병렬 공간 색인을 위한 검색 기법

Search scheme for parallel spatial index

서영덕*
Young-Duk Seo

요 약 디클러스터링과 병렬 색인 구조는 효과적인 대용량 데이터의 검색을 위한 데이터베이스의 중요한 연구 분야이다. 기존의 연구에서 다중 디스크에서 다양한 방법들을 이용하여 R-tree등의 색인을 위한 다양한 방법들이 제시되었으나, 이에 적절한 검색 기법에 관한 연구는 거의 없었다.

이 연구에서는 기존의 연구에서 제시된 병렬 공간 색인 구조에서 검색 성능을 향상시킬 수 있는 새로운 검색 기법을 제시한다. 제시된 기법은 디스크로부터 동시에 여러 개의 노드를 읽어 들일 수 있는 병렬 디스크의 장점을 최대한 활용한 기법이다. 기존의 한번에 한 노드를 읽는 알고리즘을 개선하여 여러 노드를 동시에 접근하기 위한 최적화된 기법이다. 실험을 통하여 제시된 기법은 기존의 연구에서 제시된 알고리즘들을 검색 기법의 변경을 통하여 최대 40%이상의 성능 향상을 유발함을 입증하였다.

Abstract Declustering and parallel index structures are important research areas to improve a performance of databases. Previous researches proposed several distribution schemes for parallel R-trees, however there is no search schemes to be suitable for the index.

In this paper, we propose schemes to improve the performance of range queries for distribute parallel indexes. The proposed schemes use the features that a parallel disk can read multiple nodes from various disks. The proposed schemes are verified using various implementations and performance evaluations. We propose new schemes which can read multiple nodes from multiple disks in contrast that to the previous schemes which can read a node from disk. The experimental evaluation shows that the proposed schemes give us the performance improvement by 40% from the previous researches.

주요어 : 공간 색인, 병렬 공간 색인, 병렬 색인, 디클러스터링
KeyWords : Spatial Index, Parallel Spatial Index, Parallel Index, Declustering

1. 서 론

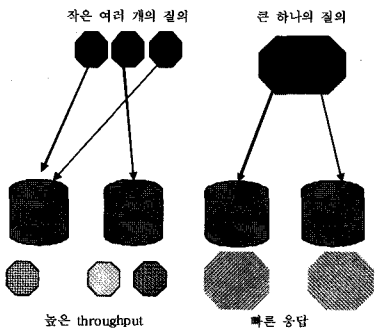
다차원 색인 분야는 오랫동안 다양한 응용의 필요성에 의하여 많은 연구가 있어 왔다. 이러한 응용들은 다차원 데이터를 다룰 수 있는 색인 능력을 필요로 했으며, 많은 색인들이 이를 다루기 위하여 연구되어 왔다. 예를 들어 공간 데이터의 경우 수 많은 공간 객체를 다루기 위하여 많은 색인이 연구되었으며, 대표적으로 R-tree계열의 색인을 들 수 있다.

이러한 색인을 활용하는 지리정보 시스템과 데이터베이스 시스템은 그 시스템의 빠른 응답시간을 목적으로 보다 좋은 성능을 발휘하기 위한 시스템 구축에 목적을 가지고 있다. 빠른 응답시간에 대한 요구에 대한 접근 방법은 다양할 수 있으나, 이 연구에서는 병렬 시스템의 이용에 그 초점을 둔다. 병렬 시스템 중에서 병렬 디스크를 활용하여 데이터를 저장하기 위한 기법으로 디클러스터링과 병렬 색인 기법에서 보다 기존의 연구 성과를 보다 더 향상시킬 수 있는 방

* 경성대학교 컴퓨터과학과 ydseo@star.ks.ac.kr

법을 모색한다.

디클러스터링(Declustering)과 병렬 색인 기법은 <그림 1>에서와 같이 한번의 질의에 대하여 동시에 읽어질 가능성이 있는 데이터를 다른 디스크로 분산 저장하는 방법을 말한다. 여기서 같이 읽혀질 가능성이 있다고 하는 것은 각 객체의 시공간 데이터가 서로 인접하고 있어서 특정 질의가 요청 되었을 때 함께 결과로 선택될 가능성이 높다는 것을 말한다. 즉 인접한 데이터일수록 질의에 대한 결과 집합에 포함될 가능성은 더욱 커지는 점을 이용한 방법이다. 바로 이러한 데이터들을 서로 다른 디스크에 저장하고 같은 I/O 시간에 각각의 디스크에 동시에 접근함으로써 하나의 큰 질의에 대하여는 둘 이상의 디스크로부터 동시 적재를 수행함으로써 빠른 응답시간을 확보하고, 작은 여러 개의 질의에 대하여는 각 디스크에 각 질의를 분산 수행할 수 있도록 함으로써 높은 throughput을 얻고자 고안된 방법이다. 일반적으로 분산 배치가 데이터베이스 자체에 대하여 발생하면 디클러스터링이라 정의되며, 데이터베이스의 색인의 노드에 대하여 분산 배치를 수행하면 병렬 색인이라 정의된다.



<그림 1> 디클러스터링의 목적

기존의 다중 디스크를 이용한 데이터베이스 분산 배치기법 즉, 디클러스터링에 관한 연구는 다양한 환경에서 많은 연구가 있어 왔으며, 특히, 1차원 데이터[8][10]와 2차원 이상의 다차원 데이터/공간 데이터[5][6], 이동체 데이터[9][11]등에서 많은 연구가 있어 왔다. 공간 데이터의 병렬 색인에 관한 연구는 주로 색인에 대한 병렬 색인에 관한 연구가 많은 편이다. 예를 들어 다중 디스크 시스템을 이용한 Parallel R-tree에 관한 연구가 있으며[5], 비 공유 병렬 시스템(shared-nothing parallel system)을 기반으로 하는 연구 등이 있다. [6] 또한 이동체 데이터베이스에서의 병렬 색인에 관한 연구가 있다.[9][11] 이러한 연구들은 데이터를 여러 개의 디스크에 분산 배치하는 디클러스터링기법과 색인의 노드를 병렬 배치하는 병렬 색인에 관한 연구로 나눌 수 있다.

기존의 병렬 색인에 관한 연구들은 색인의 분산 배치 기법에 관한 연구와 색인의 구조에 관한 연구가 주류를 이룬다. 즉, 검색 성능의 향상을 위하여 색인의 노드를 각 병렬 디스크에 분산 배치하는 연구를 많이 했다. 그러나, 병렬 색인은 색인의 병렬 검색 성능이 검색의 기준에 만족하는 한 노드의 자식노드의 개수에 의하여 결정되는 문제점이 있다. 즉, 검색 성능이 디스크의 개수에 비하여 크게 증가하지 않는 문제점이 있다.

또한 기존의 데이터베이스나 데이터 검색 알고리즘에서 색인은 전체 공간에 대한 검색을 피하기 위하여 다양한 목적으로 사용된다. 그러나, 기존의 방법들은 대다수 깊이 우선 정책이나 넓이 우선 정책의 기법등의 노드 단위의 검색 기법을 활용한 기법을 이용한다. 즉, 임의의 한 노드에서 다음 레벨의 노드를 탐색할 때, 단일 디스크를 사용하는 것을 전제로 개발된 알고리즘들을 활용하므로, 한번에 하나의 노드를 디스크로부터 메모리에 적재하는 것을 전제로 알고리즘이 작성되어 있다. 이것은 병렬 디스크를 활용하여 한번에 여러 개의 디스크로부터 동시에 적재하는 알고리즘에 부적절한 기법이다.

이 논문에서는 기존의 연구에서 고려하지 않고 있는 검색 방법에 대한 새로운 방법을 제시한다. 즉, 이 논문에서는 기존에 병렬 색인에서 일반적으로 사용되는 깊이 우선 혹은 넓이 우선 탐색기법의 한계를 제시하고, 이 기법을 개선한 새로운 검색 기법을 제시한다. 이 논문에서는 기존의 공간 데이터 혹은 다차원 데이터의 병렬 색인에 관한 연구들에서 사용되던 일반적인 트리 기반 색인 구조, 특히 R-tree를 이용한 병렬 색인에 대한 검색 기법을 새롭게 제시한다. 연구의 결과 기존의 깊이 우선 정책(DFS)나 넓이 우선 정책(BFS)의 탐색 기법은 병렬 색인에서 그리 좋지 못한 성능을 발휘하는 것으로 나타났으며, 이를 보완하고 높은 병렬화률을 지원하는 새로운 검색 기법은 다른 알고리즘의 변화 없이 성능을 최대 40%까지 향상시킬 수 있었다.

이 논문의 구성은 다음과 같다. 2장에서는 디클러스터링과 이동체의 궤적 색인에 관한 기존의 연구를 기술한다. 3장에서는 병렬 궤적 색인을 기존의 방법으로

검색 할 때의 한계상황에 대한 문제점을 소개한다. 또한 4장에서는 3장에서 제시된 문제점을 해결하기 위한 새로운 검색 기법을 제시하며, 5장에서 제시된 검색 기법의 성능 평가 결과를 제시한다. 마지막으로 6장에서 결론 및 향후 연구에 대하여 기술한다.

2. 관련연구

디클러스터링과 병렬 색인에 관한 다양한 연구 결과들은 주로 페이지 혹은 노드의 분산 배치 기법에 관한 연구이다. 즉, 주어진 데이터베이스에 대하여 색인을 어떠한 방식으로 혹은 어떤 기준으로 서로 다른 디스크에 저장할 것인가에 관한 연구이다.[2][5][7][9][11] 특히, 다차원/공간 데이터베이스에서의 병렬 색인에 관한 연구는 트리 구조의 색인(예:R-tree계열의 색인)을 많이 이용하는데, 다음과 같이 공간 데이터를 대상으로 하는 색인 구조에 관한 연구와 이동체 등의 다른 특징을 가진 색인 구조에 관한 연구로 나누어 볼 수 있다.

다차원 속성 데이터에 관한 최근의 연구로서 [1]를 들 수 있다. [1]에서는 PN-tree를 이용하여 다차원 병렬 색인을 제시하였다. PN-tree(Projected Node tree)는 B+-tree를 이용하여 다차원공간에서의 점 데이터(속성 데이터)를 각 차원 축에 투영(Project)하여 각 축에 대한 여러 개의 B+-tree를 형성시키는 방법을 취한다. 또한 각 트리의 노드는 [5]에서 제시한 proximity를 기반으로 병렬 시스템(다중 프로세서, 다중 디스크) 시스템의 각 디스크에 적재하는 방식을 취하고 있다.

공간 데이터베이스에서의 색인 구조에 관한 연구는 공간 객체간의 인접성인 proximity를 이용한 Parallel R-trees[5]과 비 공유 병렬 시스템(shared nothing parallel system)에서의 페이지 분산 배치 방법을 들 수 있다.[7] 이들 연구 외에 직관적으로 수용 가능한 방법은 다양한 직관적인 방법들(Round robin, random, Hilbert curves등)이 사용되어 왔으며, 제시된 기법과의 성능 평가의 목적으로 주로 사용되었다.

또한 공간 데이터 유사한 형태를 가지나 좀 더 복잡한 특징을 가지는 이동체의 궤적에 관한 색인의 병렬 색인 구조에 관한 연구로서 [9][11]을 들 수 있다. [9]에서는 이동체의 시간적 인접성과 공간적 인접성을 동시에 고려한 노드 분산 정책을 제시하였으며, [11]에서는 이동체의 시간/공간적 근접성을 처리한 후 이를 KT proximity로 표현하는 기법을 적용하였다.

위에서 제시한 기존의 연구 성과에 의한 기법들은 데이터베이스의 검색 성능을 향상시키기 위하여 배치 기법에만 중점을 둔 특징이 있다. 즉, 데이터의 특성에 따라 한 노드의 자식 노드들의 수가 많지 않고, 자식 노드들 전체는 항상 동시에 검색될 것이라는 전제하에서 배치 기법만으로 색인의 성능을 향상시키기 위한 기법들을 제시하였다. 그러나, 3장에서 제시하듯이 검색 기법의 개선 없이 분산 배치만으로는 성능향상에 한계를 가지고 있다.

3. 문제 정의

이 장에서는 기존에 제시된 다양한 병렬 색인 모델들의 검색 성능상에서의 한계점을 도출하고 그 이유에 대한 논의를 한다. 특히, 기존의 연구에서 중요한 이슈가 되지 못한 검색 기법의 개선을 통하여 보다 향상된 성능 향상 효과를 예측할 수 있다.

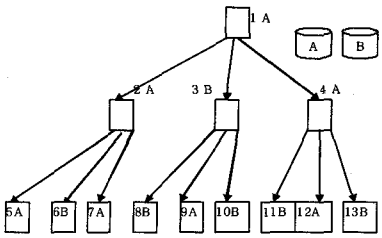
디클러스터링의 최대 속도는 일반적으로 한계 검색 성능(strict optimality)로서 제시된다. 다차원 격자(그리드 파일)를 이용한 디클러스터링 기법에서의 한계 검색 성능은 정의 1과 같이 정의된다.[4] 이 한계 검색 성능은 질의의 수행에 필요한 전체 페이지수를 디스크 수로 나눈 값의 *ceiling*이 전체 검색 횟수가 되며, 이 횟수에 단위 입출력 시간을 곱한 응답 시간이 발생하는 디클러스터링 기법을 한계 성능을 가진 디클러스터링 기법이라 정의하며, 식 1을 임의 질의에 대한 가장 최적의 응답시간이라 정의할 수 있다.

정의 1. 한계검색 성능(strict optimality): 임의의 디클러스터링 기법이 strictly optimal하다는 것은 모든 질의에 대한 검색시 응답 시간이 식 1와 같이 발생하는 것을 의미한다.

$$\left\lceil \frac{\text{전체 검색 페이지수}}{\text{디스크 수}} \right\rceil * \text{single I/O time} \quad \text{--- 식1}$$

식1은 고정 격자 형태의 색인 혹은 디클러스터링된 데이터베이스에서 최적의 검색 성능에 대한 이론적 한계치이다. 그러나, 트리 기반의 다차원 병렬 색인에서 검색 시 최적 검색 횟수를 산출하기 위한 방법은 다차원 격자에서와 같이 단순하지 않다. 그것은 다차원 색인은 검색 시 발생하는 자식 노드의 수에 의하여 최적 검색 횟수내에 검색 가능한 노드의 수가 변하기 때문이다. 즉, 시스템에 설치된 디스크의 수와 질의에

만족하는 자식 노드의 수가 같거나 이의 배수인 경우 식1에 따르지만, 그렇지 못한 경우 디스크를 모두 이용하지 못하는 단점이 있다. 또한 트리 기반의 색인은 여러 레벨에 걸쳐서 형성되어 있기 때문에 동일 레벨을 가정한 식1과는 적절하지 않음을 알 수 있다.



<그림 2> 분산 배치된 트리기반 색인

예를 들어 <그림 2>는 노드의 용량이 3인 병렬 색인이 두 개의 디스크에 분산 배치되었음을 가정한 것이다. 색인의 영역 전체에 대한 영역 검색을 수행할 때, 식 1에 의한 가장 이상적인 디스크 접근은 ceiling of (총 노드 수 / 디스크 수)이므로, ceiling(13/2) = 7이 가장 최적화된 디스크 접근횟수이다.

위에서 기술한 최적 검색 회수는 전통적인 검색 기법에 의하여는 성취되기 힘들다. 그것은 기존의 검색 기법은 부모 노드가 선택될 때 이를 기반으로 검색을 수행하기 때문이다. 즉, 색인이 다차원이라는 특성에 의하여 각 노드의 하위 노드를 검색할 때의 병렬 입출력의 한계가 부모 노드로부터 질의에 만족하는 자식 노드의 개수에 의하여 제약되기 때문이다. 그러므로, 부모 노드로부터 질의에 만족하는 자식노드들의 개수가 전체 디스크 수 M과 같은 경우나 M의 배수인 경우에는 식1의 최적화가 이루어 지지만, 이 외의 경우에는 각 레벨에서 누적되는 여분의 입출력으로 인하여 최적화된 입출력이 발생하지 않는다.

예를 들어 <그림 2>에서 깊이 우선 정책 정책으로 해당 색인을 탐색할 경우 다음과 같은 검색 순서를 얻을 수 있다. 즉 노드 2와 3,4등에서 자식 노드의 수가 디스크의 수보다 많으므로, 자식 노드를 탐색할 경우에는 항상 두 번의 입출력이 발생하게 된다.

1→2,3→5,6→7→8,9→10→4→11,12→13

일반적으로 트리 형식의 병렬 색인에서 노드 단위의 탐색을 수행할 경우, 전체 응답 시간을 결정짓는 요소인 병렬 입출력 횟수는 부모 노드로부터 자식 노드로 내려가는 시점에서의 병렬화률이 결정하며, 최적의 분산 배치가 이루어 진 경우 이 값은 노드의 용적율(fanout)에 의하여 결정된다. 즉, 하나의 노드로부터 자식 노드들을 적재할 때, 동시에 적재 가능한 비율(병렬화률)이 얼마나 높은 지에 의하여 결정된다. 그러므로, 최적의 노드들의 디클러스터링이 최적인 상태를 가정하고, 단말노드까지만 검색한다고 가정할 경우 병렬 색인을 전체 검색할 때의 디스크 입출력 회수는 다음과 같이 결정된다.

$$NAP(R1) = (\text{전체 노드수} - \text{말단 노드수}) * \left\lceil \frac{\text{용적률}}{\text{디스크수}} \right\rceil \quad \text{-----식2}$$

여기서 용적률은 색인의 평균 용적률을 말하며, 전체 노드 수는 색인의 전체 노드 수이다. 식 2에 그림 2에 적용할 경우, 최적의 전체 입출력 횟수는 8회이다. 만일 단말노드로부터 데이터 페이지를 포함하는 구조로 데이터베이스가 형성되어 있다면 전체 디스크 검색 회수는 다음과 같다.

$$NOAP(R1) = \text{전체 노드수} * \left\lceil \frac{\text{용적률}}{\text{디스크수}} \right\rceil \quad \text{-----식3}$$

식 3에서 제시된 최적화된 입출력 회수는 일반적인 색인의 탐색 기법으로는 획득될 수 없다. 이 논문에서는 위 식에 가장 근접한 디스크 입출력 횟수를 구하기 위하여 색인의 탐색 기법으로부터 찾는다.

4. 개선된 검색 기법들

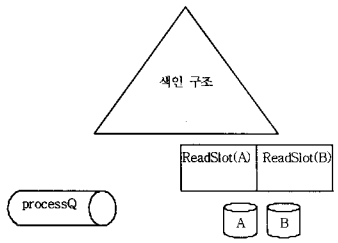
3장에서 기술한 바와 같이 병렬 색인의 비용은 깊이 우선 정책에 의한 탐색기법이나 넓이 우선 기법에 의한 탐색기법 모두 최적화된 성능을 발휘하지 못한다. 그것은 각 탐색기법에서 검색하는 노드의 병렬성의 단위가 레벨 혹은 노드의 자식 노드의 개수로 제한되기 때문이다. 이 장에서는 3장에서 제시된 문제점을 하기 위하여 두 가지의 영역 탐색 기법을 제시한다. 우선 넓이 우선 탐색 기법을 수행하면서 인접노드를 검색하기 위한 제한된 레벨 단위 검색(BFSW)기법과 큐를 이용한 레벨단위 검색(BFSQ)를 제시한다.

4.1 제한된 레벨 단위 검색(BFSW)

제한된 레벨 단위 검색은 기존의 병렬 입출력을 위한 단위가 노드 단위이기 때문에 발생하는 문제점을

해결하여 보다 많은 병렬 디스크 선택 기회를 부여하기 위하여 고안되었다. 즉, 레벨 우선 검색(Breadth First Search)를 기본적으로 수행하되, 하위 노드에 대한 디스크 검색 시 인접 노드에 대하여 일정한 검색 범위 내에서 검색을 수행하는 것이다.

예를 들어, 단일 디스크를 이용할 경우 그림 3의 색인은 기존의 넓이 우선 탐색 기법을 통하여 검색을 수행할 경우 1 → 2,3 → 4 → 5,6 → 7 → 8,9 → 10 → 11,12 → 13의 순서로 디스크의 노드를 적재하게 된다. 이 경우 총 10회의 디스크 접근이 필요하다.



<그림 3> 분산 배치된 트리기반 색인

BFSW의 기본적인 아이디어는 자식 노드의 디스크 탐색 시 저장 큐를 이용하여 큐 내에 있는 병렬 가능한 노드를 찾아서 동시 탐색을 수행하는 것이다. 그림 3에서 각 디스크에 할당된 ReadSlot은 각 디스크마다 하나씩 생성되며, 해당 디스크에 요청될 하나의 노드를 보관하게 된다. processQ는 넓이 우선 탐색의 진행을 위하여 사용된다. 만일 processQ의 첫 노드를 pop하여 질의와 비교하였을 때, 성공하는 노드는 각 노드가 저장된 디스크의 slot에 적재되며, 만일 하나의 노드에 대한 조사에서 모든 디스크의 ReadSlot이 하나 이상의 노드를 가지게 되면 slot에 대한 병렬 입출력을 수행한다. 또한 이때 해당 디스크에 이미 읽을 노드가 있다면 proceessQ에 저장한다. 해당 노드에 대한 질의 연산이 모두 끝이나서 다음 읽기를 위하여서 각 디스크에 할당된 ReadSlot를 검사해서 만일 빈slot 이 발생하면 processQ내의 특정한 범위(Window)까지를 검색하여, 범위내에 해당하는 entry를 가진 노드가 존재를 찾아서 ReadSlot에 적재한다. 이후, 모든 ReadSlot에 요청 노드가 있거나, Window내에 해당되는 노드가 없을 때, 병렬 입출력을 수행한다. 여기서 processQ의 탐색 범위를 제한한 것은 색인의 크기가 클 때, 모든 하위 형제노드를 탐색하는 비용을 없애기 위한 것이며, 실험치에 의하여 그 크기를 결정한다.

이렇게 검색을 수행함으로써, 이 알고리즘은 한번의 질의시에 가능하면 많은 디스크를 동시 탐색할 수 있으며, 성능의 향상을 가져올 수 있다. 또한, 최악의 경우에도 기존의 깊이 우선 탐색 기법과 동일한 검색 성능을 보장하게 된다. 제시하는 BFSW방법은 아래의 알고리즘을 이용하여 1 → 2,3 → 4 → 5,6 → 7 → 8,9 → 10 → 11,12 → 13등의 방법으로 진행할 수 있다. 즉, 총 8회의 디스크 접근으로 모든 노드를 검색할 수 있다.

```
Algorithm BFSW
Begin
 Node node = GetRoot();
 Queue queue;
 Queue readQueueOfArray
 Insert root to queue
 While(queue not empty)
 Begin
       Pop node from queue
       Test node for query
       for all qualified entry of node
       Begin
          If ReadSlot of node is not empty assign
          entry to the slot
          Else enque the node to readQueueOfArray
       if readQueueOfArray has non-assigned disk
       Begin
          for node in queue within SEARCH_WINDOW
          Begin
            Find node having entry in non-assigned disk
          End
          Pop the node
          assign entry to readQueueOfArray
       End
       ParallelRead
       Insert readed nodes to queue
   End
 End
```

(알고리즘 1) 제한된 레벨단위 검색(BFSW)

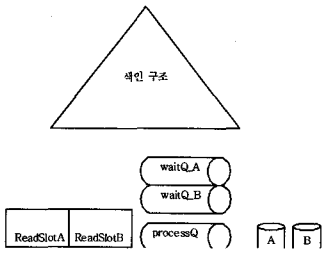
(알고리즘 1)은 이 논문에서 제시하고 있는 제한된 레벨 단위 검색 기법인 BFSW를 수행하기 위한 알고리즘이다. 이 알고리즘에서는 SEARCH_WINDOW내의 질의 큐로부터 디스크에 적재 가능한 노드 집합을 찾아서 병렬 적재한다.

4.2 큐를 이용한 레벨단위 검색(BFST)

전 절에서 제시한 BFSW는 각 디스크 별로 하나의 slot을 할당하고 병렬 검색 시 일정한 범위 내에서 최대한의 노드를 동시 검색하기 위한 알고리즘이다. 이

알고리즘은 각 디스크의 slot에 요청이 없는 경우 이를 processQ에서 찾아서 할당한다. 이 알고리즘은 매번 processQ를 확인해야 하는 단점이 있다.

이 절에서 제시하는 큐를 이용한 레벨단위 검색(BFST)는 여러 개의 queue를 이용하여 각 디스크에 할당된 노드 ID의 집합으로부터 전체 디스크가 읽혀질 수 있도록 고안된 알고리즘이다. 즉, (알고리즘 2)에서 제시된 Queue를 이용한 레벨단위 검색 기법은 기존의 넓이 우선 탐색기법보다 디스크 개수만큼의 queue를 더 사용한다. 디스크 개수만큼의 queue는 질의의 결과에는 만족하나 병렬 탐색대상에는 포함되지 않는 entry의 집합이 저장된다.



<그림 4> 큐를 이용한 레벨단위 검색을 위한 시스템 구조

예를 들어 그림 4을 BFST탐색기법을 통하여 탐색할 때, 다음과 같이 2개의 임시 queue가 필요하며, 하나의 디스크 entry list, 그리고 탐색 진행을 위한 queue가 필요하다.

질의 q에 대한 영역 질의 수행을 예를 들어 설명하면 다음과 같다. 프로그램의 초기에 processQ에는 루트 노드인 1번 노드가 저장되며, 이후 pop된다. 이후, pop된 노드로부터 각 entry들을 질의 q에 의하여 조사하여 검색에 성공한 자식 노드에 대한 참조(pointer)중 디스크 슬롯이 비어 있는 참조들은 빈 디스크 슬롯에 저장되며, 그렇지 못한 참조들은 각 디스크의 wait queue에 저장된다. Wait queue에 저장된 참조들은 다음 차례에 비교없이 디스크 슬롯에 저장되며, readSlot이 다 차면 readSlot내에 있는 하위 노드에 있는 참조들은 병렬로 메모리에 적재된다. 적재된 노드들은 processQ의 맨 뒤에 위치하게 된다. 즉, 알고리즘 4는 하위 노드에 대한 읽기 작업을 검색된 순서에 의하지 않고 병렬화를 우선시 하여 읽어 들임으로써, 병렬 입출력의 효과를 최대한으로 끌어 올리는 효과를 가지게 된다.

```
BFST (Query q,Tree t)
Begin
 Queue queue;
 Queue NodeList[DiskNum];
 List ParallelReadQueue[DiskNum];
 Read root(t) to queue;
 While queue nonempty
 Begin
     Pop node from queue
     If node is leaf node Do for leaf node(node)
     For DiskNum in NodeList[DiskNum]
     Begin
        If NodeList[DiskNum] is Not empty
        Begin
          Move NodeList[DiskNum] to
          ParallelReadQueue[DiskNum];
        End
     End
     Query_test(q,node)
     For all entry in node
       If ParallelReadQueue[DiskOfEntry] is empty then
          Insert entry to
          ParallelReadQueue[DiskOfEntry]
       Else
          Add entry to NodeList[DiskNum];
       End If
     ParallelRead(ParallelReadQueue);
     Add Readed nodes to the end of queue
  End
End
```

(알고리즘 2) 개선된 넓이 우선 탐색 기법

5장의 성능평가에서 나타나듯이 (알고리즘 2)는 기존의 디클러스터링 정책의 성능을 40%이상 향상시키는 효과를 가지고 있다. 병렬 색인을 구성하기 위한 병렬화 기법은 한 노드의 분산 배치 여부에 의하여 성능이 결정되지만, (알고리즘 2)에 의한 기법은 전체 노드를 대상으로 병렬 탐색의 대상으로 하므로, 최적화된 병렬 색인에 가장 가까운 성능을 발휘할 수 있다.

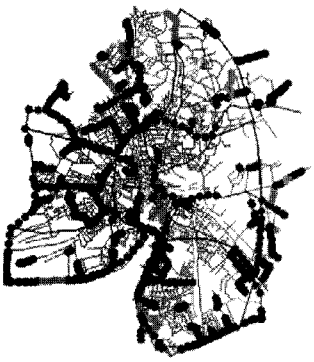
5. 구현 및 성능평가

이 장에서는 지금까지 제시된 디클러스터링 방법과 제시된 검색 기법에 대한 성능 평가의 결과를 소개한다. 또한, 제시된 비용 모델/최적 디스크 입출력 회수와 실제 실험결과와의 차이에 대한 논의를 수행한다.

5.1 시스템 구조/실험 환경

제안된 방법의 성능 평가를 수행하기 위하여 이 논문에서는 2차원 궤적 집합에 대한 성능평가를 수행하

였다. 모든 궤적 집합은 [3]에서 제공하는 도로 데이터 생성기(Traffic data generator)를 이용하여 Oldenburg의 네트웍 데이터[3]를 기반으로 총 25,044개의 궤적 집합<그림 5>을 이용하여 영역질의와 궤적 질의에 대한 성능을 평가 하였다. 또한 비용 모델의 검증에 사용된 데이터는 동일한 실험 조건으로부터 총 100,000개의 데이터를 생성하여 수행하였다.



<그림 5> 실험 데이터의 visualization

많은 요소들이 성능평가에 영향을 줄 수 있다. 그러나, 이 논문에서는 다음과 같은 대표적인 요소들을 고려하여 성능 평가를 수행하였다.

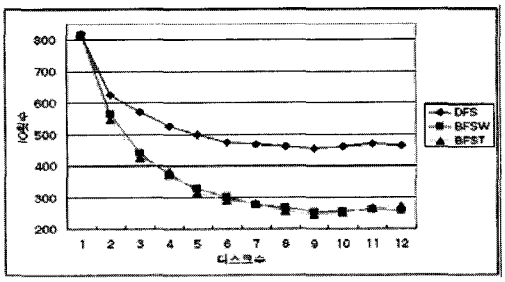
이름	값	비고
NB	0~1024	버퍼수
SB	1K~16K	페이지의 크기
D	1~12	디스크 수
SQ	1%, 5%, 10%	질의의 크기

또한 질의는 총 500개를 임의로 random하게 생성하였다. 또한 이 실험에서 성능 비교의 대상이 되는 기법은 기본적으로 많이 사용되는 round robbin 기법과 random assign, 기존의 공간 데이터에 대하여 적용되는 PI(Proximity index)기법과 [9]에서 제시하고 있는 시공간 인접성을 이용한 proximity기법(STP)간의 성능을 비교하였다.

5.2 검색 기법에 따른 성능 평가

<그림 6>은STP기법으로 분산 배치된 병렬 색인에 대한 1%의 영역질의에 대하여 기존의 깊이 우선 정책과 4장에서 제시한 BFSW와 BFST와의 성능을 비교한 그래프이다. 즉, 다음과 같은 설정을 사용하였다.

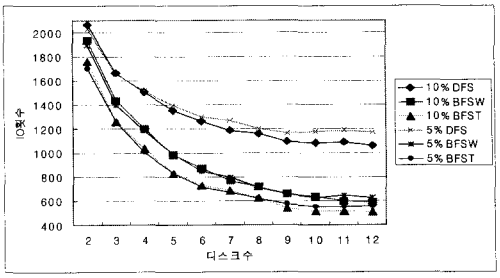
* NB = 100
* SB = 1024
* D = 1, ..., 12
* SQ = 1%



<그림 6> 검색 기법에 따른 성능 평가

<그림 6>에서 DFS는 전통적인 깊이 우선 정책에 의한 검색 기법에 의한 성능을 나타낸다. <그림 6>의 그래프에서 나타나듯이 레벨 단위 검색은 약 40%이상의 성능 향상을 가져온다. 이것은 디스크 읽기의 병렬화에 기인한 것으로 단순히 노드 단위에서의 병렬화에 대한 성과를 뛰어 넘는 것으로 볼 수 있다. 또한 BFSW와 BFST는 성능의 차이가 그리 크지 않음을 알 수 있다. 이것은 대다수의 검색 대상 노드가 BFSW의 검색 윈도우 내에서 찾아져서 전체적으로 검색하지 않아도 일정한 범위내의 노드에 각 페이지들이 골고루 산재함을 의미한다.

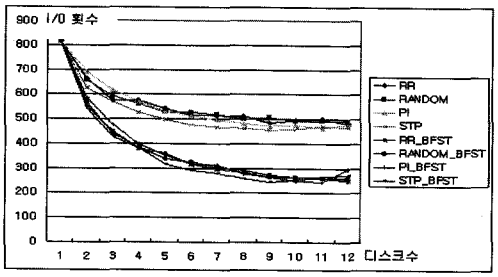
또한 5% 질의와 10%질의에 대하여도 그림 7와 같은 성능을 나타낸다.



<그림 7> 검색 기법에 따른 성능 비교(5%와 10%질의)

또한 <그림 8>은 기존에 제시된 검색기법과 본 논문에서 제시된 검색기법들에 대하여 큐를 이용한 레벨

단위의 검색 기법을 적용하였을 때의 성능의 변화를 나타냈다. 그림에서 RR기법은 기존의 Round-Robin기법을 이용한 방법이며, RANDOM은 임의로 디스크에 할당한 방법을 이용하여 수행하였다. 또한 PI는 [5]에서 제시한 Proximity index기법을 활용한 방법이다. 그림에서 나타나듯이 기존의 기법에 의한 성능의 한계를 10개 이상으로 낮추었으며, 검색 기법의 변화에 의하여 실제 성능상에서 40%이상의 성능 향상 효과를 가져왔다.



<그림 8> 검색 기법에 따른 검색 성능의 비교

6. 결론 및 향후 연구

디클러스터링과 병렬 색인 구조는 효과적인 대용량 데이터의 검색을 위한 데이터베이스의 중요한 연구 분야이다. 기존의 연구에서 다중 디스크에서 직관적인 방법들을 이용한 R-tree등의 색인을 위한 다양한 방법들이 제시되었으나, 이에 적절한 검색 기법에 관한 연구는 거의 없었다. 이 연구에서는 기존의 연구에서 제시된 병렬 색인 구조에서 검색 성능을 향상시킬 수 있는 새로운 검색 기법을 제시하였다.

이 논문에서는 기존의 검색 기법인 깊이 우선 혹은 넓이 우선 정책 등의 한계를 지적하고 이에 대한 성능 향상을 위한 대책으로 제한된 레벨단위 검색기법과 queue를 이용한 확장된 넓이 우선 검색 정책을 제시하였다. 제시된 기법은 디클러스터링 기법의 성능을 보다 향상시켰으며, 최적화된 성능 한계에 근접하는 성능을 발휘하였다. 제시된 기존의 잘 알려진 검색 기법에 보다 최대 40%이상의 성능 향상 효과를 가지고 있음을 실험을 통하여 제시하였다.

향후 병렬 색인에 대한 질의 수행 시 병렬 색인에서의 비용모델과 최소 비용에 대한 연구를 수행할 것이다.

참고문헌

[1] M. H. Ali, Amani A. Saad, Mohamed A. Ismail: "The PN-Tree: A Parallel and Distributed Multidimensional Index," Distributed and Parallel Databases, vol.17,no.2, pp. 111-133, 2005.

[2] Sanjiv Behl , Rakesh M. Verma, "Efficient Declustering Techniques for Temporal Access Structures", Proceedings of the 12th Australasian conference on Database technologies, pp. 91-98, 2001

[3] T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects", GeoInformatica, Vol. 6, No. 2, pp. 153-162, 2002.

[4] H. C. Du and J.S. Sobolewski, "Disk allocation for cartesian product files on multiple-disk systems", ACM Transactions on Database Systems , Vol. 7, No.1, pp. 182-101, 1982.

[5] Ibrahim Kamel , Christos Faloustsos , "Parallel R-trees", In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 195-204, 1992

[6] Nick Koudas and Christos Faloutsos and Ibrahim Kamel, "Declustering Spatial Databases on a Multi-Computer Architecture", 5th International Conference on Extending Database Technology, pp. 592-614 , 1996

[7] Bernd Schnitzer, Scott T. Leutenegger, "Master-Client R-trees: A New Parallel R-tree Architecture", 11th International Conference on Scientific and Statistical Database Management, Proceedings, pp. 68-77, 1999.

[8] Bernhard Seeger, Per-Ake Larson , "Multi-Disk B-trees", In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 436-445, 1991.

[9] Youngduk Seo, Bonghee Hong, " Declustering of Trajectories for Indexing of Moving Objects Databases", Database and Expert Systems Applications, International Conference 2004, will be published.

[10] Peter J. Varman and Rakesh M. Verma, "An Efficient Multiversion Access Structure", IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 3 pp. 391-409 , 1997

[11] 서영덕, 홍은석, 홍봉희, "이동체 데이터베이스를 위한 디클러스터링 정책", 한국정보처리학회, 논문지 D, Vol.11 No.7, pp. 1399-1408 2004.12

서영덕

2004년 부산대학교 대학원 컴퓨터공학과(박사)

2004년 ~ 현재 경성대학교 컴퓨터과학과 초빙교수

관심분야 : 지리정보 시스템, 병렬 지리정보 시스템, 객체 지향 데이타베이스