이동체의 관성을 이용한 궤적 색인의 병렬화 기법

Parallelization scheme of trajectory index using inertia of moving objects

서영덕* / Young-Duk Seo
홍봉희** / Bong-Hee Hong

요 약

최근 가장 활발하고 많은 연구가 이루어 지는 시스템중의 하나는 교통 제어 시스템이다. 이 시스템을 효과적으로 지원하기 위해서는 이동체를 효과적으로 저장하고, 시간 혹은 시공간 질의를 효과적으로 수행하기 위하여 높은 성능을 가진 이동체 데이터베이스 시스템이 필요하다. 빠른 이동체 데이터베이스 시스템의 성능은 병렬 색인을 이용하여 구축될 수 있다. 이 논문에서는 시공간 인접성과 이동체 특성에 기반한 디클러스터링 정책을 제시한다. 이동체의 진행 방향에 대한 예측을 통하여 색인의 노드에 대한 성장을 예측하고, 이를 토대로 병렬 색인을 구축한다. 실험 결과 제시된 기법은 기존의 다른 정책에 비하여 최대 15%이상의 성능 향상이 있다. 이 결과는 단일 디스크를 사용할 때에 비하여 디스크당 50%이상의 성능향상 결과이다.

Abstract

One of the most challenging and encouraging applications of state-of-the-art technology is the field of traffic control systems. It combines techniques from the areas of telecommunications and computer science to establish traffic information and various assistance services. The support of the system requires a moving objects database system (MODB) that stores moving objects efficiently and performs spatial or temporal queries with time conditions. In this paper, we propose schemes to distribute an index nodes of trajectory based on spatio-temporal proximity and the characteristics of moving objects. The scheme predicts the extendible MBB of nodes of index through the prediction of moving object, and creates a parallel trajectory index. The experimental evaluation shows that the proposed schemes give us the performance improvement by 15%. This result makes an improvement of performance by 50% per one disk.

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

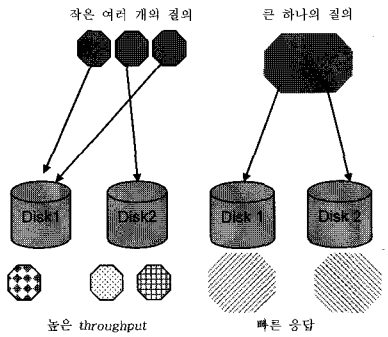
■ 이 논문은 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2005-003-D00465)
■ 논문접수 : 2005. 11. 11 ■ 심사완료 : 2006. 03. 20
* 교신저자 경성대학교 컴퓨터과학과 초빙교수(youngduk@gmail.com)
** 부산대학교 컴퓨터공학과 교수(bhhong@pusan.ac.kr)

1. 서 론

최근의 가장 활발하고 많은 연구가 이루어 지는 시스템중의 하나는 교통 제어 시스템이다. 이러한 시스템은 교통 정보 서비스와 다양한 지원 서비스를 위하여 통신 기술과 컴퓨터 기술을 결합한 형태의 기술이 요구된다. 이 시스템을 지원하기 위해서는 이동체를 효과적으로 저장하고, 시간 혹은 시공간 질의를 효과적으로 수행하기 위한 이동체 데이터베이스 시스템이 필요하다. 시공간 데이터를 관리하는 것은 많은 연구가 진행되고 있는 연구분야이다. 예를 들어 [5] 등이 대표적인 경우로 볼 수 있다.

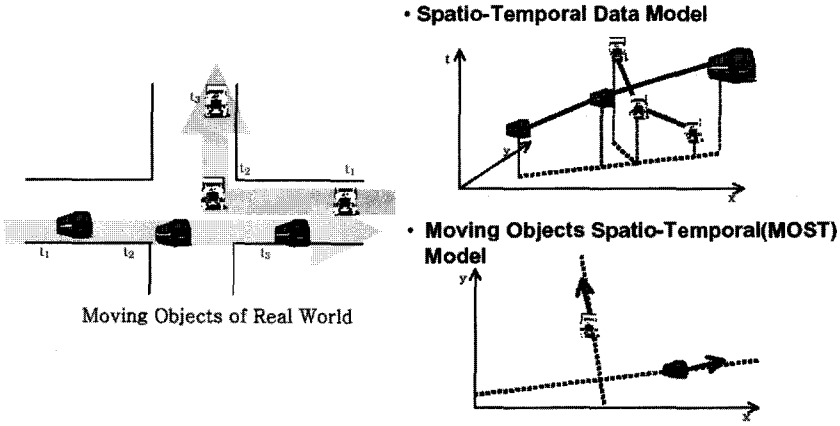
이동체들은 자신의 위치 정보를 일정 주기를 가지고 각각의 서버로 전송하게 된다. 즉, 그림 1과 같이 서로 다른 도로 네트워크를 따라 이동할 때, 이동체는 자신의 위치가 변경되거나, 혹은 일정한 보고 시점에 자신의 위치를 서버 시스템에 보고한다. 이런 식으로 보고된 이동체 위치 정보는 데이터베이스의 모델에 따라 MOST Model[15] [16] 나 Spatio-Temporal model[20]등의 다양한 이동체 모델에 의해 추상화된다. 이렇게 추상화된 데이터는 데이터베이스의 저장 구조 내에 궤적 형태로 저장가능하며, 궤적에 대한 다양한 질의 등이 가능하다.

이때 발생하는 궤적은 빠른 속도로 누적될 수 있으며, 양적으로 많은 양을 가지게 되는 특성이 있다. 즉, 위치기반 서비스와 관련하여 각 서버 시스템으로 보내지는 이동체 데이터의 양은 방대하다. 그리고 이동체의 데이터는 시간에 따라 증가하는 형태로 보고 되기 때문에 계속 증가하는 데이터의 양은 대단히 많으며 그에 대한 아주 많은 연산 처리가 필요하다. 즉 대용량 이동체의 데이터에 대한 수정, 검색 시 시스템의 빠른 처리에 대한 요구가 높아지고 있다.



<그림 2> 디클러스터링의 목적

일반적인 데이터베이스 환경에서 성능 향상을 위하여 많은 방법들이 제시되어 왔다.



<그림 1> 이동체의 이동과 시공간 모델

그 중에서 디클러스터링(declustering) 방법은 그림 2에서와 같이 한번의 질의에 대하여 *동시에 접근되어질 가능성*이 있는 데이터를 다른 디스크로 분산 저장하는 방법을 말한다. 여기서 "*동시에 접근되어질 가능성*"이 있다고 하는 것은 각 객체의 시공간 데이터가 서로 인접하고 있어서 특정 질의가 요청 되었을 때 함께 결과 혹은 결과 집합의 후보로 선택될 가능성이 높다는 것을 말한다. 디클러스터링은 바로 이러한 데이터들을 서로 다른 디스크에 저장하여 같은 I/O 시간에 각각의 디스크에 동시에 접근하고자 하는 방법이다. 이렇게 함으로써, 하나의 큰 질의에 대하여는 둘 이상의 디스크로부터 결과를 찾아 냄으로써 빠른 응답시간을 확보하고, 작은 여러 개의 질의에 대하여는 각 디스크에 각 질의를 분산 수행할 수 있도록 함으로써 높은 처리율을 얻을 수 있다.

기존의 디클러스터링에 관한 연구에서 다차원 속성 데이터베이스에 대한 디클러스터링은 많은 연구가 있어 왔으며, 1차원 속성 데이터의 색인에 관한 연구 또한 몇몇 진행되어 왔다[19][13]. 그리고, 2차원 데이터의 병렬 색인에 관한 연구는 약간의 연구가 있다[6]. 다차원 데이터베이스의 병렬 색인에 관한 연구는 다중 디스크 시스템을 이용한 Parallel R-tree에 관한 연구가 있으며[6], 비 공유 병렬 시스템(shared-nothing parallel system)을 기반으로 하는 연구 등이 있다[9]. 그러나, 2차원 공간 데이터가 시간 차원을 이동하는 즉, 이동체의 궤적 색인에서의 병렬 처리에 관한 연구는 거의 없다.

기존의 방법인 공간 디클러스터링 방법을 이동체 데이터에 대하여 그대로 적용할 경우 시간 도메인에 대한 고려가 전혀 없거나 시공간 모두를 고려한 방법이 없어서 적절한 디클러스터링이 이루어지지 않는 문제점이 있다. 왜냐하면 이동체 데이터는 시공간 특성을 가지는데 그 모든 도메인에 대하여 관련성을 고려하지 않기 때문이다. 특히 이동 객체가 보고하는 데이터는 시간과 공간 모두에 관련성을 가지고 있어서 이들 모두를 함께 고려하는 것은 중요한 문제가 된다. 또한, 이동체의 궤적 데이터는 가장 최근에 보고된 궤적 정보와 연결되는 특성이 있으며, 가고자 하는 일종의 목적지를 가지고 진행하는 특징이 있다. 그러므로 위와 같은 특성을 고려하여 성능 향상을 위한 새로운 이동체 디클러스터링 방법의 제시가 필요하다. 이 논문에서는 이동체에 대한 영역 질의 시 빠른 응답시간을 얻고 전체 시스템의 처리율 향상을 위한 이동체 궤적 색인의 병렬화 방안에 관한 새로운 기법을 제시한다. 이 기법에서는 기존에 정적인 데이터베이스 환경을 가정한 병렬 색인 모델과 달리 동적인 이동체의 방향에 대한 고려를 통하여 보다 현실적인 이동체 색인의 병렬 색인의 구축 기법에 대한 제시를 한다. 즉, 이동체의 진행방향에 대한 직관적인 예측을 통하여 보다 정확한 병렬 색인 기법을 제안한다. 이동체의 진행 방향에 대한 예측을 통하여 색인의 노드에 대한 성장을 예측하고, 이를 토대로 병렬 색인을 구축한다.

실험 결과 제시된 기법은 기존의 다른 정책에 비하여 최대 15%이상의 성능 향상이 있다. 이와 함께, 이 논문에서는 다음과 같은 실험결과를 알 수 있었다.

- 궤적 색인(다차원 색인)의 병렬화 시의 병렬화 율은 노드의 용적률에 의하여 좌우된다.
- 질의의 크기가 작은 경우에는 버퍼의 수에 의하여 성능이 영향을 받고, 질의가 큰 경우에는 버퍼의 크기에 의하여 영향을 받는다.

이 논문의 구성은 다음과 같다. 2장에서는 디클러스터링과 이동체의 궤적 색인에 관한 기존의 연구를 기술한다. 3장에서는 병렬 궤적 색인을 기존의 방법으로 생성할 때의 다양한 문제점을 소개한다. 또한 4장에서는 이동체의 이동 방향을 예측한 디클러스터링

방법을 기술한다. 5장에서 제시된 방법에 대한 구현 및 성능 평가의 결과를 제시한다. 마지막으로 6장에서 결론 및 향후 연구에 대하여 기술한다.

2. 관련연구

이 논문과 관련된 기존의 연구는 크게 두 가지 방향에서 볼 수 있다. 첫 번째는 이동체의 색인에 관한 연구이며 두 번째는 디클러스터링 기법에 관한 연구이다. 그러나, 이 논문에서는 병렬 색인에 관한 기법에 대한 기존의 연구를 위주로 소개한다.

이동체의 빠른 검색과 연산을 위하여 다양한 색인 구조들이 제시되었다. 일반적으로 궤적 데이터에 대한 검색과 연산을 위하여는 공간 데이터베이스에서 사용되던 R-tree를 3차원 시공간 차원에 확장한 자료구조인 3DR-tree[22]와 궤적 질의를 주 연산 대상으로 하는 TB-tree, STR-tree[15], TA3DR-tree[7], MV3R-tree[21], CR-tree[27]등이 있다. 이 논문에서는 공간 데이터베이스에서 가장 보편적으로 사용되는 R-tree 계열의 색인을 기반으로 궤적 데이터를 처리하는 3DR-tree 를 실험의 기준으로 삼았으며, 성능 평가의 대상 색인으로 선정하였다.

데이터베이스의 디클러스터링은 데이터베이스의 성능을 향상시키기 위한 중요한 연구 분야이다. 디클러스터링 혹은 디스크 할당은 지금까지 관계 형 데이터베이스에서 부분 적합질의(partial match query)혹은 영역질의의 차원에서의 성능 향상을 목적으로 많은 연구가 있어왔다[3] [5]. 예를 들어, Field-wise Exclusive(FX) [1], Hilbert (HCAM) [3], Cyclic Allocation [11], Hierarchical Declustering [1]등을 들 수 있다.

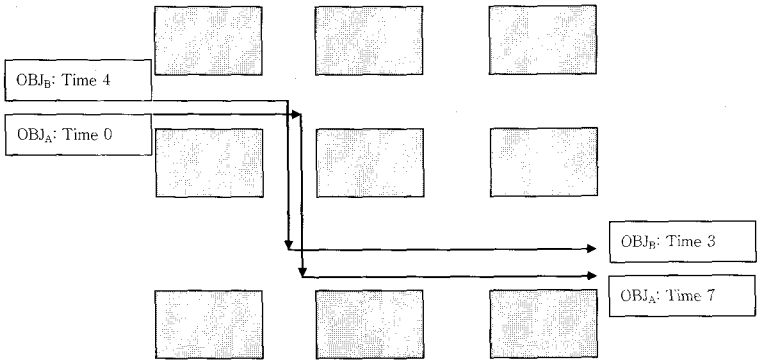
2차원 공간 데이터베이스의 디클러스터링기법은 Coordinate modulo distribution (CMD) 기법[10]과 클러스터링 후의 디클러스터링 기법[8]등을 들 수 있다. CMD 기법에서는 주어진 d 차원의 파일에 대하여 d차원 공간을 디스크 개수인 M개의 파일로 분할 한다. 즉, 그 파일이 M개의 부분 파일로 분할되는 것이다. 각 부분 공간은 디스크에 유일하게 할당되며, 동일 부분 공간에 있는 부분 파일은 동일한 디스크에 할당되도록 한다.

공간 색인에서 기존의 연구는 공간적 인접성만을 이용한 병렬 색인의 구성방법이 제시되어 있다. 즉, 공간 객체간의 인접성인 인접성(proximity)를 이용한 Parallel R-trees[7]과 비-공유 병렬 시스템(shared nothing parallel system)에서의 페이지 분산 배치 방법[12], 그리고, 시공간 인접성을 이용한 STR기법[14]등을 들 수 있다.

STR기법[14]은 이동 객체의 연결성을 기반으로 공간 인접성을 제시하고 있다. 즉, 두 공간 객체가 포함된 노드가 임의의 질의에 의하여 동시에 검색될 확률을 구한다. 이를 두 궤적 노드의 연결성 즉, 궤적의 연결성 정도에 따라 연결성을 기반으로 공간 인접성을 제시하고 있다. 이렇게 함으로써, 기존의 공간 인접성만을 고려한 기법에 비하여 5~10%이상의 성능 향상을 이룰 수 있음을 제시하고 있다. 이 논문에서는 STR기법을 기반으로 이동체의 이동성을 고려하여 병렬 색인 기법을 확장시킨 기법을 제시한다. 이 방법의 결과는 단일 디스크를 이용한 객체 저장 기법에 비하여 디스크 당 50%이상의 성능을 향상 시킨 것이다.

3. 문제 정의

공간객체의 분포는 일반적으로 균등분포나 임의분포를 가정하나, 차량등의 이동체는 일정한 공간 영역(도로 등)위를 이동한다. 그러므로, 이동체의 궤적 집합이 일정한 공간 영역내에 서로 다른 시간대별로 중첩된 위치정보를 가지게 될 가능성은 임의의 공간

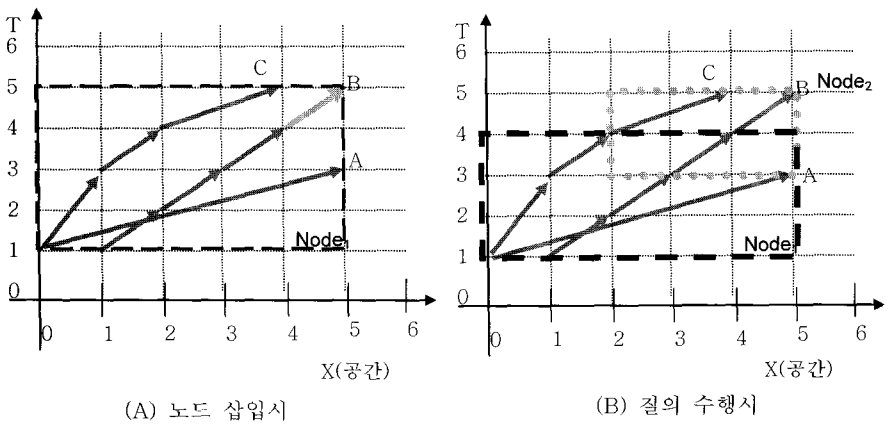


<그림 3> 도로를 따라 이동하는 이동체 집합

상에 있는 공간 객체 집합에 비하여 높다. 즉, 그림 3 같이 일정한 도로 영역을 지나는 차량은 시간 축을 포함하는 경우 많은 이동체간의 중첩이 주로 시간 축과 공간 축에 대하여 서로 다른 패턴으로 발생한다. 이와 같이 이동체가 이동할 때, 이동체의 시공간 분포는 공간 축에 대하여 많은 겹침을 보이게 되며, 시간 축에 대한 겹침과 공간 축에 대한 겹침은 서로 다른 양상을 보인다.

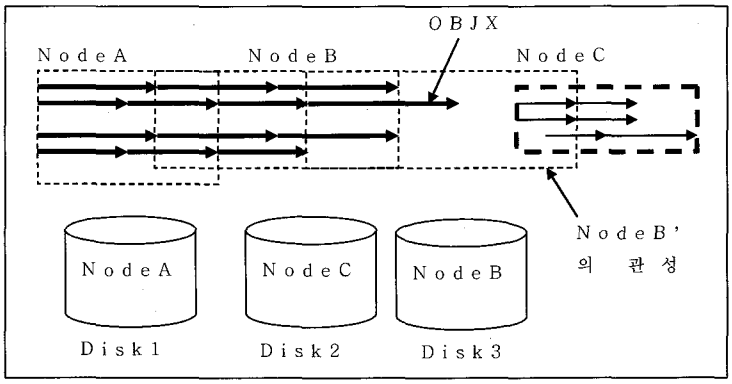
기존의 디클러스터링 기법은 색인의 노드가 생성되는 시점의 공간 관련성을 이용하여 인접성에 대한 판단을 수행한다. 즉, 데이터베이스 혹은 색인에 데이터가 삽입되는 시점과 질의가 수행되는 시점에서 노드의 크기가 변하지 않음을 가정하고 있다. 이러한 이유로 기존의 색인에 대한 병렬화는 데이터가 삽입되는 시점 혹은 존재하는 데이터에 대하여 분산 배치가 수행되며, 이후 삽입을 고려하지 않는다.

그러나, 이동체의 궤적 색인은 데이터가 삽입되는 시점에 생성된 노드와 일정한 시간이 지난 후 즉 질의가 수행되는 시점의 노드의 크기와 모양이 다를 수 있다. 그것은 이동체의 시간 축에 대한 연속적인 변수의 입력에 그 원인이 있다.



(A) 노드 삽입시 (B) 질의 수행시

<그림 4> 노드 삽입 시점과 질의 수행 시점의 노드 변화



<그림 5> 이동체의 진행 속도와 디스크 배치

예를 들어 공간 축을 1차원으로 가정하고, 시간에 따라 이동하는 3개의 공간 객체를 그림 4 (A)와 같이 둘 때, 각각의 객체는 하나의 색인의 노드에 삽입될 수 있다. 여기서 세 개의 공간 객체 A,B,C는 공간 축을 시간의 증가에 따라 진행한다고 가정하였다. 일반적인 공간 객체의 경우, 생성된 노드는 질의 시점까지 유지된다고 가정할 수 있으나, 이동 객체의 경우 그렇지 못함을 알 수 있다.

이러한 형태의 궤적 데이터의 색인에 대한 삽입은 공간 데이터와는 서로 다른 성격을 보여준다. 즉, 공간 데이터의 입력은 색인에 삽입될 위치에 대한 예측이 불가능하며, 임의적인 성격이 있으나, 궤적의 경우 항상 직전 궤적이 포함된 노드에 저장될 가능성이 높아 진다. 즉, 그림 4같은 예를 들 수 있다. 그림 4에서 (A)는 노드 삽입 시점의 색인 노드의 모양으로 가정하면, 일정한 삽입이 발생한 시점인 그림 (B)는 노드의 분할이 발생하는 시점으로 볼 수 있다. 즉, 노드가 생성되는 시점의 노드 형태와 노드가 분할될 시점의 노드의 형태와 데이터의 내용은 공간 데이터에 비하여 상당히 많은 유사점을 가지면서 일정 정도 예측이 가능함을 알 수 있다. 예를 들어, 그림 5의 궤적을 3DR-tree로 색인 할 때, NodeB에서 분할되는 NodeB' 은 기존의 방법에 의하면 공간 인접성이 가장 적은 NodeC가 Disk2에 적재될 것이다. 그러나, 이러한 방법은 NodeB' 의 관성 결정 객체인 ObjX의 이동 진행 방향과 속도에 의해 노드 B' 이 가득차는 시점에 NodeC와 겹치게 될 가능성이 크다.

이 논문에서는 기존 연구의 이러한 문제점을 극복하고, 보다 효과적인 질의의 병렬화를 이루기 위하여 이동체의 속도와 진행 방향을 이용한 병렬 색인 구축 기법을 제시한다.

4. 이동체의 관성을 이용한 예측 기반 노드 분산 배치

기존에 제시된 이동체 궤적의 인접성 모델[14]은 노드가 생성되는 시점과는 상관없는 모델이다. 즉, 기존의 인접성 모델 혹은 디클러스터링 기법은 새로운 노드가 발생하는 시점 혹은 페이지를 처음 저장하는 시점을 기준으로 제안된다. 그러나, 궤적 노드는 이동체의 보고가 계속 되면서 지속적으로 수정되며, 새롭게 생성된 노드에 삽입될 확률이 높다. 즉, 생성된 노드의 진행 방향이나 크기의 변화 가능성에 대한 예측이 가능하다. 그러므로, 이 장에서는 이동체의 인접

성을 노드 생성 시점이 아닌 실제 질의가 실행될 시점인 안정화 이후의 시점에서의 인접성으로 확장하기 위한 기법을 제시한다. 이를 위하여 이 장에서는 이동체의 관성을 정의한다. 또한, 이를 색인의 노드 관점으로 확장하여 이동체의 이동에 따른 색인 노드의 진행 방향을 예측할 수 있는 노드의 관성을 정의한다.

4.1 이동체의 관성

이동체는 일정한 속도와 진행하고자 하는 방향을 가지는 다음과 같은 특성을 가진다.[11] 이 논문에서는 이를 다음과 같이 가정한다.

가정1) 이동체의 진행방향은 그 이동체의 최종 목적지에 부합하고자 하는 방향을 위한 경로를 따라 이동한다.

가정 2) 이동체의 속도는 기존의 속도의 변화 범위 내에서 유지된다.

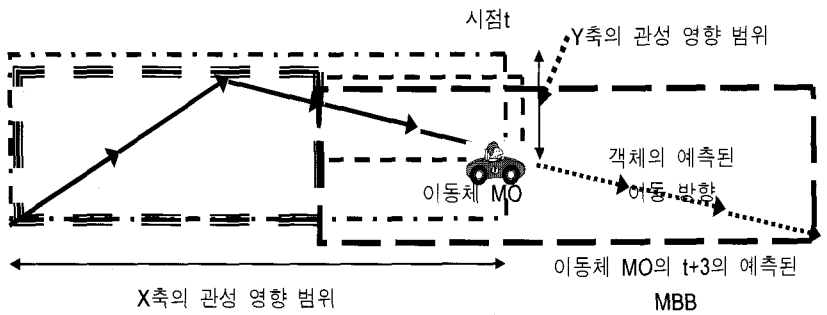
가정 1은 이동체의 진행 방향에 대한 가정이다. 즉, 이동체의 진행 방향은 해당 이동체가 도착하고자 하는 최종 목적지를 향해 간다는 의미이며, 실 세계의 이동체는 이러한 가정을 따른다고 볼 수 있다. 또한 가정 2는 이동체의 급격한 속도의 변화에 대하여 가정하는 것으로, 실 세계의 이동체에 대한 타당한 가정이다. 또한, 이를 통하여 일정한 시간 간격 동안에 발생한 궤적의 양과 다음 일정한 시간 동안 발생할 궤적 데이터의 양에 대하여 추측할 수 있다.

이동체의 궤적은 이동체 데이터베이스에서 이동체가 보고한 하나의 간격에서의 이동 경로를 의미하는 세그먼트의 집합이다. 그러므로, 가정 1과 2에 의하여 이동체의 방향과 속도가 일정하고자 하는 성질에 의해 이동체의 궤적 역시 일정한 속도와 방향으로 진행하고자 하는 특성을 가지게 되며, 이를 다음과 같이 정의한다.

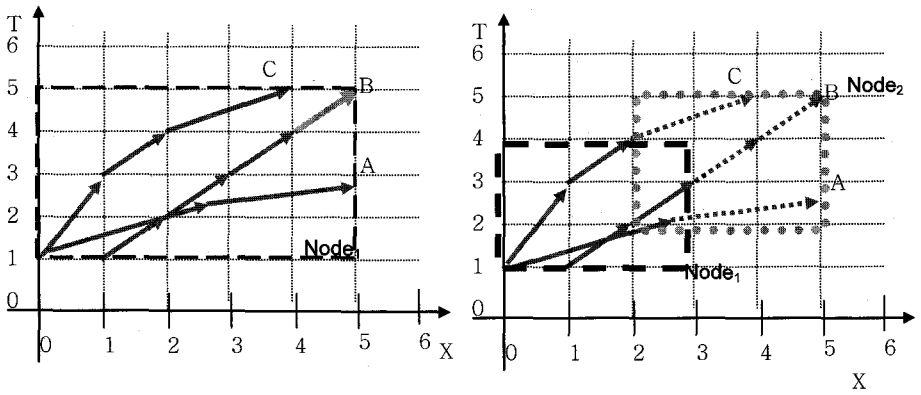
정의 1) 관성의 영향 범위(P)(Influence scope of Inertia): 궤적 segment si와 si+1의 속도가 반전되는 지점 즉, 각 축의 속도가 + → - 가 되거나 - → +로 변하는 i로부터 현재 지점

정의 2) 궤적의 관성: 데이터베이스 내의 임의의 이동체 A의 궤적 trjA가 존재하고, 이의 세그먼트S가 있을 때, 시점 n에서 이동체의 궤적 세그먼트 S_n^A 는 S_{n-P}^A 로 부터 S_{n-1}^A 까지의 방향과 속도를 따르고자 하는 성질을 가지며 이를 관성이라 한다.

정의 1의 관성의 영향 범위는 가정 2로부터 정의된 것이며, 정의2는 가정 2로부터 정의된 것이다. 정의 1은 짧은 시간 간격에서 정지, 역방향 진행 등의 순간적인 돌출 행동 등을 무시하고 효율적으로 진행 방향을 계산하여, 색인이나 데이터 페이지의 단위를 넘지 않는 범위 내의 이동체의 진행 방향과 속도에 의해 이동체의 방향과 속도를 예측하고자 함이다.



〈그림 6〉 이동체 관성과 관성 영향 범위



(A) 이동체 B에 의한 노드 분할 (B) 노드 분할 후

<그림 7> 노드 삽입 시점과 질의 수행 시점의 노드 변화

예를 들어, 그림 6에서 이동체 MO는 X축의 양의 방향으로 진행하고 있으며, Y축에 대하여는 양의 방향으로 증가하다 최종 보고 시점 t에는 현재 음의 방향으로 진행하고 있다. 여기서 이동체는 다음 일정한 시점 t+3에 현재의 MBB와는 다른 예측 MBB를 가짐을 예상할 수 있다. 또한, 이 크기는 가정 2에 의하여 예측할 수 있음을 알 수 있다.

4.2 이동체 궤적 색인과 예측 MBB

이동체의 궤적은 전 절에서 기술한 바와 같이 일정한 방향에 대한 예측 가능성을 가지고 있다. 이것은 이동체의 궤적 데이터에 대한 색인의 구축 시에도 동일한 개념으로 적용될 수 있다.

그림 4가 그림 7로 시간에 따라 객체가 추가되었을 경우, 그림 7에서 원래의 노드가 $Node_1$과 $Node_2$로 분할되었음을 가정하자. 이때, $Node_1$에는 실선의 궤적이 포함되며, $Node_2$에는 점선의 궤적이 포함된다고 가정하자. 이후 이동체의 이동에 의하여 궤적은 지속적으로 발생할 것이며, 새롭게 발생하는 궤적들은 주어진 색인의 삽입정책에 따라 색인에 삽입될 것이다. 여기서 기존의 삽입 정책들은 색인의 식별력을 높이기 위한 정책으로 진행되며, 대다수 색인의 삽입 정책은 면적이나 겹침을 최소화 시키기 위한 방향으로 진행된다. 그러므로, $Node_1$은 더 이상의 궤적이 삽입되지 않을 가능성이 높다. 이 논문에서는 이 노드를 안정노드($Node^S$)라 정의하며, 안정노드는 더 이상 새로운 궤적 집합이 삽입될 가능성이 낮은 노드를 의미한다. 반면, $Node_2$에는 새로 발생하는 궤적이 지속적으로 삽입될 가능성이 $Node_1$에 비하여 현저히 높음을 알 수 있다. $Node_2$와 같이 새로운 객체의 삽입이 예측되는 노드를 *활성 노드($Node^A$)*라 정의한다. 즉, 활성 노드는 새로운 궤적이 삽입될 가능성이 있는 노드를 의미한다. 활성 노드의 판단은 노드의 시간 축이 노드 분할 시 현재 시점 T_N과 동일한지의 여부로 판단한다. 즉, 노드에 포함된 객체의 시간 축 좌표가 현재 시간인지 여부를 판단한다. 이것은 현재 시간을 포함하는 노드에 새로운 객체가 삽입될 가능성이 높기 때문이다. 활성 노드(예를 들어 그림 7의 $Node_2$)는 지속적으로 삽입되는 궤적 집합에 의하여 특정 시점 후 분할 될 것이다. 이 시점을 이 논문에서는 *미래 분할 시점*(T_F)라 정의한다.

미래 분할 시점 T_F에서 활성노드($Node^A{}_{TF}$)

는 안정노드($Node^{S}_{TF}$)로 성격이 변하며 더 이상 새로운 궤적을 받을 가능성이 낮아진다. 그러나, $Node^{A}_{TF}$ 의 MBB와 $Node^{S}_{TF}$ 의 MBB의 크기와 방향은 틀려진다. 이 논문에서는 기존의 연구[14]에서 T_N에서의 $Node^{A}_{TN}$를 이용하여 생성하던 병렬 색인의 생성 기법을 $Node^{S}_{TF}$의 인접성을 이용한 병렬 궤적 색인기법으로 변환하여 적용하는 기법을 제시한다.

미래 분할 시점 T_F는 색인에 삽입된 궤적에 대한 정보를 이용하여 색인의 특성에 따라 계산 가능하며, 이 논문에서는 가정 2에 의하여 다음과 같은 방식으로 결정한다.

$T_F = MIN(Node^{A}_{TN},T) + LENGTH(Node^{S}_{TN},T)$ 식1

MIN함수는 주어진 노드($Node^{A}_{TN}$)의 특정 축(T)에 대한 최소값을 돌려주는 함수이며, LENGTH는 주어진 노드($Node^{S}_{TN}$)의 특정 축(T)의 길이를 돌려주는 함수이다. 식1은 궤적 색인 노드의 시간 축에 대한 평균 크기는 동일하다고 가정한다. 즉, 가정 1에 의하여 이동체의 이동 속도는 동일하다고 가정한다.

그러나, T_F는 미래 시점의 분할 시점만을 산출하고 있으며, 분할 후 안정노드의 시간축의 크기는 T_F와는 다르다. 그것은 3DR-tree의 경우 분할된 노드의 끝점으로부터 새로운 노드를 생성하는 것이 아니라, 노드 내에 포함된 객체 집합을 특정한 규칙에 따라 분배하는 특성을 가지기 때문이다. 분배 기준에 따라 예측하긴 힘드나, 이 논문에서는 시간축의 경우 용적율(C)에 의해 줄어든다고 가정한다. 즉, TF 시점에서 안정화된 노드($Node^{S}_{TF}$)의 시간 축의 MBB는 다음과 같이 결정된다.

$MIN(Node^{S}_{TF},T) = MIN(Node^{A}_{TN},T)$

$MAX(Node^{S}_{TF},T) = MIN(Node^{A}_{TN},T) + LENGTH(Node^{S}_{TN},T)*C$ ——식2

즉, 시간 축은 현재 분할된 노드의 MIN값과 이전 노드의 시간축 길이의 용적률의 크기만큼 늘어나는 것을 가정한다.

공간축의 경우 시간 축과는 다른 방식으로 계산해야 한다. 즉, 미래 분할 시점 T_F에서의 공간 축의 크기는 벡터 값으로 표현되어야 한다. 이것은 시간의 경우 지속적으로 증가하는 특징을 가지지만, 공간 축의 경우 이동체는 현재 속도의 방향에 따라 증가 혹은 감소할 수 있기 때문이다. 또한 색인 노드의 공간 MBB는 공간 축을 이동하는 객체 중 가장 큰 객체의 MBB에 의하여 결정된다. 즉, 궤적 색인의 노드는 특정 축에 대하여 가장 외곽에 존재하는 객체의 경계에 의하여 노드의 MBB가 결정된다. 이때, 가장 외곽에 존재하는 궤적을 이 논문에서는 *모양 결정 궤적*(trjD)라 정의한다. 그러나, T_N에서의 trjD인 $trjD_{TN}$는 T_F에서의 trjD 즉, $trjD_{TF}$와 다를 수 있다. 이것은 이동체의 속도와 관련이 있다. 즉, 현재 시점에서 노드의 중앙에 위치하지만 빠른 속도로 이동하는 객체는 T_F시점에서 trjD가 될 수 있으며, 이것은 T_F와 빠른 속도로 이동하는 이동체의 T_F에서의 이동거리로 판단할 수 있다. 즉, $trjD_{TF}$는 다음 조건을 만족한다.

조건.

For all trajectory i in $Node^{S}_{TN}$

$LENGTH(trjD,T_F) = MAX(LENGTH(i, T_F))$

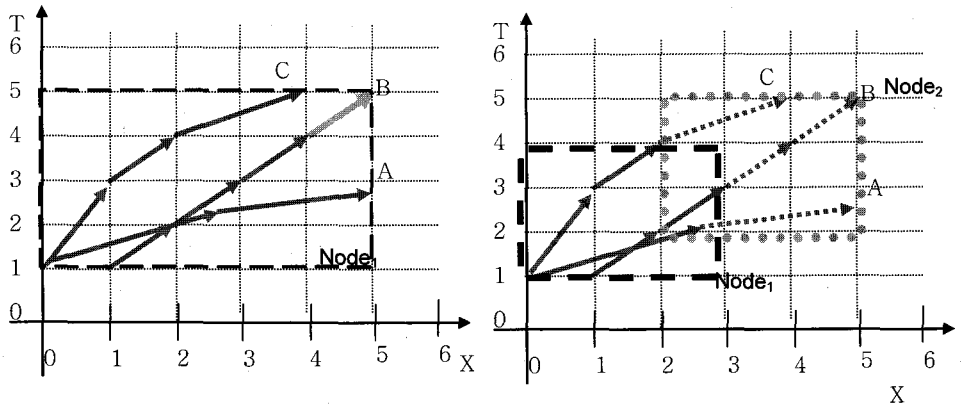
즉, T_N이 아닌 T_F시점에서 가장 멀리 나갈 수 있는 노드를 trjD로 결정한다. 또한 trjD는 각 축에 대하여 2개가 발생할 수 있다. 즉, 양의 방향과 음의 방향으로 이동체가 이동하며, 이 이동 방향(direction, dir)에 대하여 T_F시점에 가장 외곽 사각형을 형성할 궤적을 각각 $trjD^{+}_{dir}$와 $trjD^{-}_{dir}$로 결정한다.

그러므로, 각 방향에 대한 이동체 궤적 노드의 T_F시점에서의 MBB인 IMBB는 다음과 같이 결정된다.

IMBB(min_x, min_y, max_x, max_y, min_t, max_t) =

($traD\_X^{-}_{TF}$, $traD\_Y^{-}_{TF}$, $traD\_X^{+}_{TF}$, $traD\_Y^{+}_{TF}$, $MIN(Node^{S}_{TF},T)$, $MAX(Node^{S}_{TF},T)$)

이때, 각 방향에서 진행하는 객체가 없는 경우, 해당 축에 대한 T_N에서의 값이 해당 필드의 값이 된다.



(A) 이동체 B에 의한 노드 분할 (B) 노드 분할 후

<그림 8> 이동체의 분할과 미래 위치 추측에 의한 예측 MBB

예를 들어, 1차원 공간을 이동하는 이동체 궤적이 그림 8과 같다고 가정하자. 이때 노드의 최대 포함 객체수가 8이며 평균 용적률이 60%인 색인(R-tree계열의 색인)을 가정할 때, T_N=3인 시점에서 궤적 B에 의하여 분리되었다. 이 예에서 해당 노드의 IMBB는 다음과 같이 결정된다.

$MIN(Node^{S}_{TF},T) = MIN(Node^{A}_{TN},T) = 2$

$MAX(Node^{S}_{TF},T) = MIN(Node^{A}_{TN},T) + LENGTH(Node^{S}_{TN},T)*C = 2 + 3*0.6 = 3.8$

$trjD^{+}_{TF}$ = Trajectory_of(A)

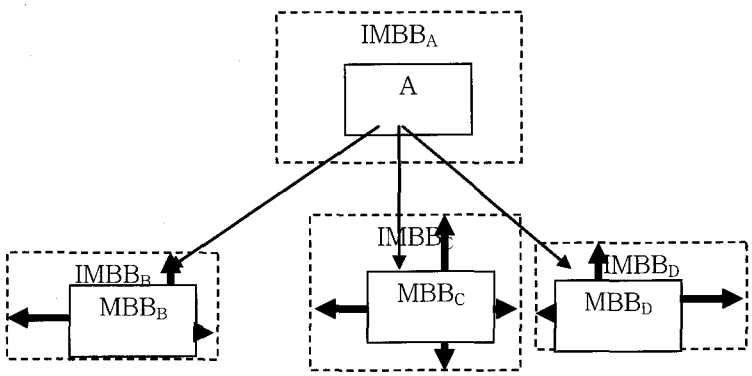
$trjD^{-}_{TF}$ = NONE

$max\_x_{TF} = 2+3*1.8 = 7.4$

$min\_x_{TF} = 2$

즉, 현재 노드내의 객체들이 현재 시점의 속도와 방을 기준으로 진행하는 경우 다음 노드가 가득 차는 시점 T_F에서의 X축은 (2,7.4)의 위치함을 예측할 수 있다. 여기서 max_x_{TF}와 min_x_{TF}는 미래분할 시점 TF에서의 X축에 대한 min값과 max값에 대한 예측값이다.

색인 노드 중 비 단말노드의 관성 역시 궤적 포함 노드의 관성과 유사하게 볼 수 있다. 즉, 포함하는 하위 노드의 MBB중 모양 결정 노드(NodeD)로 볼 수 있는 노드가 존재하며, 현재 노드의 관성은 이 관성 결정 노드에 의하여 결정된다.



<그림 9> 노드의 관성

그림 9에서 상위 노드A의 관성에 의한 진행 방향은 하위 노드들의 관성 크기중 가장 큰 방향 벡터의 합으로 결정될 수 있다. 즉, 노드 A의 관성은 각 방향에 대하여 다음 노드의 방향의 합이다.

4.3 예측 MBB에 의한 병렬 색인 생성

이 논문에서는 기존 노드의 관성을 이용하여 새롭게 생성된 노드의 인접성을 결정하는 것을 제시한다. 그림 10에서 새로이 생성되는 궤적 노드 G는 기존의 색인 노드 D의 관성이 $IMBB_D$에 의한 인접성이 적용되어야 한다.

색인 노드의 디클러스터링은 노드의 관성에 의한 MBB에 의해 생성된 인접을 기준으로 적용된다. 즉, 그림 10의 노드 A, B, C, D가 색인 노드의 관성을 그대로 가진다고 가정할 경우 새로이 생성되는 노드G는 D의 관성을 통한 IMBB를 가질 것이다. 또한 F는 D와 G의 관성에 의한 노드 관성을 가지게 된다. 이 관성에 의한 진행 방향상의 IMBB는 기존의 인접성의 연산 방법과 다른 결론을 가지게 된다.

이 논문에서는 이동체 색인 노드의 디스크 적재 위치를 정할 때, 다음과 같이 노드 적재시 관성에 의한 디스크 선택 방법을 제시한다.

임의의 주어진 노드 N_i이 새로 발생하는 노드일 때, N_i의 관성 $I(N_i)$ MBB와 형제 노드들의 관성MBB에 의하여 N_i가 적재될 디스크가 결정되며, Ni의 관성 MBB는 형제 노드의 관성 MBB와 비교함으로써 다음과 같이 결정된다.

$P(I(N_i),N_k) = P(I(N_i),I(N_k))$ ——식5

이때, N_i가 적재될 디스크 D(N)은 $P(I(N_i))$ 중에서 가장 낮은 인접성을 가진 형제 노드가 적재된 디스크로 결정된다. 즉, 다음과 같다.

$D(N) = \{x | P(I(N_i)) = \min\}$ ——식6

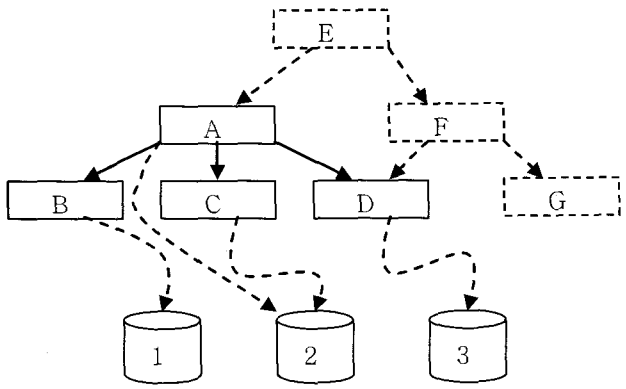
이 논문에서는 인접성 모델을 [14]로부터 확장하여 동적인 환경에 적합하도록 다음과 같이 확장하였다. 기존의 새로 생성된 노드 $A(x_0,y_0,t_0)$-(x_1,y_1,t_1)와 임의의 주어진 궤적 노드 $B(X_i,Y_i,t_i)$-(X_j,Y_j,t_j)의 예측기반 궤적 인접성(P_T)은 다음과 같다.

$$P_T = \frac{1+N_C}{C+1}\left(\frac{1}{2}+O(IMBB(Ax),Bx)\right)\left(\frac{1}{2}+O(IMBB(Ay),By)\right)\left(\frac{1}{2}T_N+O(PMBB(At),Bt)\right)$$

여기서 O(A,B)는 선분의 겹침이며, C는 노드의 capacity, NC는 노드내에서 연결된 선분의 수이다.

이를 이용한 노드 분할시의 디스크 선택을 위한 방법은 알고리즘 1과 같다.

알고리즘 1는 현재 분할된 노드의 형제 노드로부터 각각의 인접성을 계산하여 가장 낮



<그림 10> 궤적 노드의 분할과 디스크 할당

```
DiskID Function FindDisk(Node node){
    Proximity MinProx;
    DiskID SelectedDisk;
    MinProx = MaxValue;
    For (snode in All SiblingNodesExceptNode(node)){
        nodeProx = TrjProximity(node,snode)
        if(nodeProx < MinProx){
            MinProx = nodeProx;
            SelectedDisk = snode;
        }
    }
    return SelectedDisk;
}
```

<그림 11> <알고리즘 1> 디스크 선택을 위한 알고리즘

은 인접성을 가지는 노드가 적재된 디스크를 찾는다. 또한 이 알고리즘에서는 적절한 디스크를 찾기 위하여 이웃 노드의 인접성을 조사하여 인접성이 가장 낮은 디스크를 선정한다.

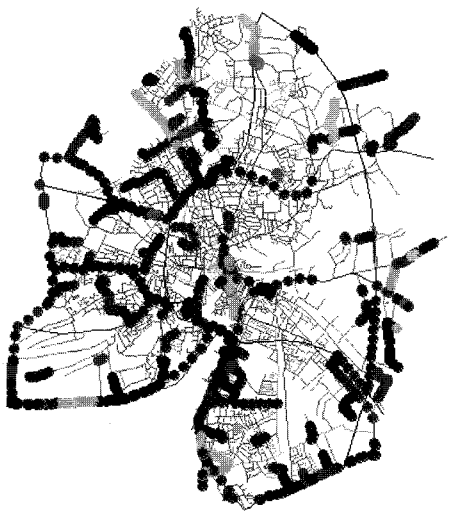
예를 들어 그림 10에서 발생하는 새로운 노드인 NodeG의 적재 위치는 알고리즘 1에 의하면 관성에 의한 MBB의 겹침이 가장 적은 디스크인 Disk1에 적재된다.

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 수행하였다. 모든 궤적 집합은 [2]에서 제공하는 도로 데이터 생성기(Traffic data generator)를 이용하여 Oldenburg의 네트웍 데이터를 기반으로 총 25,044개의 궤적 집합[그림 11]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 12> 실험 데이터의 visualization

또한 다중 디스크를 활용하기 위하여 3DR-tree를 구현하여 실험하였다. 구현된 3DR-tree는 Microsoft Windows □의 C++로서 구현되었으며, 실험은 Intel□ Pentium IV 시스템에서 수행되었다. 또한 이 실험에서는 최대 12개까지의 디스크를 실험 대상으로 삼았다.

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

- NB: 버퍼수(0…1024개)
- SB: 버퍼 혹은 페이지의 크기(1K…16K bytes)
- D:디스크의 수(1…12개)
- SQ:질의의 크기(데이터 영역의 0.1%)

또한 질의는 총 500개를 임의로 random하게 생성하였다. 또한 이 실험에서 성능 비교의 대상이 되는 기법은 기본적으로 많이 사용되는 round robin 기법과 random assign, 기존의 공간 데이터에 대하여 적용되는 PI(Proximity index)기법[7]과 본 논문에서 제시하고 있는 시공간 인접성을 이용한 인접성 기법(spatio-temporal proximity: STP 기법)간의 성능을 비교하였다.

5.2 시공간 인접성 모델을 이용한 성능 비교

첫 번째 실험은 제시한 STP기법이 다른 방법과의 성능 비교를 통하여 기존에 제시된 방법에 대한 상대적 성능을 비교/분석하기 위한 목적으로 실험을 진행하였다.

첫 번째 실험에서는 제시한 기법 즉, STP기법과 PI, Round robbin방법, random Assign 방법의 성능을 비교하였다. 이 실험에서는 다음과 같은 실험 요소를 이용하였다.

NB = 100
SB = 1024
D = 1, ..., 12
SQ = 0.1%

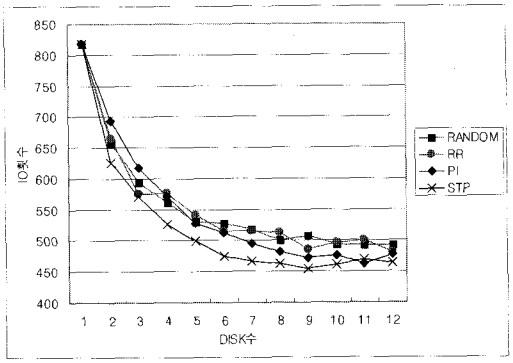
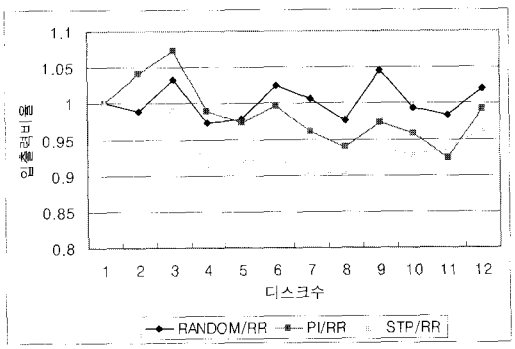


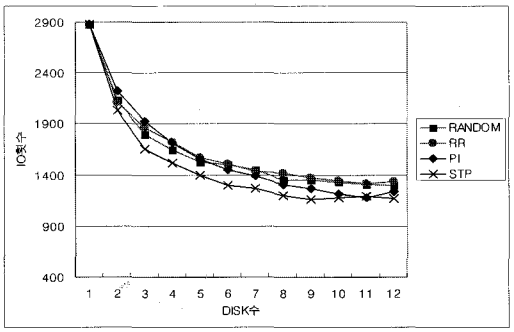
그림 13 깊이 우선 탐색에 의한 성능 평가

그림 12에서 보이듯이 제시된 알고리즘, STP는 Round robin방법과 Proximity index(PI)에 비하여 최대 10%정도의 성능 향상을 가져왔다. 그림 13에서 디스크 수에 따라 각 방법의 성능 증가가 어느 정도인지를 나타내고 있다. 제시된 방법들은 모두 디스크가 많아질수록 전체적인 성능이 비슷해짐을 알 수 있다.

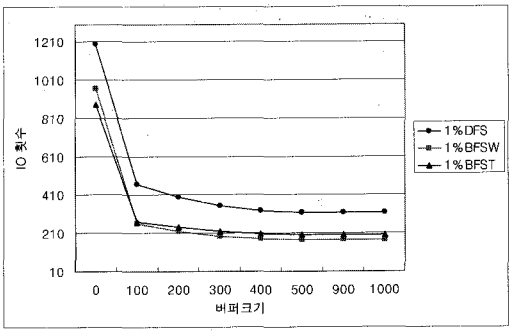


<그림 14> RoundRobin방식에 대한 성능 비교

동일한 실험조건에서 질의의 크기를 0.5%로 늘렸을 때의 그래프가 그림 14이다. 다른 인접성 모델에 비하여 최대 11%의 성능 향상을 가져 왔으며, 디스크 입출력 횟수는 디스크수가 8개를 넘기면서 거의 고착화되었다. 이것은 자식 노드에 대한 선택률이 디스크 수에 비하여 그리 많아 지지 않아서 인 것으로 분석된다.



<그림 15> 질의크기 0.5%인 경우에 따른 입출력 횟수



<그림 16> 버퍼 개수 변화에 따른 입출력 회수

5.3 버퍼 개수에 대한 성능 평가

버퍼 개수가 시스템의 성능에 미치는 영향을 조사하기 위하여 이 논문에서는 버퍼의 크기와 개수를 달리하는 실험을 수행하였다. 즉, 다음과 같은 변수를 적용하여 실험을 수행하였다.

NB = 100, 200, 400

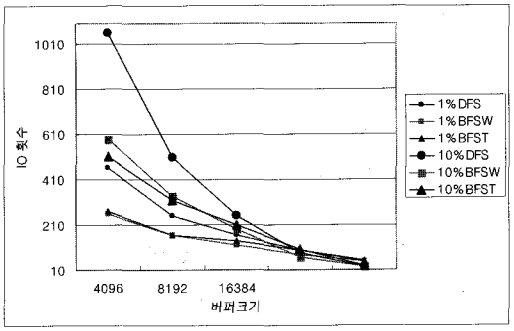
SB = 1K,2K,4K,8K,16K

DD = 1…12

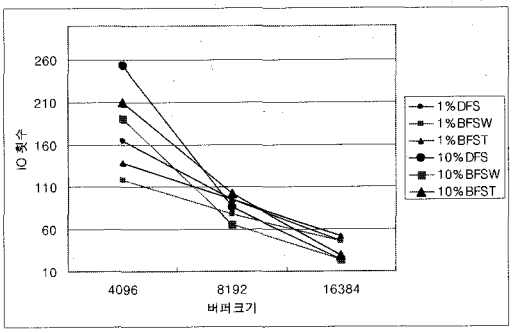
SQ = 0.1%, 0.5%

그림 15 에서 질의의 크기가 0.1%이고, 버퍼의 크기를 1K, 디스크 수는 12개로 고정시킨 상태에서 제시된 STP기법을 통하여 구축된 병렬 색인에 대한 영역 질의를 수행하였을 때, 각 검색 방법에 대하여 성능을 비교하였다. 질의의 수가 500개인 상태에서 버퍼 수는 100개를 넘어서면서 거의 효용성이 없어짐을 알 수 있다. 또한 미세하지만, 버퍼수가 크질 때, BFSW에 의한 검색 기법이 BFST에 의한 검색 기법에 비하여 버퍼가 많을 때의 성능이 높음을 알 수 있다.

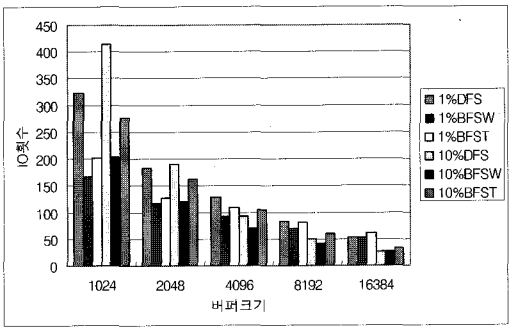
또한 버퍼의 개수를 고정시킨 상태에서 버퍼의 크기를 증가하였을 때에 대한 성능은 그림 16에 나타나고 있다. 성능상에서 버퍼를 많이 사용할수록 좋은 결과를 나타내는 것은 당연한 결과이나, 버퍼 수가 늘어나더라도 본 논문에서 제시한 기법의 성능이 좋은 것을 알 수 있다.



<그림 17> 버퍼크기 변화에 따른 성능



<그림 18> 54의 확대 그래프



<그림 19> 버퍼에 동일한 메모리 사용시 디스크 접근 횟수

그림 18은 동일한 메모리를 사용하여 질의를 수행하였을 때의 수행결과를 나타낸다. 표 1과 같은 버퍼 크기와 버퍼수를 각각 적용하여 실험을 수행하였으며, 12개의 디스크를 활용하여 실험하였다.

실험결과, 질의의 크기가 작은 경우에는 버퍼의 수가 많을수록 성능이 좋아지는 영향을 받고, 질의가 큰 경우에는 버퍼의 크기가 클수록 유리하다. 또한 깊이 우선 정책의 경우에는 버퍼의 크기가 클수록 성능이 좋아졌으며, 큐를 이용한 넓이 우선 정책으로 검색을 수행하는 경우 버퍼의 크기가 커지더라도 질의 성능의 향상에는 많은 도움을 주지 못했다.

<표 1> 버퍼 크기에 따른 성능평가 실험에 사용된 메모리 량

버퍼크기	버퍼수	사용메모리
1024	1280	1.2M
2048	640	1.2M
4096	320	1.2M
8192	160	1.2M
16384	80	1.2M

6. 결론 및 향후 연구

고성능 데이터베이스 시스템은 무선 통신과 GPS (Global Positioning System)등의 위치 인식 시스템의 보급이 늘어남에 따라 발생하는 위치기반 서비스, 물류관제 서비스 등 이동체와 관련된 여러 가지 새로운 서비스에서 효과적인 데이터 처리를 위하여 많은 필요성이 제기되고 있다. 고성능 데이터베이스 시스템에서의 성능은 다양한 방법으로 확보할 수 있다. 그 중에서 저비용으로 높은 성능을 발휘할 수 있는 병렬 디스크는 많은 가능성을 포함하고 있다.

이 논문에서는 이동체에 대한 궤적 질의와 영역 질의 시 빠른 응답 시간을 얻고 전체 시스템의 처리율 향상을 위한 이동체 색인의 병렬화 기법을 제시하였다. 즉, 이동체의 진행 방향에 대한 관성을 통하여 다음 시점에서의 이동체의 진행 방향에 대한 판단을 통하여 이동체 궤적의 관성을 정의하였다. 또한 이를 이용하여 이 논문에서는 색인 노드의 관성을 정의하고, 이를 통한 이동체 데이터베이스에서의 색인의 디클러스터링을 제안하였다.

이상의 실험 결과를 통해서 이 논문에서 제시한 방법이 기존에 제시된 방법에 비하여 15%이상의 빠른 성능 향상을 가져왔음을 알 수 있다. 이러한 결과는 단일 디스크를 사용할 때에 비하여 디스크당 50%이상의 성능향상 결과이다.

이와 함께, 이 논문에서는 다음과 같은 실험 결과를 알 수 있었다.

- 궤적 색인(다차원 색인)의 병렬화 시의 병렬화 율은 노드의 용적률에 의하여 좌우된다.
- 질의의 크기가 작은 경우에는 버퍼의 수에 의하여 성능이 영향을 받고, 질의가 큰 경우에는 버퍼의 크기에 의하여 영향을 받는다.

향후 이 연구에서 제시된 영역 질의 기법을 보다 확장하여, TB-tree에서의 질의 특히 영역 질의와 복합질의의 성능 향상을 위한 기법에 관한 연구를 수행할 것이다. 또한 궤적질의 수행 시 병렬 색인에서의 비용모델과 최소 비용에 대한 연구를 수행할 것이다.

참고문헌

1. Andrew U. Frank, Stephan Winter. "CHOROCHRONOS Consortium", First CHOROCHRONOS Intensive Workshop CIW' 97, TECHNICAL REPORT CH-97-02.
2. A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain and Son Dao, "Querying the uncertain position of moving objects". Temporal Databases: Research and Practice. Springer Verlag, 1997, pp. 310-337.
3. A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain and Son Dao. "Modelling and

Querying Moving Objects", Proceedings of the Thirteenth International Conference on Data Engineering, 1997, pp. 422-432.

4. T.-S. Yeh and B. de Cambray. "Modelling Highly Variable Spatio-Temporal Data", Proceedings of the Sixth Australasian Database Conference, Vol. 17, No. 2 1995.
5. Bernhard Seeger, Per-Ake Larson , "Multi-Disk B-trees", In Proceedings of ACM SIGMOD Conference on Management of Data, 1991, pp. 436-445.
6. Bonggi Jun, Bonghee Hong, Byunggu Yu, "Dynamic Splitting Policies of the Adaptive 3DR-tree for Indexing Continuously Moving Objects", Database and Expert Systems Applications, 14th International Conference, DEXA 2003, 2003, pp. 308-317.
7. Nick Koudas and Christos Faloutsos and Ibrahim Kamel , "Declustering Spatial Databases on a Multi-Computer Architecture", 5th International Conference on Extending Database Technology, 1996, pp. 592-614.
8. Dieter Pfoser, Christian S. Jensen, Yannis Theodoridis, "Novel Approaches to the Indexing of Moving Object Trajectories" , In Proceedings of 26th International Conference on Very Large Data Bases, 2000, pp. 395-406.
9. Y. Tao and D. Papadias, "MV3R-tree: A spatio-temporal access method for timestamp and interval queries," Proc. of Int' l Conf. on Very Large Data Bases, 2001, pp. 431-440.
10. Ibrahim Kamel , Christos Faloustsos , "Parallel R-trees" , In Proceedings of ACM SIGMOD Conference on Management of Data, 1992, pp.195-204.
11. C. Faloutsos, P. Bhagwat, "Declustering using fractals", In proceedings of the 2nd International Conference on Parallel and Distributed Information systems. 1993, pp. 18-25.
12. Randeep Bhatia, Rakesh K. Sinha, and Chung-Min Chen. "Hierarchical declustering schemes for range queries". EDBT 2000, 7th International Conference on Extending Database Technology, 2000, pp.27-31.
13. S. Prabhakar, K. Abdel-Ghaffar, D. Agrawal, and A.El Abbadi, "Cyclic allocation of two-dimensional data", Proceedings of the Fourteenth International Conference on Data Engineering, 1998, pp. 94-101.
14. Jianzhong Li, Jaideep Srivastava, Doron Rotem, "CMD: A Multidimensional Declustering Method for Parallel Database Systems", 18th International Conference on Very Large Data Bases, 1992, pp. 3-14.
15. Hak-Cheol Kim, Ki-Joune Li,"Declustering Spatial Objects by Clustering for Parallel Disks", Database and Expert Systems Applications, 12th International Conference, 2001, pp. 450-459.
16. 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, 1999, pp. 68-77.
17. Youngduk Seo, Bonghee Hong, "Declustering of Trajectories for Indexing of Moving Objects Databases", Database and Expert Systems Applications, International Conference 2004, will be published.
18. T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects", GeoInformatica, Vol. 6, No. 2, 2002, pp. 153-162.

서영덕
2004년 부산대학교 대학원 컴퓨터공학과 박사
2005년 현재 경성대학교 컴퓨터과학과 초빙교수
관심분야: 지리정보 시스템, 병렬 지리정보 시스템, 객체 지향 데이터베이스

홍봉희
1982년 서울대학교 컴퓨터공학과 졸업 (공학사)
1984년 서울대학교 컴퓨터공학과 졸업 (공학석사)
1988년 서울대학교 컴퓨터공학과 졸업(공학박사)
1987년~현재 부산대학교 컴퓨터공학과 교수
2002년~현재 한국공간정보시스템학회 상임이사
2004년~현재 차세대 물류 IT 사업단 단장
관심분야: 데이터베이스, GIS, RFID 미들웨어, USN(유비쿼터스 센서 네트워킹)