 처리할 수 있게 되어 캐쉬의 적중률이 증가하고, 디스크 입출력 비용을 감소시킬 수 있다.

이 논문의 구성은 다음과 같다. 2장에서 영역 질의를 처리할 때 디스크 입출력 비용을 줄이기 위한 기존 연구를 기술하고, 3장에서 다중 공간 질의를 정의한 후 다중 공간 질의의 문제점인 질의 간 임의 탐색을 분석한다. 4장에서는 다중 공간 질의 처리 모델을 제시한 후, 질의 스케줄링을 위해 필요한 개념인 질의 간 위치 관련성과 질의 대기 시간, 질의 처리 이력에 대해서 설명하고, 이들을 이용한 질의 스케줄링 기법에 대해 기술한다. 5장에서 구현 및 성능 평가 실험을 통해 질의 스케줄링 기법에 대한 성능을 분석하고, 6장에서 결론 및 향후 연구로 끝을 맺는다.

2. 관련 연구

다중 공간 질의를 구성하는 영역 질의를 처리할 때 발생하는 디스크 입출력 비용을 줄이기 위한 방법론은 지금까지 많이 연구되어 왔다. 대표적인 것으로 디스크에 공간 데이타를 배치하는 방법인 디클러스터링(declustering) 기법과 클러스터링(clustering) 기법에 관한 연구가 있다. 그런데 클러스터링은 단일 디스크 구조에서 데이타의 논리적 인접성을 물리적으로 반영할 수 있도록 저장하는 방법이므로, 다중 디스크 구조를 기반으로 하는 이 논문에서는 다루지 않는다. 디클러스터링은 데이타를 다중 디스크 상에 적절한 기준에 따라 분산 배치하는 기법으로써, 크게 디클러스터링 기법에 대한 연구와, 기존에 제시되었던 여러 기법에 대한 분석 및 성능 평가에 관한 연구로 나눌 수 있다. 여기서는 공간 데이타베이스를 기반으로 수행되는 지도 보기 응용의 특성상 관계형 데이타베이스에서의 디클러스터링에 대한 연구는 살펴보지 않는다.

공간 데이타를 어떻게 분산 배치할 것인가에 대한 디클러스터링 기법에 대한 연구는 크게 R-Tree 공간 색인 계열과 고정 그리드(fixed-grid) 공간 색인 계열로 나누며, 그래프(graph)를 사용한 계열로 분류할 수 있다.

우선 R-Tree 계열에서의 디클러스터링에 관한 연구로 [1]과 [2]가 있다. [1]은 단일 CPU/다중 디스크 상에서의 R-Tree를 사용한 디클러스터링 기법에 대해 기술하였다. R-Tree의 각 노드(node)들을 다중 디스크 상에 분산시킨 MX R-Tree 구조를 기반으로 하여, 인접성 색인(proximity index) 개념을 도입하여 효율적으로 공간 데이타를 분산시킬 수 있는 방법을 제시하였다. [2]는 비공유 다중 컴퓨터(shared-nothing multi-computer) 구조 상에서 R-Tree를 이용하여 디클러스터링을 수행하는 방법에 대한 연구를 하였는데, 다중 디스크상에 배치하는 공간 데이타의 디클러스터링 단위(chunk)를 최적화할 수 있는 공식에 관해 연구하였다. 그러나 LAN

을 이용해 다중 컴퓨터를 구현하였기 때문에, 영역 질의를 수행할 때 다중 컴퓨터를 구성하는 각 노드들간의 통신 비용이 필요 이상 큰 문제점이 있다.

고정 그리드 색인을 사용한 디클러스터링 기법으로는 CMD[3], Z-Order[4], HCAM[5], FX[6]등을 들 수 있는데, 이들은 고정 그리드 색인의 셀을 디스크에 할당하는 방법에 따라 구분할 수 있다. CMD 기법[3]은 고정 그리드 색인을 구성하는 각각의 셀(cell)에 X, Y 좌표를 할당한 후, 해당 좌표 값의 합을 디스크 개수로 나눈 나머지를 이용하고, Z-Order 기법[4]과 HCAM 기법[5]은 공간 데이타들의 공간적 인접성을 보존하기 위해 사용하는 공간 채움 커브(space filling curve)중에서 각각 Z-Order curve와 Hilbert curve를 사용한다. FX 기법[6]은 셀의 좌표 값을 이진수(binary)로 변환한 후 해당 이진수 값을 서로 XOR한 결과를 사용하게 된다. 각 방법의 예시는 그림 1과 같다. 2.2절에서 기술하지만, 일반적인 성능은 CMD와 HCAM 기법이 우수하다고 알려져 있다.

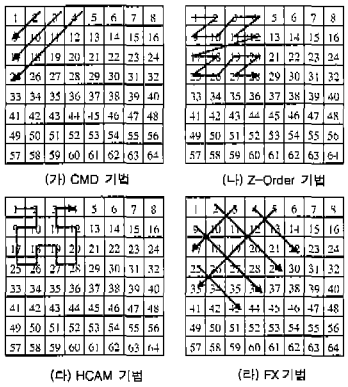


그림 1 디클러스터링 방법의 예시

기존의 디클러스터링 기법 중 고정 그리드 색인을 사용하는 디클러스터링 기법들을 분석하고 성능을 평가한 연구로 [7], [8], [9]가 있다. [7]은 CMD[3], HCAM[5], FX[6]에 대해서 다양한 분포상태를 데이타 집합을 대상으로 다양한 실험을 수행하였다. CMD 기법의 경우 디스크 개수가 작을 때는 성능이 좋지만 디스크 개수가 증가하여도 성능이 향상되지 않으며, 디스크 개수가 많을 때는 HCAM기법의 성능이 좋다는 사실을 제시하였다. 그리고 HCAM 기법은 데이타 분포가 균등할 때 가장 성능이 우수하며, FX 기법은 디스크 개수에 따라서 확장성의 제한이 있다는 것을 밝혀 내었다.

[8]은 일반적으로 많이 사용되는 디클러스터링 방법인 CMD와 FX의 확장성에 대해서 증명을 통한 수식과 실험을 통해 연구하였다. 그 결과, 디스크 개수가 작고 질의 영역의 범위가 큰 질의 상황에서는 디클러스터링 기법들의 성능이 서로 유사하였다는 사실을 제시하였다. 그리고 CMD 기법은 다양한 환경 하에서도 괜찮은 성능을 발휘하므로 적절한 선택이 될 수 있고, FX 기법은 디스크의 개수가 2의 제곱이 되지 않으면 확장성이 크게 저하되고 질의의 크기가 클 경우 성능이 저하될 수 있는 단점이 있다는 사실을 밝혀 내었다. 한편 [9]는 CMD, Z-Order, HCAM, FX 기법들을 분석해서, 모든 종류의 영역 질의에 대해 최적의 성능을 발휘하는 방법은 없다는 것을 이론과 증명을 통해 도출하였다. 그리고 다양한 질의 형태와 공간 데이타 분포 상태에 따른 실험을 통해 CMD와 HCAM 기법이 영역 질의의 경우 우수한 성능을 낼 수 있고, 데이타 분포 상태가 각 방법의 성능에 영향을 줄 수 있다는 사실을 연구하였다.

3. 다중 공간 질의의 문제점

이 장에서는 다중 공간 질의를 효율적으로 처리하기 위해 다중 공간 질의를 정의하고, 다중 공간 질의의 문제점을 분석한다. 3.1절에서 다중 공간 질의를 정의하고, 3.2절에서 다중 공간 질의의 고유한 특징과 다중 공간 질의의 처리 환경에 대해 기술한다. 그리고 3.3절에서 다중 공간 질의의 문제점인 질의 간 임의 탐색에 대해서 기술한다.

3.1 다중 공간 질의의 정의

이 논문에서 다루는 다중 공간 질의는 아래와 같이 정의한다.

[정의 1] 다중 공간 질의(Multiple Spatial Query) : For any given time interval δ

$$MSQ = \{Q_i \mid |Time_{push}(Q_i) - Time_{push}(Q_j)| < \delta, i, j \in n \geq 2, Q_i \in RQ\}$$

Q_i : Time i에 들어온 영역 질의

MSQ : 다중 공간 질의(Multiple Spatial Query)

RQ : 영역 질의(Region Query)

Timepush(Qi) : 질의 Qi가 질의 처리 큐[1]에 들어오는 시간(timestamp)

다중 공간 질의를 구성하는 영역 질의(RQ)는 다음과 같이 정의한다.

[정의 2] 영역 질의(Region Query)

1) 질의 처리 큐(query processing queue)는 4.1절에서 설명한다.

-질의 영역이 임의의 두 점 (xmin, ymin), (xmax, ymax)으로 구성되는 최소 경계 사각형(MBR) 형태로 주어지며, 질의 영역과 교차하거나 겹치는 모든 공간 객체들을 질의 결과로 삼는 공간 질의

다중 공간 질의는 서버가 클라이언트로부터 들어온 질의들을 질의 처리 큐(queue)에 삽입하는 시간을 기준으로 구성된다. 질의가 큐에 삽입되는 시간이 실제로 동일할 수 없으므로, 시간 차이 δ를 두어 δ 이내에 큐에 들어오는 영역 질의들을 다중 공간 질의로 구성한다. 즉 다중 공간 질의는 동일 시간대에 큐에 삽입된 다수의 독립적인 영역 질의의 집합으로 볼 수 있다. 다중 공간 질의의 예는 그림 2와 같다.

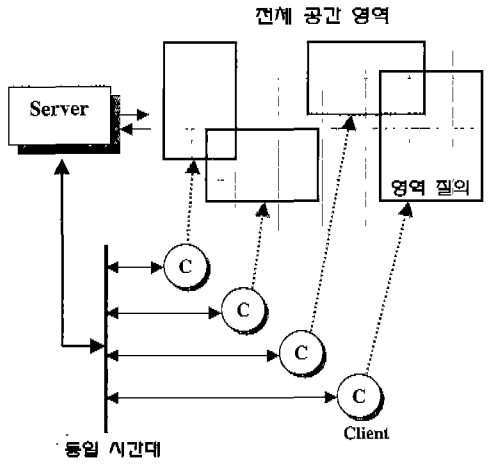


그림 2 다중 공간 질의의 예시

3.2 다중 공간 질의의 특징 및 처리 환경

다중 공간 질의가 주 연산이 되는 인터넷 기반 지도 보기 응용은 다음과 같은 특징을 가지고 있다.

- 클라이언트들은 영역 질의를 순차적으로 수행하고, 각 클라이언트들의 영역 질의는 서로 독립적으로 다른 클라이언트의 영향을 받지 않는다.
- 클라이언트들은 처음 질의를 수행한 후, 첫 질의 영역의 주변 지역에 대해 질의를 수행할 확률이 있다.
- 서버는 클라이언트에서 수행하는 질의를 가능한 빠른 시간 내에 처리해야 한다.
- 서버는 질의 처리 이전에 클라이언트의 질의 영역을 알 수 없다.

다중 공간 질의는 위와 같은 지도 보기 응용의 특징과 정의 1에서 알 수 있는 다음과 같은 특징을 가지고 있다.

[특징 1] 전체 질의 대상 영역 중 질의가 집중되는 지역이 발생할 수 있다.

-2개 이상의 클라이언트가 공통적으로 관심이 있는 지역(시내 중심가나 상가 밀집 지역 등)을 선택해서 질의를 수행할 수 있다.

[특징 2] 질의들을 적정 임계 시간 안에 처리해야 한다.

-서버는 처리해야 할 모든 질의에 대해, 클라이언트에서 질의 처리를 요청한 시간부터 적절한 처리 한계 시간 내에 수행해야 한다. 즉 특정한 클라이언트의 질의 처리가 지연되어 질의 처리 시간이 증가하는 문제점을 방지해야 한다.

[특징 3] 이전에 처리했던 질의들의 정보를 질의 처리 중에 유지할 수 있다.

-다중 공간 질의는 영역 질의의 집합으로 구성되므로, 각 질의들을 처리하는 도중에 처리했거나 처리 예정인 다른 질의들의 위치 정보를 알 수 있고, 이 정보를 질의 처리에 이용할 수 있다.

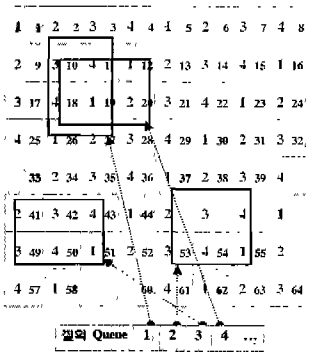
다중 공간 질의는 2개 이상의 영역 질의로 구성되고 이를 빠른 시간 내에 처리해야 한다. 따라서 데이타베이스를 병렬화하는 두 가지 목적인 *speedup*과 *scaleup* 중에서, 다수의 질의를 다수의 CPU에 할당하여 수행함으로써 동시에 처리하는 작업의 수를 늘리는 *scaleup*이 적절하다[12]. 그 중 공유된 데이타베이스에 대해 다수의 질의가 수행되는 형태인 *transaction scaleup*으로 볼 수 있다[12].

그런데 공간 데이타가 공유된 단일 디스크에 저장된 구조에서 다중 공간 질의를 처리할 경우, 디스크에 접근하는 CPU의 수가 증가함에 따라 디스크 병목 현상이 발생하게 되어 병렬로 질의를 처리할 때 성능 저하 요인이 된다. 공간 데이타의 양이 방대하고 처리해야 하는 질의의 수가 많다면 디스크 입출력 시간이 더욱 증가하므로, 기존 연구에서는 이를 해결하기 위해 다중 디스크 구조에서 디클러스터링 기법을 사용하여 디스크 입출력 시간을 감소시켰다[1, 3]. 다중 공간 질의를 구성하는 영역 질의는 전체 처리 시간에서 디스크 입출력 비용이 차지하는 비중이 크므로, 디클러스터링 기법을 적용할 경우 질의 처리 성능의 향상을 기대할 수 있다. 따라서, 이 논문에서는 디클러스터링을 수행한 다중 디스크 구조에서 다중 공간 질의를 처리하는 것을 전제로 한다. 디클러스터링 기법으로는 CMD 기법[3]을 사용하였다. CMD 기법은 고정 그리드 색인 상의 셀에 X, Y 좌표를 부여한 후, 각 좌표를 더한 값을 디스크 개수로 나눈 나머지를 이용해서 디스크에 배치하는 방법으로, 디스크

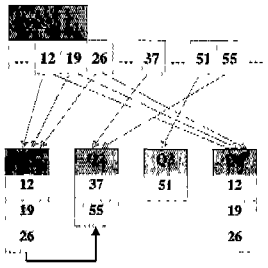
개수가 작을 때 좋은 성능을 발휘하며, 일반적으로 안정된 성능을 내는 장점이 있다[7, 8].

3.3 질의 간 임의 탐색

그림 3은 4개의 디스크에 CMD 디클러스터링 기법을 사용해 공간 데이타를 분산 배치한 환경에서의 질의 처리 예시를 보여주고 있다. 각 질의들은 4장에서 설명한 서버의 질의 처리용 큐(queue)에 저장된 후 수행된다고 가정한다.



(a) 수행되는 영역 질의들



(b) 질의에서 필요한 디스크1의 셀

그림 3 질의 간 임의 탐색의 예시

그림 3 (a)와 같이 질의 처리 큐에 들어온 질의를 처리하기 위해, 디스크 1에서는 먼저 질의 Q1을 처리하기 위해 12, 19, 26번 셀을 읽어 들이게 된다. 그런데 질의 Q2를 처리하려면 37, 55번 셀을 읽어야 하므로 디스크 임의 탐색이 발생하게 되고, 질의 Q3과 Q4를 처리할 때도 역시 디스크 임의 탐색이 발생한다. 이것은 일반적인 다중 프로세스(Multi-process) 운영체제하에서 입출력시스템에서 빈번하게 발생한다. 이러한 현상은 다중 CPU/다중 디스크 기반 구조에서 디클러스터링 기법을 도입해 공간 데이타를 분산시킨 후 질의를 처리한다 해도, 다수의 질의를 처리할 때 각 질의가 연관성 없이 지역 디스크에 대해서 동시에 수행되므로 질의 처리 중에 디스크 임의 탐색이 발생하게 된다. 이 논문에서는 다중 공간 질의 처리 중에 발생하는 임의 탐색을 '질의 간 임의 탐색(inter-query random seek : IRS)'이라고 하고, 다음과 같이 정의한다.

[정의 3] 질의 간 임의 탐색(Inter-query Random Seek : IRS)

−RQ (where RQ MSQ) 처리시 Local 디스크내에서 발생하는 임의 탐색

일반적으로 디스크 입출력 시간에서 디스크 탐색 시간이 가장 큰 비중을 차지하므로, 질의 간 임의 탐색이 발생하면 디스크 입출력 시간이 증가하여 질의 처리 속도가 저하된다. 또한 그림 3의 질의 Q1과 Q4처럼 같은 지역(12, 19, 26번 셀) 또는 유사한 지역을 처리해야 하는 경우, 해당 지역에 대한 디스크 입출력이 중복되고 처리 순서가 연속적이지 않아 디스크 캐쉬의 이용률이 떨어지므로 질의 처리가 더욱 늦어지는 문제점이 있다.

또한 질의 간 임의 탐색은 질의 처리 도중에 발생하는 문제점이므로, 질의 처리 단계 이전의 공간 데이타 배치 단계에서 해결하기 어려운 문제점이 있다. 예를 들어 다중 공간 질의를 구성하는 N개의 질의를 N개의 디스크가 부착된 N개의 CPU에 할당한 후 병렬 처리한다고 가정하자. 즉 디스크 D_1을 가진 CPU C_1이 질의 Q_1을 담당하고, 디스크 D_2를 가진 CPU C_2가 질의 Q_2를 담당하는 형태이다. 질의를 처리할 때 CPU 소요 시간을 T_{CPU}, 디스크 입출력 시간을 T_{DISK}라고 하자. 그러면 단일 디스크 구조에서 소요되는 질의 1개의 처리 시간은 $(T_{CPU}+T_{Disk})$이고, 디클러스터링 기법을 적용한 다중 디스크 구조에서 소요되는 질의 1개의 처리 시간은 메시지 전송 비용을 고려하지 않았을 때 $(T_{CPU}+T_{Disk}/N)$으로 볼 수 있다. 즉 디스크 입출력 시간이 1/N으로 줄어드는 효과를 기대할 수 있다. 그러나 처리해야 할 질의 N개가 동시에 들어오는 다중 공간 질의의 경우, 먼저 Q_1을 처리하기 위해서 C_1이 C_2~C_n에게 공간 데이타를 요청하면, C_2~C_n은 자신의 지역 디스크 D_2~D_n에서 Q_1을 처리하는데 필요한 공간 데이타를 읽어오게 된다. C_2~C_n은 자신이 맡은 질의 Q_2~Q_n을 처리하는데 필요한 데이타는 Q_1을 처리한 후 읽어 올 수 있다. 즉 N개의 질의를 처리할 때의 CPU 시간은 병렬로 수행되므로 T_{CPU}만큼 소요되지만, 지역 디스크 D_2~D_n에 저장된 데이타는 물리적 디스크에서 연산에 필요한 객체단위로 색인수준에서 seek후 접근하므로 임의 접근해야 한다. 즉 순차적인 접근이 발생하게 된다. 그러므로, 입출력 시간은 $(T_{Disk}/N)\times N$이 되어, 질의 N개를

처리하는데 소요되는 전체 시간은 ($T_{CPU}+T_{Disk}$)가 된다. 즉 다수의 영역 질의를 순차적으로 처리하는 경우는 디클러스터링의 효과를 볼 수 있지만, 다중 공간 질의와 같이 다수의 영역 질의를 동시에 처리해야 할 때는 디클러스터링의 효과를 기대하기 어려우므로, 다른 방법으로 디스크 입출력 시간을 줄일 필요가 있다.

4. 다중 공간 질의의 처리 기법

다중 공간 질의를 처리하는 도중 발생하는 문제점인 질의 간 임의 탐색은 일반적인 영역 질의의 처리 기법으로 해결하기에는 부적합하다. 질의 간 임의 탐색으로 발생하는 문제점으로 디스크 탐색 시간의 증가와 공간 데이타에 대한 중복 접근을 들 수 있는데, 이것은 다중 공간 질의를 구성하는 영역 질의 간의 공간 및 시간적 관련성을 고려하지 못한 데에서 비롯된다고 볼 수 있다. 따라서 이 논문에서는 다중 공간 질의를 구성하는 질의들에 대해 시간 및 공간적 관련성을 부여한 후, 관련성이 높은 질의들을 먼저 처리함으로써 공간 데이타에 대한 중복 접근을 줄이고 디스크 탐색 시간을 줄이고자 한다.

이를 위해 우선 4.1절에서 다중 공간 질의를 위한 질의 처리 모델을 기술하고, 질의 간 임의 탐색을 해결하기 위해 필요한 개념인 질의 간 위치 관련성과 질의 대기 시간을 각각 4.2절과 4.3절에서 설명한다. 그리고 이들을 이용해서 다중 공간 질의를 구성하는 질의들 간의 동적 스케줄링을 수행하는 알고리즘을 4.4절에서 기술하고, 예시를 통해 여기서 제시한 질의 스케줄링 알고리즘이 질의 간 임의 탐색을 해결할 수 있음을 보인다.

4.1 다중 공간 질의의 처리 모델

이 논문에서는 다중 공간 질의 처리를 위해 그림 4와 같은 구조를 가지는 마스터-슬레이브(Master-Slave) 모델을 사용하였다. 마스터-슬레이브 모델은 일반적으로 사용되는 병렬 처리 모델으로, 마스터(master)는 주로 작업 할당 및 결과 수집, 부하 균등화(load-balancing) 등을 담당하고 슬레이브(slave)는 실제 작업을 수행하는 역할을 담당하게 된다.

이 논문에서 마스터는 클라이언트로부터 들어온 영역 질의들을 질의 처리 큐에 넣은 후 질의 스케줄러를 통해서 질의를 분배하는 역할을 수행한다. 실제 질의를 처리하는 것을 슬레이브로써, 디클러스터된 데이타에 직접 접근할 수 있는 CPU들로 구성되어 있다. 슬레이브들은 마스터에서 넘겨 받은 질의에 대해 여과 및 정제 작업을 수행하여 질의 결과를 생성한 후, 이를 마스터에게 다시 전송한다.

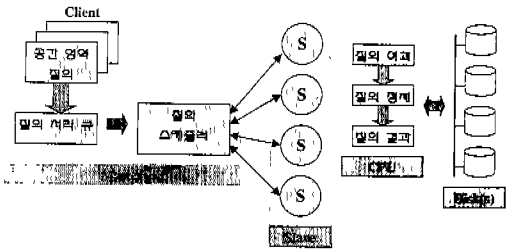


그림 4 다중 공간 질의 처리 모델

그림 4와 같은 질의 처리 모델에서, 클라이언트로부터 들어온 질의들은 마스터 내부의 질의 처리 큐(queue)에 저장된 후 다중 공간 질의로 구성된다. 질의 처리 큐는 질의 스케줄링을 위한 영역 질의의 저장 및 대기 장소로 사용되는데, 다음과 같이 정의된다.

[정의 4] 질의 처리 큐(Query Processing Queue)
- 일정 시간 간격 이내에 들어온 2개 이상의 질의들을 처리하기 위한 자료구조

질의 처리 큐는 우선 순위 큐(priority queue)로써, 큐 안의 요소(item)들 중 가장 높은 우선 순위를 가진 요소를 먼저 출력한다. 일정 시간 간격 이내에 질의들이 올 경우 질의 처리 큐에 입력하게 되어 있고, 클라이언트에서 들어온 질의들의 시간 간격이 크면 질의 처리 큐를 거치지 않고 질의가 들어오는 대로 처리하게 된다. 질의 처리 큐 안의 요소(item)들은 질의 MBR과 질의가 큐에서 대기한 시간, 그리고 질의 우선 순위로 구성된다. 질의 MBR은 질의가 수행될 위치를 보존하고 질의 간 위치 관련성을 계산하기 위함이고, 질의가 큐에서 대기한 시간은 질의 대기 시간을 계산하기 위한 것이다. 질의 우선 순위는 질의 큐 내의 아이템들에게 부여하는 우선 순위이다.

4.2 질의 간 위치 관련성

3.2절에서 제시한 다중 공간 질의의 특징 1에 의하면, 다수의 질의들이 어느 한 곳에 몰리게 되는 질의 집중 지역이 발생할 수 있다. 질의 집중 지역에 대해 수행된 질의들은 서로 공간적으로 겹쳐 있고, 겹쳐진 지역 내에 존재하는 동일한 공간 객체들을 이용할 확률이 크다. 이 논문에서는 이러한 특징을 이용하기 위해, 질의 간 위치 관련성이라는 개념을 도입한다.

'질의 간 위치 관련성(inter-query positional relationship)'은 다중 공간 질의를 구성하는 영역 질의 간의 공간 관련성을 나타내는 기준이 되며, 다음과 같이 정의한다.

[정의 5] 질의 간 위치 관련성(Inter-query Positional

Relationship)

$$PR(Q_i) = \sum_{K=0}^{n-1} Overlap\_Area(Q_{Prev[k]}, Q_i) \quad (1)$$

$PR(Q_i)$=질의 Q_i의 위치 관련성

$Overlap\_Area(Q_1, Q_2)$=질의 Q_1과 Q_2의 겹치는 면적

$Q_{prev}[k]$=위치 관련성을 계산할 때의 기준 질의

수식 1에서 볼 수 있듯이, 질의 간 위치 관련성은 위치 관련성을 측정할 k개의 기준 질의에 대해 n개의 영역 질의들이 겹치는 면적의 합이다. 질의가 겹치지 않거나 두 질의가 단순히 만나는(meet) 경우의 질의 간 위치 관련성은 0으로 정한다. 질의 간 위치 관련성은 질의들이 겹치는 면적과 비례하게 된다. 기준질의는 이전에 수행하였던 질의들 중 가장 최근에 수행한 질의로 선택되어진다.

질의 간 위치 관련성이 크다는 말은 해당 질의가 서로 인접한다는 뜻으로, 같은 지역의 객체를 사용할 확률이 높다는 의미이다. 즉 질의 처리 큐에 대기하고 있는 질의들 가운데에서, 이전에 처리한 k개의 질의에 대한 질의 간 위치 관련성이 높은 질의들의 우선 순위를 높이면 근접한 지역에 대해 집중되는 영역 질의를 같이 처리할 수 있으므로, 디스크 탐색의 중복을 줄이면서 질의 처리 시간을 단축시킬 수 있는 방법이 된다. 그림 5에서 질의 Q1에 대한 다른 질의들의 질의 간 위치 관련성을 수식 1을 이용하여 계산한 것을 나타내었다. Q1은 방금 수행된 질의이고, Q2 Q5는 이후 수행될 질의라고 가정한다.

그림 5를 보면, Q1에 대한 질의 간 위치 관련성이 제일 큰 질의는 Q3임을 알 수 있다. 따라서 Q1 이후에 Q3를 처리하는 것이 유리하다. Q2의 경우, Q1과 겹치지 않으므로 Q1에 대한 질의 간 위치 관련성은 0이 된다.

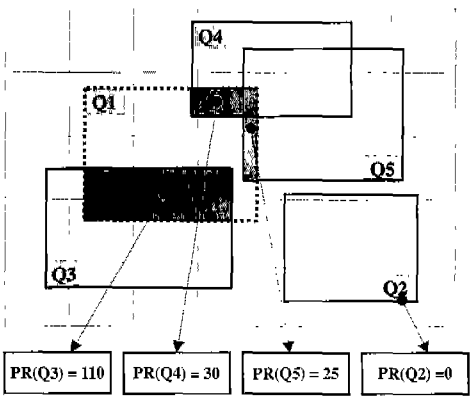


그림 5 질의 간 위치 관련성의 계산 예시

4.3 질의 대기 시간

다중 공간 질의를 처리할 때는 다중 공간 질의의 특징 2에서 알 수 있듯이 모든 클라이언트의 질의를 공평하게 처리해야 할 필요가 있다. 따라서 질의 간의 공간적 측면을 고려하는 요소인 질의 간 위치 관련성 이외에도 질의 사이의 시간적 측면을 고려해야 한다. 예를 들어 그림 6을 보면, 질의 Q2는 서버에 들어온 시간이 Q3~Q5보다 빠른데도 불구하고 Q1과의 질의 간 위치 관련성이 적으므로 질의 처리 순서가 늦어지게 되어, Q2의 처리 시간이 증가하는 것을 알 수 있다. 이러한 문제가 심화될 경우, 질의 처리 큐에 질의가 남아 있는데도 불구하고 다른 질의가 들어올 때까지 기다리는 기아(starvation) 현상이 발생하게 되어 질의 처리가 늦어지는 문제가 발생할 수 있다.

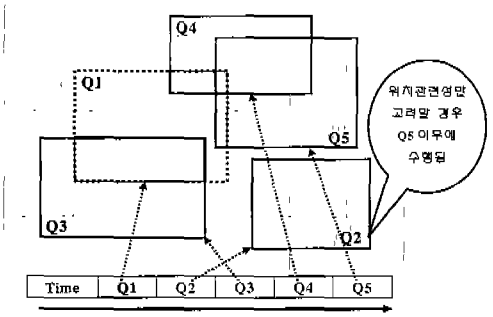


그림 6 시간 관련성을 고려해야 하는 예시

따라서 질의 사이의 공간적 측면만을 고려할 때 발생하는 문제점인, 다른 질의와 공간적으로 떨어져 있어 질의 간 위치 관련성이 낮기 때문에 처리 순서가 계속 늦어지는 질의가 발생하는 것을 방지해야 할 필요가 있다. 이 논문에서는 질의 처리 큐에 존재하는 질의들의 대기 시간을 이용해서 질의 사이의 시간적 측면을 고려한다. 질의 대기 시간(query latency time)은 다음과 같이 정의하였다.

[정의 6] 질의 대기 시간(Query Latency Time)

$$LT(Q_i) = Time_{cur}(Q_i) - Time_{push}(Q_i) \quad (2)$$

$LT(Q_i)$=질의 Q_i에 대한 질의 대기 시간

$Time_{cur}Q_i)$=현재 시간

$Time_{push}(Q_i)$=질의 Q_i가 질의 처리 큐에 입력되는 시간

$LT(Q_i)$는 큐 안에 있는 질의 Q_i의 대기 시간으로써, 큐에 입력된 후 현재까지의 시간이다. 질의의 큐 입력 시간과 출력 시간이 같다면 $LT(Q_i)$=0이다. 이 논문에서는 큐에서 계속 대기하는 질의가 발생하는 문제점을

방지하기 위해, 질의 대기 시간의 최대 임계값 T_{MAX}를 설정해서 질의 대기 시간이 T_{MAX}를 초과하지 못하도록 한다. 즉 임의의 질의 Q_i에 대한 질의 대기 시간이 T_{MAX}에 접근할수록 Q_i가 처리되지 못하고 큐에 오래 머물러 있다는 의미이므로, Q_i의 우선 순위를 높여서 빨리 처리해야 할 필요가 있다. 따라서 질의 대기 시간이 증가할수록 시간적인 측면에서의 질의 처리 우선 순위가 증가하게 된다.

질의 대기 시간을 계산할 때 실제 시간을 측정할 경우 큐 안의 각 질의마다 입력 시간과 출력 시간을 연산해야 하므로, 큐에 대기하고 있는 질의의 개수에 비례하여 연산 비용이 증가하는 문제점이 있다. 그러므로 이 논문에서는 큐에서 대기하고 있는 각 질의에 대해 시간 단위를 설정하고, 큐에서 다른 질의가 빠져 나갈 때마다 남아 있는 질의들의 시간 단위를 늘리는 방법을 사용한다. 이 방법을 사용한 질의 대기 시간은 다음과 같이 정의한다.

[정의 7] 개선된 질의 대기 시간

$$LT'(Q_i) = Count_{pop}(Q_i) \quad (3)$$

$$Count_{pop}(Q_i) = \sum_{Time_{push}(Q_i)}^{T_{MAX}} Query_{pop} \quad (3\text{-}1)$$

$LT'(Q_i)$ = 질의 Q_i에 대한 질의 대기 시간

T_{MAX} = 질의 대기 시간의 임계값

$Count_{pop}(Q_i)$ = Q_i가 들어온 이후 질의 큐에서 출력된 질의의 개수

$Time_{push}(Q_i)$ = 질의 Q_i가 큐에 들어온 시간

수식 3의 질의 대기 시간 $LT'(Q_i)$는 $Count_{pop}(Q_i)$와 동일한 값을 가진다. $Count_{pop}(Q_i)$는 질의 큐에서 대기하고 있는 질의 Q_i가 들어온 후 큐에서 출력된 질의의 개수를 계산하는 함수로써, 큐에서 질의가 출력될 때마다 재계산된다. T_{MAX}≒(질의 큐의 길이)라고 가정하면, 질의 Q_i가 큐에 들어와서 대기하는 동안 $Count_{pop}(Q_i)$는 0에서 T_{MAX} 사이의 값을 가지게 된다. 즉 $LT'(Q_i)$는 T_{MAX}를 초과할 수 없으므로, 결국 수식 3은 수식 2와 결과적으로 동일하다. 이 논문에서는 T_{MAX}=(질의 큐의 길이)로 정한다.

4.4 질의 스케줄링 기법

4.2절과 4.3절을 바탕으로, 이 논문에서는 질의 간 임의 탐색을 해결하기 위한 방법으로 동적 질의 스케줄링 기법을 제안한다. 질의 스케줄링은 다음과 같이 정의한다.

[정의 8] 질의 스케줄링(Query Scheduling)

-질의 처리 큐에서 대기하고 있는 질의들의 공간 및 시간적 측면을 고려하여 질의들의 우선순위를 설정하는 작업

디클러스터링은 순차적인 영역 질의를 처리할 때 효과적이고, 수행 시에는 질의 위치를 예측할 수 없다. 따라서 독립적인 다수의 질의가 동시에 수행되기 때문에 발생하는 질의 간 임의 탐색을 해결하기에는 부적합하다. 질의 스케줄링을 수행하면, 질의 큐 내의 영역 질의들의 처리 순서를 재조정해서 유사한 지역에 대한 질의들을 연속적으로 처리할 수 있다. 따라서 질의를 처리할 때 읽는 객체의 중복을 최소화할 수 있고, 디스크 캐쉬의 적중률을 높일 수 있는 장점이 있다.

질의 스케줄링을 수행하려면 질의 처리 큐에 대기하고 있는 질의들의 우선 순위를 결정해야 한다. 이 논문에서는 공간적 우선 순위로써 4.2절의 질의 간 위치 관련성을 사용하고, 시간적 우선 순위로써 4.3절의 질의 대기 시간을 사용해서, 그 둘의 합으로 질의 처리 시의 우선 순위를 계산한다. 그런데 질의 간 위치 관련성과 질의 대기 시간의 측정 단위는 각기 다르고, 측정값의 차이도 다르다. 그리고 공간적 측면과 시간적 측면 중 어디에 비중을 두는지에 따라서 우선 순위가 변경될 수 있기 때문에, 질의 간 위치 관련성과 질의 대기 시간의 계산 결과를 합한 값을 질의 처리 우선 순위로 바로 사용할 수 없다.

따라서 공간적 우선 순위(질의 간 위치 관련성)와 시간적 우선 순위(질의 대기 시간)에 대한 비율 조정이 필요한데, 이 논문에서는 수식 4를 사용해서 시간적 우선 순위를 기준으로 공간적 우선 순위의 비율을 조정한다. 수식 4는 비율 조정 인자 SF와 시간 우선순위의 임계값 T_{MAX}를 이용해서, 질의 간 위치 관련성 $PR(Q_i)$과 질의 대기 시간 $LT'(Qi)$의 비율을 조정한 개선된 공간 관련성 $PR'(Q_i)$를 계산하는 식이다.

[정의 9] 질의 대기 시간과의 비율을 조정한 후의 질의간 위치 관련성

$$PR'(Q_i) = PR(Q_i) \times (T_{MAX} / \max(PE(Q_i))) \times SF \quad (4)$$

$PR'(Q_i)$: 비율 조정 후의 질의 Q_i에 대한 질의 간 위치 관련성

$PR(Q_i)$: 질의 Q_i의 질의 간 위치 관련성

T_{MAX} : 질의 대기 시간의 최대값

SF : 공간과 시간의 비율 조절 인자(상수)

$Max(PR(Q_i))$: $PR(Q_i)$중 최대값

비율 조절 인자 SF는 시간적 우선 순위에 대한 공간적 우선 순위의 비율로 정의하는데, 다음과 같은 값을 가질 수 있다.

1. 시간적 우선 순위=공간적 우선 순위 : SF=1
2. 시간적 우선 순위 > 공간적 우선 순위 : 0.5 < SF < 1

3. 시간적 우선 순위 < 공간적 우선 순위 : 1 < SF < 2

이 논문에서는 SF를 세 번째 경우인 1에서 2사이의 값을 가지는 것으로 설정했는데, 시간적 우선 순위보다 공간적 우선 순위에 비중을 두는 이유는 다음과 같다. 공간적 우선 순위인 질의 간 위치 관련성을 계산하는 목적은 유사한 지역에 위치한 공간 객체들을 연속적으로 접근함으로써 디스크 입출력 비용을 줄이는 데 있다. 그리고 시간적 우선 순위인 질의 대기 시간을 계산하는 목적은 질의 처리 순서가 늦어지는 질의가 발생하여 질의 처리 시간이 너무 길어지는 것을 방지하는데 있다. 따라서 질의 스케줄링의 주 목적인 유사한 질의를 연속적으로 처리하기 위해서는 시간적 우선 순위보다 공간적 우선 순위의 역할이 크다고 할 수 있다.

지금까지 기술한 사항을 모두 고려한 질의 스케줄링 알고리즘은 그림 7과 같다. 질의 간 위치 관련성을 계산할 때 기준 질의가 k개인 경우를 기준으로 작성한 것으로, 시간 및 공간적 우선 순위를 이용하여 다음에 수행할 질의를 선택하는 알고리즘이다.

```
queue_item get_next_query (query_queue[], prev_query[])
    query_queue[] : 질의 queue(질의 MBR 의 집합)
    prev_query[] : 이전에 수행되었던 query 들
    queue_item : 다음에 수행할 질의
    begin procedure
        // 질의 간 위치 관련성-PR(Qi) 계산
        for (queue_item in query_queue[])
        begin
            for (query in prev_query[])
            begin
                area = positional_relationship (query_item, query);
                sum_area[] += area;
            end for
        end for
        // 질의 대기 시간-LT`(Qi) 계산
        temporal_schduling (query in query_queue[]);
        // 시간과 공간의 비율 조정-PR`(Qi) 계산
        adjust_scale (query_time[], sum_area[]);
        set_priority (sum_area[], queue_length[]);
                //최종 우선순위 설정
        queue_item = max_area_query in query_queue[];
        return queue_item;
```

그림 7 질의 스케줄링 알고리즘

이 논문에서 제시한 질의 스케줄링 알고리즘은 질의 큐에 새로 질의가 들어올 때마다 큐 안의 질의들의 우선 순위를 재계산하게 되는 동적(dynamic) 알고리즘이다. 질의 큐의 크기가 클 경우 우선 순위의 재계산에 따르는 CPU 비용이 증가하는 부담이 있지만, CPU 연산 시간과 디스크 입출력 시간의 단위를 비교해 볼 때 CPU의 연산 비용에 비해 디스크 캐쉬의 적중률을 높여서 디스크 입출력 회수를 줄이는 장점이 더 크다.

그림 7의 질의 스케줄링 알고리즘을 이용해서 질의를 선택하고 할당하는 예제는 그림 8과 같다.

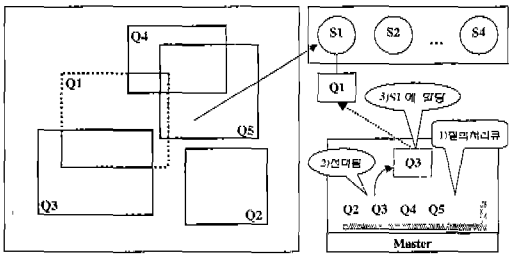


그림 8 질의 스케줄링 수행 예시

질의 큐의 길이는 5라고 가정하자. 마스터는 현재 질의 큐에 있는 질의 Q2 Q5들에 대해서, 질의 Q1에 대한 질의 간 위치 관련성을 수식 1을 이용해서 계산한다(단계 1).

[단계 1] 질의 간 위치 관련성(PR) 계산

PR(Q2)=0, PR(Q3)=110, PR(Q4)=30, PR(Q5)=25 [결과 1]

그리고 질의 대기 시간을 수식 3을 이용해서 계산한다(단계 2). 이 때 Q_i=1, T_{MAX}=5라고 가정한다.

[단계 2] 질의 대기 시간(LT) 계산

LT`(Q2)=3, LT`(Q3)=2, LT`(Q4)=1, LT`(Q5)=0 [결과 2]

질의 스케줄링을 위해서 결과 1과 결과 2 사이의 조정이 필요하므로, 수식 4를 사용해서 시간적 우선 순위에 대해 조정된 공간적 우선 순위 값을 구한다(단계 3). 비율 조정 인자 SF는 1.5로 가정한다.

[단계 3] 개선된 질의 간 위치 관련성(PR`) 계산

PR`(Q2)=0 * (5/110) * 1.5=0

PR`(Q3)=110 * (5/110) * 1.5=7.5

PR`(Q4)=30 * (5/110) * 1.5=2.04

PR`(Q5)=25 * (5/110) * 1.5=1.70 [결과 3]

시간적 우선 순위(결과 2)와 공간적 우선 순위(결과 3)를 이용해서 최종적인 질의 우선 순위를 결정한다(단계 4).

[단계 4] 질의 우선 순위 결정

Q2=0+3=3

Q3=7.5+2=9.5 (우선순위가 가장 높음)

Q4=2.04+1=3.04

Q5=1.70+0=1.70 [결과 4]

마스터는 결과 4와 같이 Q1 이후에 처리할 질의로 Q3을 선택한다. 그리고 그림 9처럼, 질의 Q3을 선택한 후 질의 처리 이력을 고려해서 Q3을 슬레이브 1에게 전송하게 된다.

다중 공간 질의의 특징을 이용해서 질의들의 처리 순서를 결정하는 동적 질의 스케줄링을 수행하게 되면, 유사한 지역에 대한 질의들을 연속적으로 처리할 수 있게 된다. 즉 질의 스케줄링은 다중 공간 질의 처리 시에 발생하는 질의 간 임의 탐색을 질의 간의 관련성을 이용해 줄임으로써, 디스크 캐쉬의 적중률을 향상시키고 디스크 임의 탐색을 감소시킬 수 있는 장점을 가지고 있다.

5. 구현 및 성능 평가

이 장에서는 다양한 질의 처리 환경 하에서 질의 스케줄링을 수행할 때의 질의 처리 시간과 캐쉬 적중률을 조사함으로써, 질의 스케줄링 알고리즘의 효과를 알아본다. 5.1절에서 실험 및 구현을 위한 환경을 설명하고, 5.2절에서는 다양한 질의 크기와 질의 지역의 특성에 따른 성능 평가 결과를 살펴 본다.

5.1 실험 환경

이 논문의 구현을 위해서 다중 디스크/다중 CPU 구조를 가진 Parsytec사의 CC16 병렬 컴퓨터를 사용하였다. CC16은 4개의 클러스터(cluster)로 이루어져 있는데, 각 클러스터 내부에는 4개의 CPU와 1개의 디스크, CPU를 고속 네트워크로 연결하는 라우터(router)가 있다. CPU는 비공유 메모리를 가진 PowerPC 604 CPU이며, 디스크에 직접 접근 가능한 CPU는 입출력 노드(I/O node)의 역할을, 나머지 CPU는 연산 노드(processing node)의 역할을 담당한다. CPU간의 통신은 메시지 전달(message passing) 방식을 사용하게 된다. CC16에는 전체 CPU에서 접근 가능한 전역 공유 디스크가 있는데, 이 디스크는 엔트리 노드(entry node)라는 하나의 CPU가 관리하고 있다. 엔트리 노드는 128MB의 메모리를 가지고 있으며 나머지 CPU들은 64MB의 메모리를 가지고 있다. CC16은 엔트리 노드에서 병렬 프로그램을 수행하면 시스템 내부에서 각 연산 노드로 이를 전파해서 수행하는 구조를 가지고 있으며, 개념적인 구조는 그림 9와 같다.

프로그램은 CC16에서 동작하는 병렬 프로그래밍 라이브러리인 EPX 라이브러리와 C 언어를 사용하여 작성하였다. EPX 라이브러리는 병렬 디스크 입출력을 지원하며, 표준 병렬 프로그래밍 라이브러리인 MPI(Message Passing Interface)와 유사한 기능을 가지고 있다. 성능 평가에 사용한 공간 데이타는 Sequoia 2000 벤치마크 데이타[13] 중 37개 계층으로 구성된 다각형 데이타로써, 원래 데이타를 변형한 3~100개 사이의 점으로 표현되는 60,000여 개의 다각형 객체들로 이루어져 있다. 원래 데이타 중 섬(island) 계층을 제외한 데이타의 대략적인 외형은 그림 10과 같다.

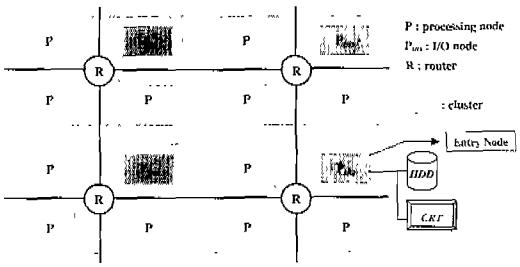


그림 9 CC16의 구조

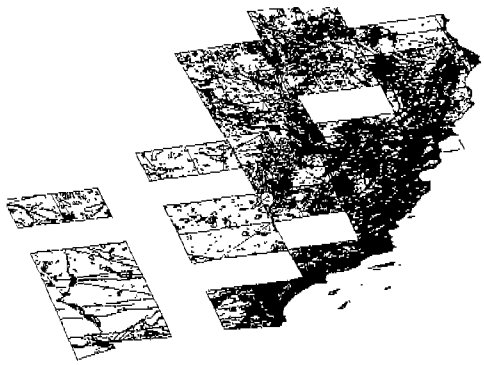


그림 10 벤치마크 데이타의 외형

5.2 전체 수행 시간 비교

여기서는 전체적인 질의 처리 시간을 비교함으로써 질의 스케줄링 알고리즘의 적용 유무에 따른 성능 차이를 비교한다. 다중 공간 질의의 처리 시간은 CPU 처리 시간과 디스크 입출력 시간, 메시지 전송 시간으로 크게 분류할 수 있는데, 질의 스케줄링 알고리즘의 적용 여부에 따른 시간을 비교하고 그 차이를 분석한다.

성능 평가를 위해서 영역 질의의 크기를 4가지로 분류하고, 전체 공간 데이타 영역에 대해 질의 크기별로 각각 200개의 질의를 생성해서 수행하였다. 질의 크기와 질의 처리에 영향을 줄 수 있는 사항은 다음 표와 같이 설정하였다.

표 1 질의 영역의 크기/질의 영향 인자

A) 질의 영역의 크기

	Small	Medium	Large	Vlarge
질의 크기	0.5%~1%	1%~2%	2%~3%	3%~4%

B) 질의 영향인자

질의 큐의 크기	20
캐쉬의 크기	200
비율 조정 인자 SF	1.5
질의 대기 시간의 최대값 T_{MAX}	20

전체 데이타 영역에 대해 수행한 성능 평가 결과 중 전체 처리 시간과 CPU 처리 시간, 디스크 입출력 시간, 캐쉬 적중률을 그림 11에 나타내었다. 메시지 전송 시간은 질의 스케줄링 알고리즘 자체의 성능 평가와는 관계없으므로 제외시켰다.

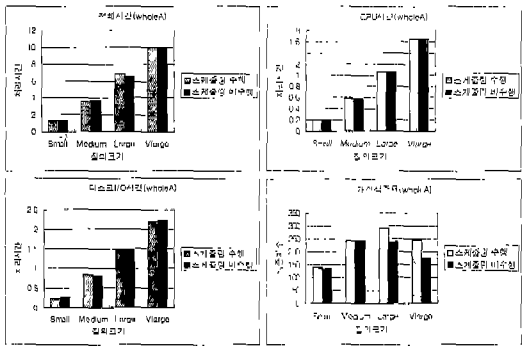


그림 11 전체 수행 시간의 비교

전체 시간을 비교해 보면 질의 스케줄링을 수행했을 때 질의 스케줄링을 수행하지 않았을 때보다 오히려 처리 시간이 조금씩 증가했는데, 이는 CCI6 내부의 메시지 전송 속도가 느려서 질의 스케줄링으로 인한 시간 단축의 효과가 없어졌기 때문이다. CPU 처리 시간의 경우 거의 차이가 없고, 디스크 입출력 시간은 미소하지만 전체적으로 볼 때 1% 단축되었다. 캐쉬 적중률의 경우 스케줄링을 수행했을 때 16% 증가했는데, 질의 크기가 클수록 캐쉬 적중률이 높으며 질의 크기가 작을 경우는 오히려 감소할 수도 있다. 그 이유는 질의가 전체 영역에 대해 수행되었기 때문에 질의 크기가 작을수록 수행된 질의 영역 간의 위치 관련성이 적어서 각 질의에 공통으로 사용되는 객체의 수가 작기 때문이다.

5.3 질의 지역에 따른 성능 비교

여기서는 공간 데이타의 분포 상태에 따른 질의 스케줄링 알고리즘의 성능을 분석한다. 공간 데이타의 분포 상태를 조사하기 위해 고정 그리드 색인의 셀 별 공간 객체 수를 분석하여 객체 밀집 지역과 객체 희박 지역을 선택하였다. 그리고 선택한 질의 지역에 다양한 크기의 질의를 수행하였을 때의 결과를 비교하여 질의 스케줄링 알고리즘의 성능을 분석하였다.

질의 지역은 객체 수가 많고 밀집도가 높은 밀집 지역 2곳과 객체 수가 작고 밀집도가 낮은 희박 지역 2곳, 그리고 임의로 선택한 임의 지역 2곳을 지정하였다. 임의 선택 지역 2곳의 경우 전체 공간 영역을 대상으로 하였다. 밀집 지역 2곳과 희박 지역 2곳의 객체 수는 표 2)와 같다.

표 2 Test data분석

질의 대상 지역	객체 개수	전체 객체 수와의 비율
밀집A (denseA)	26712	45.17%
밀집B (denseB)	19378	32.77%
희박A (sparseA)	3055	5.166%
희박B (sparseB)	6467	10.94%

각 영역에 대해 5.2절에서 정한 크기별로 100개씩 생성한 질의들을 4회 반복 수행한 평균값을 측정하였다. 여기서는 질의 스케줄링 알고리즘의 성능을 직접 반영할 수 있는 기준인 디스크 입출력 시간과 캐쉬 적중률을 측정하였다.

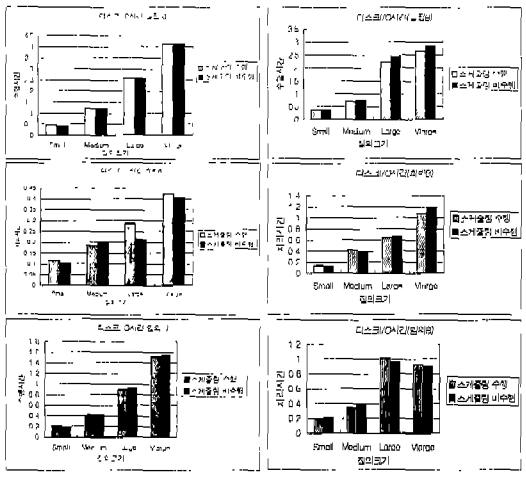


그림 12 질의 지역의 특성에 따른 디스크 입출력 시간 비교

먼저 디스크 입출력 시간을 살펴 보면, 객체의 수가 많고 서로 겹쳐 있는 밀집 지역(밀집A, 밀집B)에서는 대체로 질의 스케줄링을 수행한 경우 6% 정도 감소하는 것을 알 수 있다. 희박 지역(희박A, 희박B)에서는 처리하는 객체의 수가 작고, 디스크 입출력 시간이 크지 않으므로 질의 스케줄링의 장점이 사라지고 오히려 6% 증가했다. 이는 객체의 수가 작고 서로 멀어져 저장되어 있으므로 질의들 간에 공통되는 객체가 없기 때문이라고 할 수 있다. 임의 지역의 경우, 항상 질의 스케줄링의 장점을 얻기는 어렵다. 임의A 지역의 경우 질의 크기가 커질수록 질의 스케줄링으로 인한 이득이 발생하지만, 임의B 지역의 경우 그 반대로 질의 크기가 작을 때 질의 스케줄링의 효과가 크고, 질의 크기가 클 때는 질의 스케줄링의 효과가 줄어드는 것을 알 수 있다. 이는 전체 지역을 대상으로 질의가 수행되었고, 질의가 임의의 위치에 대해 생성되었기 때문이라고 할 수 있다.

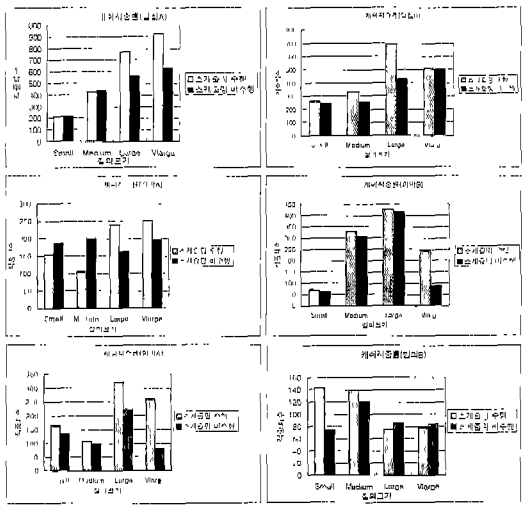


그림 13 질의 지역의 특성에 따른 캐쉬 적중률의 비교

캐쉬 적중률의 경우, 밀집 지역에서는 전체적으로 25%에서 34%까지 증가하였다. 질의 범위가 넓을수록 (Large, Vlarge) 겹치는 객체의 수가 많으므로 증가 폭이 높고(29%~42%), 질의 범위가 좁은 경우(Small, Medium)는 증가 폭이 줄어드는 것(0%~18%)을 알 수 있다. 희박 지역과 임의 지역의 경우 그 차이가 커서 오히려 캐쉬 적중률이 감소하는 경우도 발생하였는데, 이는 해당 지역에 대해 임의로 생성된 영역 질의의 위치 관련성이 작을 수 있고, 캐쉬의 크기 혹은 질의 큐의 크기가 작아서 캐쉬가 제 역할을 다하지 못했기 때문이다. 즉 질의 스케줄링 기법은 객체의 수가 많고 밀집되어 있는 지역일 경우 성능 향상에 도움이 되며, 객체가 근소하게 분포하고 있는 지역의 경우 그 효과가 작다는 것을 알 수 있다.

5.4 캐쉬 적중률의 비교

질의 스케줄링을 수행하였을 때 기대되는 것 중의 하나가 디스크 캐쉬의 적중률 증가이다. 디스크 캐쉬의 적중률이 증가한다는 것은 그만큼 디스크에 대한 접근 회수를 줄일 수 있다는 의미이므로, 질의 처리 시간에서 큰 비중을 차지하는 디스크 입출력 시간을 감소시킬 수 있다. 여기서는 전체 질의 영역에 대해 다양한 크기의 질의를 임의로 생성한 후, 질의 수행 시의 캐쉬 적중률을 조사함으로써 질의 스케줄링의 효용성을 입증하고자 한다.

캐쉬의 경우, 실제 병렬 컴퓨터의 각 지역 디스크 내부에 존재하는 캐쉬를 직접 제어할 수 없기 때문에 메모리 상에 객체 단위의 캐쉬를 생성하여 실험하였다. 캐쉬의 동작 방식은 다음과 같다. 슬레이브가 질의를 처리하는 도중 객체를 디스크에서 읽어 들이게 되면 해당 객체를 메모리에 할당한 캐쉬에 저장한다. 그리고 객체가 필요할 경우 우선 메모리 캐쉬를 탐색해서 원하는 객체가 있는지 조사한 후, 객체가 있으면 디스크 입출력을 수행하지 않고 바로 메모리에 저장된 객체를 사용한다. 객체가 없을 경우, 디스크 입출력을 수행해서 해당 객체를 읽어 오게 된다. 캐쉬의 교체 방식은 일반적으로 많이 사용하는 LRU(Least Recently Used) 교체 방식을 사용하였다. 캐쉬의 크기 및 기타 질의와 관련된 사항은 5.1절의 표1)과 동일하게 설정하였다. 5.2절에서 제시한 각 지역들의 특성에 따른 캐쉬 적중률의 차이는 그림 14와 같다.

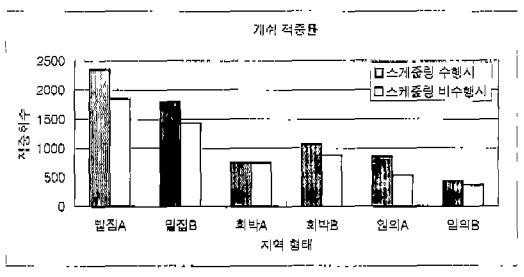


그림 14 지역 형태에 따른 캐쉬 적중률의 비교

그림 14를 보면, 질의 스케줄링을 적용했을 때 밀집 지역에서의 캐쉬 적중률의 차이가 크다는 것을 알 수 있다. 밀집 지역에서 질의 스케줄링을 적용할 경우 캐쉬

적중률은 평균 26%의 향상이 있었다. 이는 밀집 지역에 존재하는 객체의 수가 많고 서로 겹쳐 있는 경우가 많으므로 질의 간 위치 관련성이 커서 질의 스케줄링을 수행한 효과가 크기 때문이다. 반면 희박 지역에서는 밀집 지역보다 작은 11%의 적중률 향상을 보였다. 특히 희박A 지역의 경우 스케줄링을 수행하더라도 적중 회수의 차이가 거의 없는데, 그 이유는 객체의 분포 상태가 분산되어 있고 객체의 수가 작아서 질의 간 위치 관련성이 높아도 실제 캐쉬에 계속해서 남아 있는 객체의 수가 작기 때문이다. 임의 지역의 경우 희박 지역보다 높은 적중률을 보이고 있다.

질의 크기에 따른 캐쉬 적중률을 종합한 결과는 그림 15와 같다.

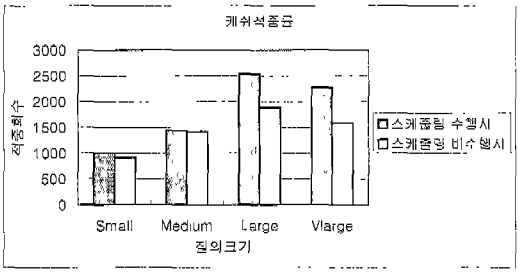


그림 15 질의 크기에 따른 캐쉬 적중률의 비교

우선 질의 지역의 특성에 관계없이 질의 스케줄링을 적용할 경우 캐쉬의 적중률이 증가하는 것을 알 수 있다. 질의 크기가 작은 경우(Small, Medium)에는 적중률의 증가비율이 1%에서 10% 정도로 작다는 것을 알 수 있다. 즉 객체가 밀집된 지역이라도 질의 영역 내에 포함되는 객체의 수가 작기 때문에 질의 스케줄링의 효과가 적음을 알 수 있다. 반면 질의 크기가 크면(Large, Vlarge) 질의 영역 내에 포함되는 객체의 수가 많으므로 질의 스케줄링의 효과가 커서 캐쉬 적중률이 향상되는데, 실험 결과에 따르면 평균 39%의 적중률 증가가 있었다.

그러므로 질의 스케줄링을 수행하게 되면 유사한 질의 지역 내의 객체들에 대해 연속적으로 접근할 수 있기 때문에 디스크 캐쉬의 적중률이 증가하게 되고, 불필요하게 중복되는 디스크 입출력을 방지할 수 있으므로 질의 처리 속도가 개선된다는 것을 알 수 있다.

6. 결론 및 향후 연구

이 논문에서는 디클러스터된 공간 데이타베이스에서의 다중 공간 질의의 특징 및 문제점을 분석한 후 해결책을 제시하였다. 다중 공간 질의는 동시에 수행되는 2개 이상의 공간 영역 질의의 집합으로 인터넷을 기반으로 하는 지도 보기 응용의 주 연산이 된다. 그런데 다중 공간 질의의 처리 시간의 대부분을 차지하는 디스크 입출력 비용을 줄이기 위해서 다중 디스크 구조에서 디클러스터링을 수행해도 질의 간 임의 탐색이 발생하게 된다. 질의 간 임의 탐색은 영역 질의가 동시에 서로 다른 위치에 수행될 때 질의를 처리하는 도중 발생하는 디스크 임의 탐색으로, 디클러스터링을 수행하여도 디스크 탐색 시간이 증가하는 문제점이 있다.

이를 해결하기 위해, 이 논문에서는 질의 간 위치 관련성과 질의 대기 시간 및 질의 처리 이력을 이용하여 질의 간의 우선 순위를 결정할 수 있는 질의 스케줄링 기법을 도입하였다. 질의 스케줄링 알고리즘은 중복되는 객체에 대한 디스크 입출력을 줄이고 디스크 캐쉬의 적중률을 높임으로써 질의 간 임의 탐색을 해결하고 다중 공간 질의의 처리 성능을 향상시킬 수 있다. 알고리즘을 구현한 후 실험을 통한 성능 평가 결과, 캐쉬의 적중률이 최대 34%까지 증가해서 질의 처리 성능이 6% 이상 개선되는 것으로 나타났다.

향후 연구로는 질의 스케줄링 알고리즘을 수행할 때 질의 우선 순위의 재계산 비용을 줄일 수 있는 기법의 연구와, 질의 스케줄링에 필요한 요소들인 시간과 공간적 우선 순위에 대한 정확한 비용 모델의 산출에 대한 연구를 수행할 예정이다. 그리고, 임의접근이 아닌 정형적인 접근형태와 패턴을 가진 공간 질의에 대하여 최적화된 질의 스케줄링을 수행하기 위한 연구를 수행할 것이다.

참 고 문 헌

[1] I. Kamel, C. Faloutsos, Parallel R-Trees, Proc. of ACM SIGMOD Conf., pp.195-204, 1992.

[2] N. Koudas, C. Faloutsos, I. Kamel, Declustering Spatial Databases on a Multi-Computer Architecture, Intl. Conf. on Extending Database Technology(EDBT), pp.592-614, 1996.

[3] J. Li, J. Srivastava, D. Rotem, CMD: A Multidimensional Declustering Method for Parallel Database Systems, Proc. of 18th VLDB Conf., pp.3-14, 1992.

[4] H. V. Jagadish, Linear Clustering of Objects with Multiple Attributes, Proc. of ACM SIGMOD Conf., pp.332-342, 1990.

[5] C. Faloutsos, P. Bhagwat, Declustering using Fractals, 2nd Intl. Conf. on Parallel and Distributed Information System(PDIS), pp.18-25,

1993.
[6] M.H. Kim, S. Pramanik, Optimal File Distribution for Partial Match Retrieval, Proc. of ACM SIGMOD Conf., pp.173-182, 1988.
[7] B. Moon, A. Acharya, J. Saltz, Study of Scalable Declustering Algorithms for Parallel Grid Files, Proc. 10th Intl. Parallel Processing Symp., pp.434-440, 1996.
[8] B. Moon, J. H. Saltz, Scalability Analysis of Declustering Methods for Multidimensional Range Queries, IEEE Trans. on Knowledge and Data Engineering(TKDE), Vol. 10, No. 2, pp.310-327, 1998.
[9] M. Coyle, S. Shekhar, Y. Zhou, Evaluation of Disk Allocation Methods for Parallelizing Spatial Queries on Grid Files, Intl. Conf. on Data Engineering(ICDE), pp.243-252, 1994.
[10] S. Shekhar, S. Ravada, V. Kumar, D. Chubb, G. Turner, Declustering and Load-Balancing Methods for Parallelizing Geographic Information Systems, IEEE Trans. on Knowledge and Data Engineering (TKDE), Vol. 10, pp.632-655, 1998.
[11] T. Brinkhoff, H. P. Kriegel, The Impact of Global Clustering on Spatial Database Systems, Proc. of 20th VLDB Conf., pp.168-179, 1994.
[12] A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, 3rd Edition, pp.543-585, McGraw Hill, 1997.
[13] M. Stonebraker, J. Frew, K. Gardels, J. Meredith, The Sequoia 2000 Benchmark, Proc. of ACM SIGMOD Conf., pp.2-11, 1993.
[14] 박영민, 서영덕, 전봉기, 홍봉희, 다중 공간 질의 처리를 위한 병렬 공간 객체 파일 서버의 설계, 한국정보과학회 '99 봄 학술발표논문집, 제 26권 1호, pp.134-136, 1999.
[15] 박영민, 전봉기, 서영덕, 홍봉희, 디클러스터링된 공간 데이타베이스에서의 다중 공간 질의 처리, 한국정보과학회 '99 가을 학술발표논문집, 제 26권 2호, pp.314-316, 1999.

서 영 덕

1997년 2월 부산대학교 컴퓨터공학과 졸업(공학사). 1999년 2월 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 1999년 3월 ~ 현재 부산대학교 대학원 컴퓨터공학과 박사과정 재학중. 관심분야는 지리정보 시스템, 병렬 지리정보 시스템, 객체 지향 데이타베이스

박 영 민

1998년 2월 부산대학교 컴퓨터공학과 졸업(공학사). 2000년 2월 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2000년 1월 ~ 2000년 12월 인터그래프 코리아 재직. 2001년 1월 ~ 현재 엔텔시스템 재직중. 관심분야는 지리정보 시스템, 분산 미들웨어 시스템, WEB기반 GIS

전 봉 기

1991년 부산대학교 컴퓨터공학과(공학사). 1993년 부산대학교 컴퓨터공학과(공학석사). 1993년 ~ 1998년 한국통신 연구소 전임연구원. 1998년 ~ 현재 부산대학교 컴퓨터공학과 박사과정 재학중. 관심분야는 데이타베이스, 지리정보시스템, 공간객체데이타베이스

홍 봉 희

982년 서울대학교 전자계산기공학과졸업(공학사). 1984년 서울대학교 대학원 전자계산기공학과 졸업(공학사). 1988년 서울대학교 대학원 전자계산기공학과졸업(공학박사). 현재 부산대학교 공과대학 컴퓨터공학과 정교수/부산대학교 컴퓨터 및정보통신 연구소 연구원. 관심분야는 병렬공간 DB, 분산 공간 DB, 개방형 GIS