이동체 데이터베이스를 위한 디클러스터링 정책

서 영 덕[†] · 홍 은 석[††] · 홍 봉 희[†††]

요 약

이동체 데이터베이스에서 이동체 궤적의 양은 엄청나게 많아서 기존의 단일 디스크 기반에서는 특정 영역의 질의에 대한 빠른 응답과 처리율의 향상을 볼 수 없다. 따라서 고성능 질의 처리를 위한 시스템의 성능 향상을 위해서는 병렬 처리 기법의 도입이 필요하다. 기존의 디클러스터링 방법에서는 시간이 지남에 따라 연속적으로 보고되는 이동체 특성을 고려하지 않고 있다. 그러므로 대용량 이동체 데이터에 대하여 고성능 질의 처리를 위한 새로운 디클러스터링 방법이 필요하다. 이 논문에서는 대용량 이동체 데이터베이스에 대한 고성능 질의 처리를 위한 새로운 디클러스터링 정책을 제시하였다. 이동체 데이터의 MBB(Minimum Bounding Box) 중 공간 좌표에 대한 근접성만을 고려하여 하나의 SD(SemiAllocation Disk)값을 설정하고 그 값과 시간 도메인을 다시 고려하여 근접성을 계산함으로써 디클러스터링을 한다. 또한 디스크 별 부하 균등화를 고려하여 보다 정확한 디클러스터링 효과를 가지도록 하였다. 이와 같이 이동체의 시공간 특성을 고려한 새로운 디클러스터링 정책으로 시스템의 성능을 향상 시킬 수 있다. 성능평가를 통해서 기존의 Round-Robin 방법보다는 5%, 10% 영역 질의 에서 평균 15% 정도의 성능향상을 보였으며 Spatial Proximity 방법보다는 평균 5%의 성능향상을 보였다.

Declustering Method for Moving Object Database

YoungDuk Seo[†] · EnSuk Hong[††] · BongHee Hong[†††]

ABSTRACT

Because there are so many spatio-temporal data in Moving Object Databases, a single disk system can not gain the fast response time and total throughput. So it is needed to take a parallel processing system for the high effectiveness query process. In these existing parallel processing systems, it does not consider characters of moving object data. Moving object data have to be thought about continuous report to the Moving Object Databases. So it is necessary think about the new Declustering System for the high performance system. In this paper, we propose the new Declustering Policies of Moving object data for high effectiveness query processing. At first, consider a spatial part of MBB(Minimum Bounding Box) then take a SD(SemiAllocation Disk) value. Second time, consider a SD value and time value which is node made at together as SDT-Proximity. And for more accuracy Declustering effect, consider a Load Balancing. Evaluation shows performance improvement of average 15% compare with Round-Robin method about 5% and 10% query area. And performance improvement of average 6% compare with Spatial Proximity method.

키워드 : 이동체 데이터베이스(Moving Object DB), 디클러스터링(Declustering), 병렬데이타베이스(Parallel Database), 공간데이타베이스(Spatial Database)

1. 서 론

차량 위치 추적 시스템은 GPS, 비콘, 루프검지기등의 차량 위치 탐지기로부터 전송받은 위치 정보를 토대로, 현재 차량의 위치 정보나 교통량의 흐름등을 실시간으로 제공해 주는 시스템이다. 이러한 이동체들은 자신의 위치 정보를 일정 주기를 가지고 각각의 서버로 전송하게 된다. 특히 이동 차량은 시간의 변화에 따라 그 위치가 연속적으로 변경되는 이동체의 특성을 가지므로, 이동체를 효율적으로 저장 및 관리하기 위한 연구가 필요하게 되었다. 지금까지 이동체의 연구는 데이터 모델링, 질의 표현, 인덱스 구조, 불확실성 처리 방법등으로 세분화되어 연구되었으며, 차량 위치 추적을 위한 관련 응용 시스템들이 개발되었다.

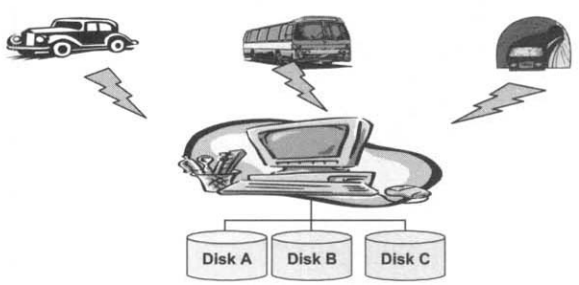
그런데 차량 위치 추적을 위해 지금까지 연구된 이동체 관리시스템들은 소규모 이동체를 대상으로, 단일 디스크 시스템에서 색인구조의 변경 등을 통한 효율성 향상에 많은 연구가 이루어 졌다. 그러나, 이동체들이 1분당 한번씩 자신의 위치를 관제 서버로 전송 한다면 하루 1,440번의 보고가 있다. 그리고 이동체의 수가 1000개만 되어도 하루 1,440,000이라는 엄청난 정보를 서버로 전송하게 된다. 즉 위치기반 서비스와 관련하여 각 서버 시스템으로 보내지는 이동체 데

※ 본 연구는 한국과학재단 지역대학 우수과학자 지원연구(R05-2003-000-10360-0) 지원으로 수행되었음.
† 정 회 원 : 부산대학교 대학원 컴퓨터공학과
†† 정 회 원 : 삼성전자 무선사업부 연구원
††† 정 회 원 : 부산대학교 컴퓨터공학과 교수
논문접수 : 2004년 8월 25일, 심사완료 : 2004년 10월 1일

이터의 양은 방대하다. 그리고 이동체의 데이터는 시간에 따라 증가하는 형태로 보고 되기 때문에 계속 증가하는 데이터의 양은 엄청 많으며 그에 대한 아주 많은 연산 처리가 필요하다. 즉 대용량 이동체의 데이터에 대한 수정, 검색 시 시스템의 빠른 처리에 대한 요구가 높아지고 있다.

기존의 성능 향상을 위한 방법은 메인 메모리 시스템, 빠른 디스크 활용 등의 다양한 방법으로 접근해 왔다. 그러나 이 논문에서는 기존의 단일 디스크 기반의 시스템 환경이 아닌 다중 디스크 기반의 디클러스터링 환경을 바탕으로 연구하였다. 즉 다중의 디스크를 병렬로 두고 데이터를 적절히 분산하여 배치함으로써 같은 I/O 시간으로 더 많은 데이터에 접근하여 빠른 응답 시간을 얻도록 하는 것이다. 만약 단일 디스크 기반의 시스템에 모든 데이터가 다 들어 있다고 가정한다면 결과를 검색하는데 있어서 순차적인 방법밖에 없을 것이다. 그러나 그러한 데이터를 여러 개의 디스크로 분산하여 저장하고 있다면 한번의 질의 수행시간에 각각의 디스크에 동시에 접근하여 읽어 들임으로써 보다 빠른 시스템을 구축할 수 있다. 이러한 디클러스터링 방법은 단일 디스크 기반의 시스템에서 있는 병목현상을 줄임으로써 빠른 응답 시간과 전체 처리율에서 성능 향상을 보이도록 하는 방법이다.

(그림 1)은 데이터의 양이 많은 이동체 데이터베이스 환경을 보여 주고 있다. 원하는 검색 결과 데이터가 많은 이동체 환경의 경우 이러한 디클러스터링 방법을 이용함으로써 보다 빠른 성능 향상을 보일 수 있다.



(그림 1) 이동체 서버 환경

그러나 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 이동체가 보고하는 데이터는 시간과 공간 모두에 관련성을 가지고 있어서 이들 모두를 함께 고려하는 것은 중요한 문제가 된다. 그러므로 위와 같은 문제점을 해결하기 위한 새로운 이동체 디클러스터링 방법의 제시가 필요하다.

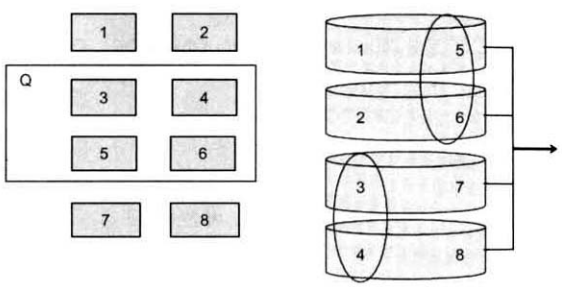
이 논문에서는 이동체에 대한 영역 질의 시 빠른 응답시간을 얻고 전체 시스템의 처리율 향상을 위한 SemiAllocation Disk with Time Proximity 방법을 제시한다. 즉 이 논문에서는 이동체 데이터의 공간 좌표를 이용한 공간 근접성과 시간 도메인을 함께 고려하여 새로운 디클러스터링 방법을 제안한다. 이렇게 함으로서 이동체 데이터에 대하여 시공간 도메인 모두를 고려할 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 디클러스터링에 관한 개요와 이에 대한 연구 결과를 소개하고, 3장에서는 디클러스터링의 기존 방법 적용시의 문제점을 기술한다. 4장에서는 이동체 데이터의 시공간 모두를 고려한 디클러스터링 방법을 설명하고 5장에서는 실험을 통한 성능평가를 분석한다. 마지막으로 6장에서는 결론 및 향후 연구에 대하여 기술한다.

2. 관련 연구

2.1 디클러스터링(Declustering) 개요

디클러스터링(Declustering)은 (그림 2)에서와 같이 한번의 질의에 대하여 같이 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 데이터 일수록 질의에 대한 결과 집합에 포함될 가능성은 더욱 커지는 것이다. 바로 이러한 데이터들을 서로 다른 디스크에 저장하고 같은 I/O 시간에 각각의 디스크에 동시에 접근 함으로써 응답시간과 전체 처리시간을 줄이고자 하는 방법이다.



(그림 2) 디클러스터링

예를 들어 (그림 2)에서는 왼쪽에 숫자로 표시된 각 공간 데이터가 순서대로 오른쪽의 각 디스크로 저장 되어 있다고 가정하자. 이때 영역 질의 Q가 요청 되면 각 디스크에서 검색을 하게 될 것이다. 여기서 해당되는 검색 결과인 3, 4, 5, 6번 영역의 공간은 각각 다른 디스크에 배치되어 있으므로 한번의 디스크 I/O 시간에 동시에 읽기가 가능해진다.

그러나 만약 위의 데이터가 모두 하나의 디스크에 저장되어 있다고 가정 한다면 3, 4, 5, 6을 읽어 들이는 네 번의 디스크 I/O 시간이 필요 할 것이다. 즉 디클러스터링 방법은 같은 질의 영역의 검색에 대하여 동시에 각 디스크에 접근함으로써 검색시간이 줄어질 수 있는 장점이 있다.

큰 영역의 질의에 대해서는 최소한의 I/O 시간으로 검색하며 작은 영역의 질의에 대해서는 한번의 I/O 시간으로 검색하고자 하는 것이 디클러스터링의 목적이 된다. 즉 데이터의 MinLoad와 UniSpread를 목적으로 하는 것이 디클러스터링이다.

2.2 기존의 디클러스터링 정책

기존의 디클러스터링 방법을 분류하면 일반적인 관계형 데이터베이스에서 사용되는 방법과 공간 데이터베이스에서 사용되는 방법의 두 가지 종류가 있다. 이 논문에서 다루고자 하는 방향은 공간 데이터베이스를 바탕으로 한다. 공간 데이터베이스에서는 일반적인 관계 형 데이터베이스와는 달리 데이터의 위상이나 점, 직선, MBR(Minimum Bounding Rectangle)과 같은 데이터를 다루고 있다. 이러한 차이점 때문에 공간 데이터베이스에서의 디클러스터링 방법은 좀더 다른 방법이 필요하게 되었다.

공간 데이터베이스에서 기존의 관련연구를 보면 수학적 heuristic한 방법과 개념적 heuristic한 방법으로 나눌 수 있다. 그 중 수학적 heuristic 방법은 Parallel R-Trees[3]를 이용한 Proximity Index[3]방법과 MVAS(Multi Version Access Structure)[5]를 이용한 KT-Proximity[2] 방법이 있다. 또한 개념적 heuristic 방법으로는 Round Robin, Minimum Area, Minimum Intersection, Hilbert Curves의 방법으로 나눌 수 있다.

수학적 heuristic 한 방법 중에서 Proximity Index[3] 방법은 새로 삽입되는 객체에 대하여 기존의 객체들과의 공간적 근접성(SP : Spatial Proximity)을 조사하고 높은 근접성을 가지는 객체들을 서로 다른 디스크에 저장하는 방법이다. 여기서 근접성이 높다고 하는 것은 한번의 질의에 대하여 동시에 읽혀질 가능성이 많기 때문에 서로 다른 디스크로 저장하도록 하는 것이다. 두 개의 사각형 R, S에 대한 근접성에 관한 공식은 다음과 같다.

|Q| : 전체 질의 수

|q| : 두 노드가 동시에 검색되는 수

Proximity(R,S) = Prob{ a query retrieves both R and S }

$$\frac{\#\ of\ queries\ retrieving}{total\ \#\ of\ queries} = \frac{|q|}{|Q|}$$

그러나 이 방법은 기본적으로 1차원적인 관련성만을 고려하고 있으며 2차원의 경우 1차원의 곱으로 해결하고 있다. 이러한 경우 연속적으로 증가하는 시간 도메인에 대한 고려가 되지 않는 문제점이 발생한다. 즉 X, Y 축에 대하여 0에서 1로 정규화 한 후에 그 데이터를 사용하여 근접성을 계산하고 있는 것이다.

이러한 방법은 계속적으로 증가하는 시간 도메인에 대하여는 적절히 정규화할 방법이 없어서 3차원 시공간 데이터에 대해서는 적절하게 적용하지 못하는 문제점을 가지고 있다. 즉 시간 도메인에 대한 고려가 없어서 이동체 데이터를 처리하기에는 부족한 면이 있다.

KT-Proximity 방법에서는 MVAS[5]를 사용하여 시간 도메인에 대해 기술하고 있다. 이 방법은 Key 값들 사이에서 시간에 따른 데이터의 변화들의 근접성을 계산하여 디클러스터링을 하는 것이다. 그러나 이 방법에서 Time은 시간에 따라 증가하는 시간 도메인을 고려한 것이 맞지만 Key는 공간 관련성이 없는 단순한 Key 값을 표현하고 있다. 즉 변하지 않는 Unique한 값이기 때문에 시공간 이동체 데이터에 대해서는 똑바로 적용하지 못하는 문제점을 안고 있다.

이상에서 제시한 기존의 공간 데이터베이스의 디클러스터링 방법들은 모두가 1, 2차원적인 면만을 고려하여 근접성(Proximity) 방법을 적용하고 있거나 단순한 계산을 통하여 데이터를 나누고 있다. 이러한 경우 2차원 공간과 계속적으로 증가는 시간 도메인 값을 가지는 시공간의 이동체 데이터에 대해서 그대로 적용하지 못하는 문제점을 가지고 있다. 3차원의 시공간 모두를 계산하지 못하고 디클러스터링을 함으로써 이동체 데이터의 특징을 고려하지 못하고 있는 것이다.

3. 기존 방법의 시공간 디클러스터링 적용 시 문제점

이 장에서는 2장에서 기술한 기존의 디클러스터링 방법을 시공간 데이터에 대하여 그대로 적용하였을 때 발생할 수 있는 문제점을 기술한다.

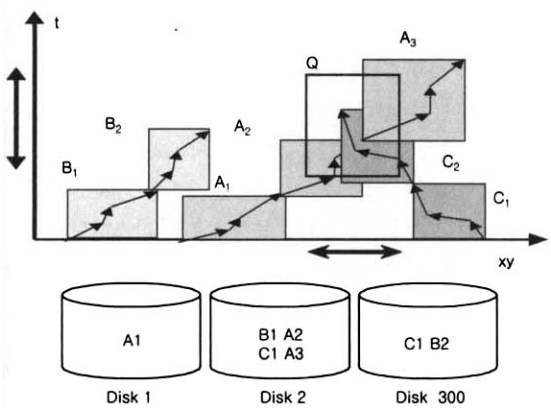
3.1 데이터의 한 디스크 집중 문제

이동체 데이터에 대해 기존의 근접성(Proximity) 방법을 적용하면 이동체의 특성인 시간 도메인을 고려하지 않는 문제점이 있다. 다차원에 대한 논의는 있지만 시간을 하나의 도메인으로 보지 않고 있다. 기존의 이러한 방법 적용은 1, 2차원 데이터에서는 적절할지 모르나 이동체 데이터에 대하여 적절한 디클러스터링이 이루어 지지 않는 문제점이 있다. 그 이유는 영역 질의란 시공간 모두에 대하여 이루어지는데 공간만으로 디클러스터링 한다면 시간 축으로는 전혀 고려되지 않고 데이터가 저장되었기 때문이다.

앞에서 설명 하였듯이 이동체 데이터는 시간, 공간 모두를 고려한 디클러스터링이 이루어져야 한다. 그러나 기존의 방법대로 시간 도메인에 대한 고려가 없거나 공간 도메인에 대

한 고려가 없는 디클러스터링 방법으로는 한 디스크에 데이터가 집중되는 등의 시공간 데이터에 대한 적절한 분산효과를 볼 수 없다. 이로 인하여 특정 영역에 대한 질의가 수행될 때 하나의 디스크에만 접근하는 부하 균등화 문제가 발생할 수 있는 것이다.

예를 들어, (그림 3)과 같이 이동체 데이터가 An → Bn → Cn 순서로 보고 된다고 가정하자. 이 그림은 SP(Spatial Proximity)[3]를 통하여 각 디스크로 데이터가 분산 저장되는 모습을 보여 주고 있다. 만약 A_3 데이터가 마지막으로 보고 된다면 이 데이터는 색인에서 가지는 자신의 형제 노드들과 공간 근접성인 SP를 검사하게 된다. 이때 A_3는 B_1 객체의 데이터와 가장 낮은 근접성을 보인다. 즉 A_3 데이터는 B_1 데이터와 가장 멀리 떨어져 있으며 Q와 같은 특정 영역의 질의에 대하여 동시에 읽혀질 가능성이 낮다는 것이다. 그러므로 A_3는 B_1이 저장되어 있는 디스크 2로 저장 될 것이다. 이와 같은 방법으로 모든 데이터가 (그림 3)의 디스크로 각각 저장될 것이다.



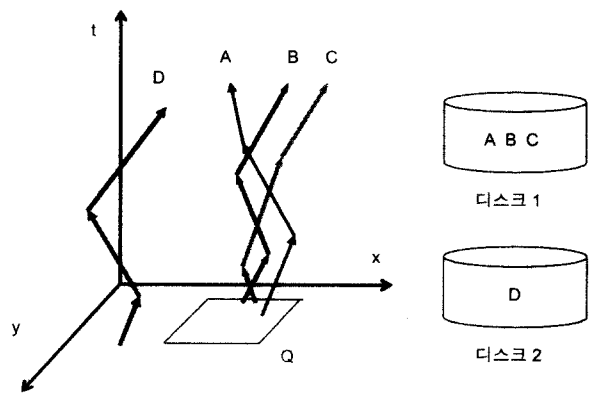
(그림 3) 이동체 데이터의 디클러스터링

이때 Q와 같은 임의의 특정 영역에 대하여 질의가 들어온다면 읽혀질 데이터는 A_2, C_2, A_3가 된다. 그러나 이 데이터들은 (그림 3)서 보듯이 하나의 디스크에 있으므로 결과적으로 세 번의 디스크 I/O가 발생한다. 이러한 결과는 디클러스터링의 MinLoad, UniSpread 목적에 맞지 않는 결과로 전체 시스템의 늦은 응답과 낮은 처리율을 보이게 된다. 만약 이 데이터가 적절히 분산되어 각각의 다른 디스크에 저장되어 있었다면 Q와 같은 질의에 대하여 한번의 I/O 시간으로 동시에 각 디스크에 접근하여 검색이 가능하게 될 것이다. 그러므로 시간 도메인에 대한 고려 없이 기존의 방법을 그대로 적용하는 것은 문제점을 안고 있다.

3.2 사향분포 데이터로 인한 문제

사향분포 데이터는 데이터의 분산이 한 곳에 집중하는 데이터를 말한다. 이러한 데이터는 실제 공간에서 흔히 있는 일로써 요즘의 도시 공동화로 인하여 자주 발생하는 것이다. 아침 시간대에는 특정 지역의 중심지에 데이터가 몰리며 저녁 퇴근 시간대에는 주택지에 데이터가 집중되는 것이다. 이러한 경우에도 하나의 디스크에 데이터가 집중되는 문제점이 발생한다.

예를 들어 (그림 4)와 같이 데이터가 입력 된다고 가정하자. 이 경우 입력 순서가 각 궤적의 라인 데이터 별로 A → D → B → D → C와 같이 된다면 Round-Robin 방법의 디클러스터링 일 경우 디스크 2에 D 궤적의 데이터가 집중될 것이다.



(그림 4) 사향분포 데이터의 디클러스터링

또한 기존의 공간 관련성만을 고려하는 SP(Spatial Proximity) 방법을 통하여 디클러스터링을 하더라도 D 궤적의 데이터는 항상 다른 궤적과의 근접성에 있어서 가장 낮은 값을 가지기 때문에 디스크 2에 데이터가 집중되는 문제점이 발생하게 된다.

4. SDT - 근접성(SemiAllocation Disk with Time Proximity)

3장에서 본 바와 같이 기존의 디클러스터링 방법은 시공간 모두를 고려한 디클러스터링 방법이 아니다. 이 논문에서 제시하는 SDT-근접성(SemiAllocation Disk with Time Proximity) 방법은 이동체 데이터 중에서 공간좌표만으로 근접성을 적용하여 가상의 값을 찾아낸 다음 그 값을 하나의 도메인 축으로 생각하고 시간 도메인과의 관련성을 다시 고려하는 방법이다. 이 장에서는 앞장에서 보인 문제점에 대하여 어떻게 효과적인 이동체 디클러스터링을 할 것인지 설명한다.

4.1 임시저장 디스크(SemiAllocation Disk)

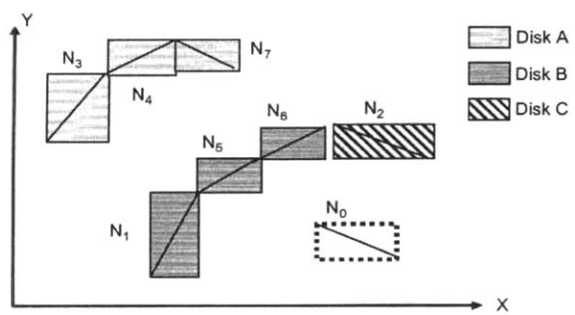
이동체 데이터 중에서 객체간의 공간 근접성인 SP(Spatial Proximity)만을 적용하여 나온 결과 디스크를 다음과 같

이 정의한다.

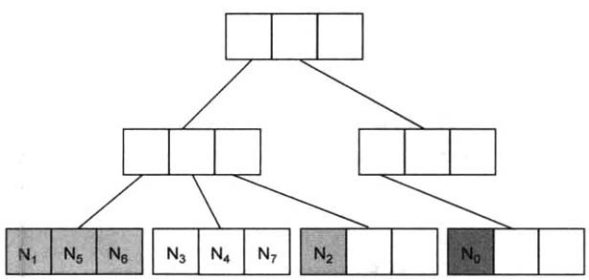
[정의 2] SemiAllocation Disk (SD) : 이동체 데이터의 MBR (Minimum Bounding Rectangle)에 대하여 Spatial Proximity (SP) 방법을 적용하여 설정된 가상 디스크

예를 들어 TB-tree[3]의 경우를 예로 들어 보자. 한 노드의 최대 객체 수를 3이라고 하고 (그림 5)에서와 같이 데이터가 분포 되어 있다고 가정하자. 이때 새로운 궤적 데이터 N_0가 삽입 되었다고 했을 때 기존의 궤적과는 다른 Trajectory를 가지기 때문에 새로운 노드를 형성하게 된다. 그때 기존의 존재하는 노드들과 Spatial Proximity를 적용하여 가장 근접성이 작은 디스크로 할당하는 것이다. 가장 근접성이 작은 디스크로 새로운 노드를 할당한다는 말은 가장 멀리 떨어져 있는 데이터의 디스크로 저장한다는 뜻이 된다. 이렇게 하는 이유는 한번의 질의에 대하여 비슷한 영역의 데이터는 같이 읽혀질 가능성을 가지고 있기 때문에 다른 디스크로 분산하는 것이다.

만약 같은 궤적의 데이터가 보고 된다면 한 노드가 다 채워 질 때까지 TB-Tree[8]에 삽입하고 노드가 다 찬 후 새로운 노드가 보고되는 시점에 다시 Spatial Proximity를 적용한다.



(그림 5) 새로운 객체의 공간 좌표가 N_0와 같이 보고 될 때



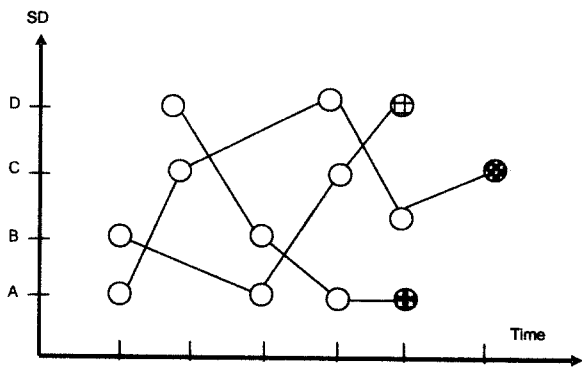
(그림 6) (그림 5)의 데이터를 TB-tree에 적용한 모습

이 예에서 보면 새로운 노드 N_0는 디스크 A에 할당 되는데 이렇게 결정된 디스크 A가 바로 SemiAllocation Disk 가 되는 것이다. 여기서 결정된 디스크인 SD(SemiAllocation Disk)는 다음 절에서 설명할 SDT-근접성에서 한 도메인 축을 형성하게 된다.

4.2 SDT - 근접성(SemiAllocation Disk with Time Proximity)

SDT-근접성(SemiAllocation Disk with Time Proximity)은 앞 절에서 정의한 SD (SemiAllocation Disk)와 시간 도메인을 함께 고려하는 근접성(Proximity) 방법이다.

SP(Spatial Proximity)방법으로 결정된 SD(SemiAllocation Disk)를 (그림 7)과 같이 한 도메인 축으로 설정하고 시간에 따른 각 노드의 입력 상황을 가상의 그래프로 형성한다. 그렇게 되면 각 이동체가 보고하는 시점에 따른 데이터의 입력은 (그림 7)과 같이 그려 질 수 있다. 이때 보고되는 포인트 한 개의 의미는 한 노드가 생성될 때 그 노드에 대하여 결정된 SD(SemiAllocation Disk)값이며 시간축의 의미는 그 노드가 생성된 시기를 뜻한다. 또한 각 포인트끼리 연결된 라인의 의미는 TB-tree[8]와 연관하여 하나의 궤적이 이동해 가는 모습을 보여 주고 있다.

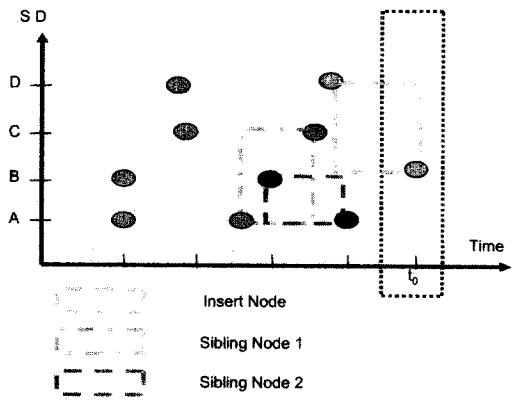


(그림 7) 이동체들이 시간에 따라 입력된 상태

(그림 7)과 같이 SD(SemiAllocation Disk)를 SDT-근접성(Proximity)의 한 축으로 설정하는 근거는 다음과 같이 들 수 있다. SD(SemiAllocation Disk) 값은 이동체 데이터 중에서 공간 차원만을 고려하여 근접성을 계산하였기 때문에 서로 관련성을 가지고 있으며 이 값 자체가 다른 노드들과의 근접성을 대표할 수 있다. 따라서 SD(SemiAllocation Disk)라고 정의되어 나온 디스크 값은 현재 삽입되는 객체의 공간 관련성을 충분히 반영하고 있다

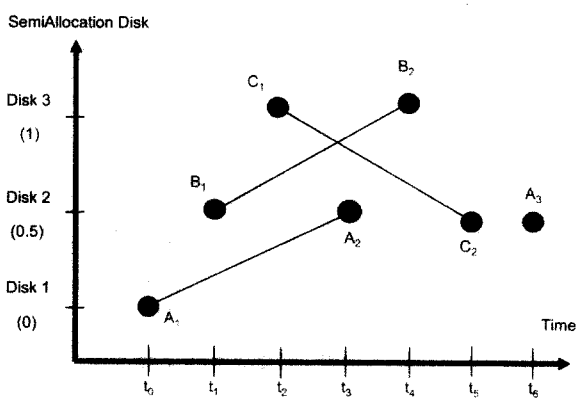
예를 들어 A, B, C 세 개의 디스크가 있다고 가정하고 제일 마지막에 보고된 노드를 $Node_t$ 라고 가정하면 기존의 데이터들과 공간 관련성을 비교한후 적절한 디스크를 선정하게 된다. 여기서 만약 디스크 B가 결정 되었다고 할 때 그 의미는 $Node_t$가 기존의 디스크 A, C에 있는 객체들과의 공간 관련성에서 가장 근접성(Proximity)이 작다는 것을 의미한다. 즉 $Node_t$에 대하여 결정된 임시저장 디스크(SemiAllocation Disk)는 충분히 이동체의 공간 관련성을 반영하고 있다.

여기서 SDT-근접성(Proximity)을 적용하는 방법은 (그림 8)과 같다.



(그림 8) 이동체가 t_0 시간에 마지막으로 보고를 한 경우

마지막 데이터의 노드가 t_0 시간에 보고되면 해당 노드가 색인을 검색하여 찾아 들어갈 부모 노드를 알게 될 것이다. 이때 그 형제 노드들과 다시 근접성을 계산하는 방법이다.



(그림 9) (그림 8)의 데이터를 SDT-근접성(Proximity) 적용

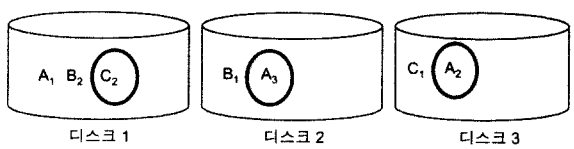
(그림 8)에서 t_0 시간에 새로운 노드가 삽입 되다고 가정하자. 새로운 노드는 색인을 Top Down 형태로 찾아 들어가다가 자신이 들어갈 단말 노드를 결정하게 된다. 이때 형제노드의 데이터와 다시 한번 더 근접성을 계산 하는 것이다. 형제 노드들은 바로 직전의 같은 궤적 노드를 찾아서 Rectangle을 형성하고 새로 삽입되는 노드도 바로 직전에 보고된 같은 궤적의 노드를 이용하여 Rectangle을 형성한다. 이러한 Rectangle 끼리 근접성을 다시 계산하여 새로운 노드의 삽입되는 디스크를 결정 하는 것이다. 이와 같은 방법으로 (그림 6)에서 보인 데이터의 결과를 보면 (그림 9)와 같이 된다.

먼저 A_1, B_1, C_1은 차례대로 디스크 1, 디스크 2, 디스크 3에 저장 되어있다고 가정하고 새로운 노드 A_2가 t_3 시간에 보고되었을 때를 고려해 보도록 하자.

A_2 노드는 이전 궤적의 노드인 A_1과 함께 Rectangle을 형성하고 기존의 노드들인 B_1, C_1과 근접성을 계산하게 된다. 우선 A_1-A_2 Rectangle 과 B_1 포인트와의 근접성은 시간 축으로 (t_3-t_1) 만큼 중첩 되어 있고 SD 축에 대하여는 거리가 없다. 따라서 시간 축에 대한 근접성 값은 Proximity 공식에 따라 $\frac{(1+2\times(t_3-t_1))}{3}$ 가 된다. 또 SD 축으로는 공식에 따라 $\frac{1}{2}$ 이 된다. 즉 위 두 식을 곱하여 A_2 노드에 대한 B_1과의 근접성 값을 구하면 $\frac{(1+2(t_3-t_1))}{9}$ 가 된다.

또 A_1-A_2 노드와 C_1과의 중첩성을 계산하여 보면 시간 축으로 t_3-t_2 만큼 중첩 되어 있고 SD 축으로는 0.5 만큼 떨어져 있다. 따라서 시간 축의 근접성 값은 공식에 따라 $\frac{(1+2(t_3-t_2))}{3}$ 이 되며 SD 축으로는 $\frac{(1-0.5)^2}{3}=\frac{0.25}{3}$ 가 된다. 즉 두 식을 곱하여 A_2 노드에 대한 C_1 과의 근접성 값을 구하면 $\frac{(0.25+0.5(t_3-t_2))}{9}$ 가 된다.

만약에 t_n의 시간 간격이 1초로 동일하다고 본다면 위 식에서 A_2, B_1 사이의 근접성 값은 $\frac{5}{9}$ 가 되고 A_2, C_1과의 근접성 값은 $\frac{0.75}{9}$ 가 된다. 즉 A_2는 B_1과 더 근접해 있으며 낮은 근접성 값을 가지는 C_1의 디스크 3으로 저장되는 것이다.

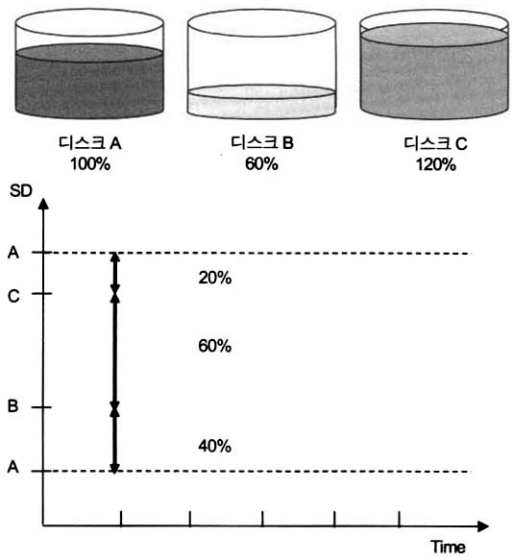


(그림 10) SDT-근접성(Proximity) 적용 후 디스크

마지막으로 A3가 입력 되었다고 했을 때 위와 같은 방법을 통하여 각 디스크 별로 디클러스터링 된 모습은 (그림 10)과 같다.

4.3 부하 균등(Load Balancing)

앞 절에서 정의한 SD(SemiAllocation Disk)는 공간좌표의 근접성만을 고려하였기 때문에 불균등 하게 디클러스터링 되는 문제점이 발생할 수 있다. 그러한 문제점을 미리 해결하기 위해 각 디스크에 저장될 데이터의 양으로 SD(Semi-Allocation Disk)축을 선정한다. (그림 11)은 데이터 분포에 따른 SD(SemiAllocation Disk)축의 변화를 보여주고 있다.



(그림 11) 데이터 분포에 따른 부하균등(Load Balancing)

하나의 디스크를 기준으로 다른 디스크의 데이터 양을 표시하고 그에 맞추어 SD(SemiAllocation Disk)축을 동적으로 구성하는 것이다.

예를 들어 (그림 11)을 보면 디스크 A를 기준으로 디스크 B의 데이터 양이 60% 만큼 있고 디스크 C의 양이 120% 있다. 즉 디스크 A를 기준으로 할 때 디스크 C와 디스크 B의 데이터 차이는 60%가 되며 이 값들이 SD(SemiAllocation Disk)축을 형성 할 때 디스크 A와 디스크 B, 디스크 C의 간격을 결정하게 되는 것이다.

이렇게 간격을 유지하는 이유는 SDT그래프에서 시간 축과 SD 축을 이용하여 다시 한번 근접성을 계산하기 때문이다. 이렇게 함으로써 특정 디스크에 데이터의 집중을 막고 더욱 효과적인 디클러스터링이 되도록 설정한다

이 장에서 제시하고 있는 방법에 대한 알고리즘은 다음과 같다.

```
/* SemiAllocation Disk 값을 반환하는 함수 */
Algorithm FindPD(R_MBB, S_MBB : Spatial MBB Data)
    Disk : Result SemiAllication Disk
    FOR ( R_MBB in Spatial Data of Live LeafNode )
        Disk = Min{Spatial_Proximity(R_MBB,S_MBB)}
    END
    RETURN Disk
END FindPD
/* 가장 작은 시공간 근접성값을 가지는 디스크를 반환하는 함수 */
Algorithm SDT-Proximity(R_NODE, S_NODE : NODE)
        Disk : Result of Confirmed Disk
        FOR ( R_NODE in All Sibling Record,)
        BEGIN
            S_NODE of Current Record )
          Disk = Min { Proximity(R_NODE, S_NODE) }
        END
END PDT-Proximity
```

5. 성능평가

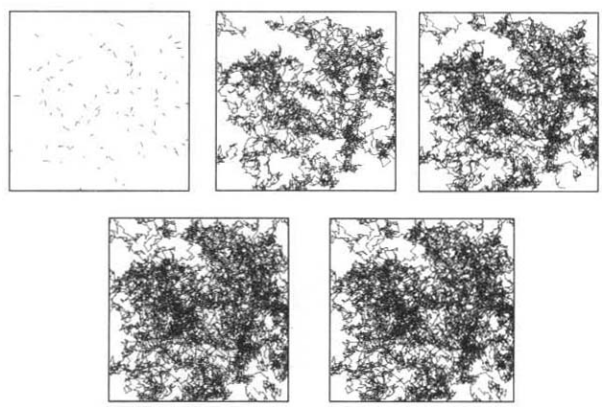
5.1 구현 및 실험 환경

이 논문에서 사용한 시스템은 윈도우 2000 Server와 256MB의 메인 메모리, CPU는 Pentium III 550Mhz Dual CPU, HDD 5개의 환경에서 실험 하였다. HDD는 SCSI 방법을 통하여 메인보드에 연결 되어 있다.

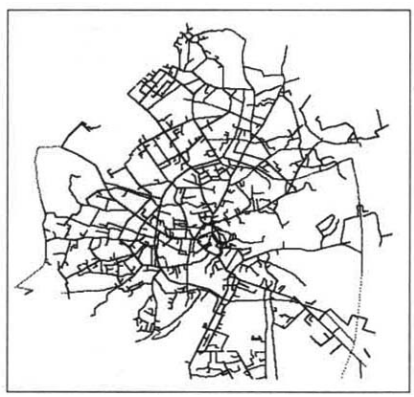
데이터는 GSTD[7]생성기를 통하여 (그림 15)와 같은 만개의 랜덤 데이터와 (그림 16)과 같은 십만 개의 가우시안 데이터를 생성하고 TB-tree[8]에 색인을 형성 하였다. 질의는 시공간 축에 5%, 10%, 20%의 영역질의 500개의 질의로 성능평가 하였다. 페이지 크기는 1024byte로 실험 하였다. 또한 버퍼의 크기를 200개로 고정하여 실험하였다.

〈표 1〉 실험 인자 테이블

실험	총 데이터 수	질의 영역	이동체 수	보고 회수
1	100,000개의 Line 데이터	5%	100개	1000번
2		10%		
3		20%		
4	10,000개의 Network Line 데이터	5%	100개	100번
5		10%		
6		20%		



(그림 12) 만개의 가우시안 분포의 데이터

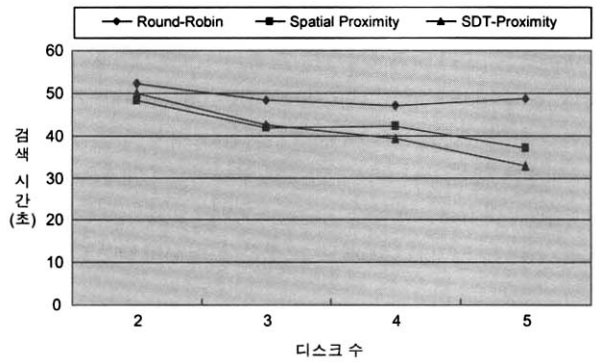


(그림 13) 만개의 네트워크 데이터

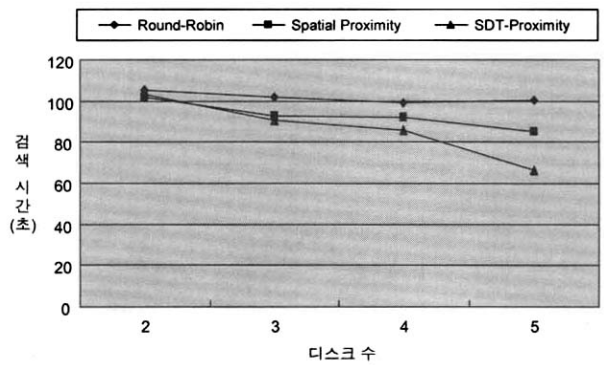
5.2 실험 결과

5.2.1 십만 개 데이터 검색 결과

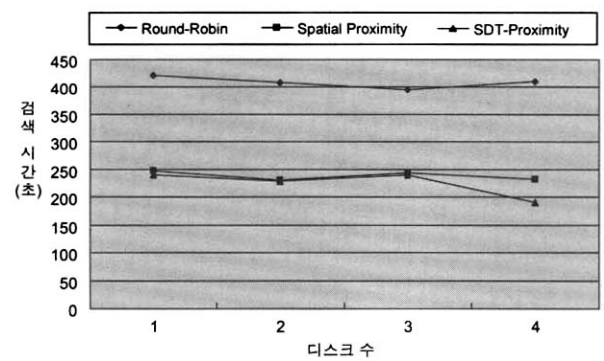
이 실험은 (그림 12)와 같은 가우시안 분포의 십만 개의 데이터를 사용하여 색인을 형성하고 3개의 질의를 사용하여 검색하였다.



(그림 14) 십 만개 데이터의 5% 영역 질의



(그림 15) 십 만개 데이터의 10% 영역 질의



(그림 16) 십 만개 데이터의 20% 영역 질의

이 결과는 십 만개의 가우시안 분포의 데이터를 사용하여 색인을 구축하였다.

(그림 14)는 한 축당 5%의 영역질의를 한 것으로 Round-Robin 방법 보다 Spatial Proximity 방법이 13.7%의 성능 향상을 보이고 있으며 SDT-근접성(Proximity) 방법은 Round-Robin 방법보다 16.1% , Spatial Proximity 방법보다는 2.8%의 질의 검색 속도의 향상을 보이고 있다.

(그림 15)에서는 10%의 영역 질의에 대한 결과를 보여주고 있다. Round-Robin 방법보다 Spatial Proximity 방법은 8.7%의 성능 향상을 보이며 이 논문에서 제안하는 SDT-근접성(Proximity) 방법은 Spatial Proximity 방법보다 6.7%, Round-Robin 방법보다 14.8%의 질의검색의 속도 향상을 보여준다.

(그림 16)은 20%의 영역 질의에 대한 검색 결과이다. 이 실험에서는 Round-Robin 방법에 있어서 낮은 성능을 보여주고 있다. Round-Robin 방법이 Spatial Proximity 방법보다 41.8%의 성능 저하를 보이고 있으며 SDT-근접성(Proximity) 방법 보다는 45%의 질의 검색 속도의 저하를 보이고 있다.

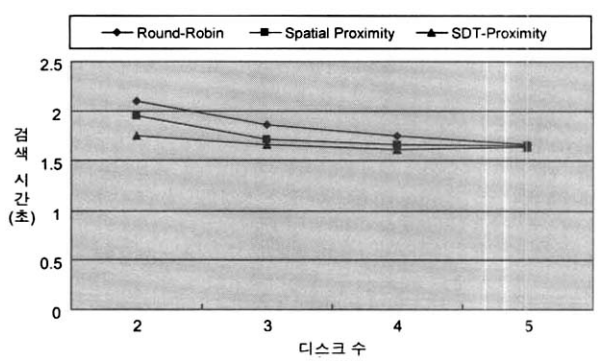
위의 실험에서는 디스크의 양이 증가함에 따라 디스크에 할당되어 있는 데이터의 검색 속도는 향상을 보고 있지만 만개의 색인 데이터의 경우 그 검색 속도가 디스크 수에 정비례 하여 감소하지는 않고 있다. 어느 정도의 디스크 수에서 비슷한 검색 결과를 보여주고 있는 것이다.

5.2.2 만개의 네트워크 데이터

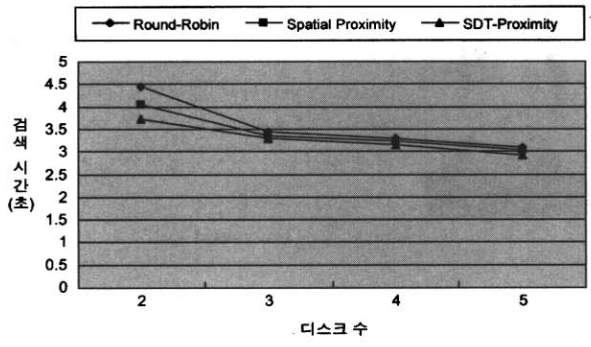
이 실험은 (그림 13)과 같은 네트워크 분포의 만 개의 데이터를 사용하여 색인을 형성하고 질의를 수행하였다.

(그림 17)을 보면 5%의 영역 질의에서 논문에서 제안한 SDT-근접성(Proximity) 방법이 Round-Robin 방법 보다 평균 9.5% , Spatial Proximity 방법보다 4.2%의 검색 성능향상을 보이고 있다. 또한 (그림 26)를 보면 10%의 영역 질의에서 Round-Robin 방법 보다 평균8.3% , Spatial Proximity 방법보다 4.2%의 검색 성능 향상을 보이고 있다. (그림 27)은 20%의 영역 질의에서 Round-Robin 보다 9.6%, Spatial Proximity 방법보다 4.5%의 검색 성능 향상을 보이고 있다.

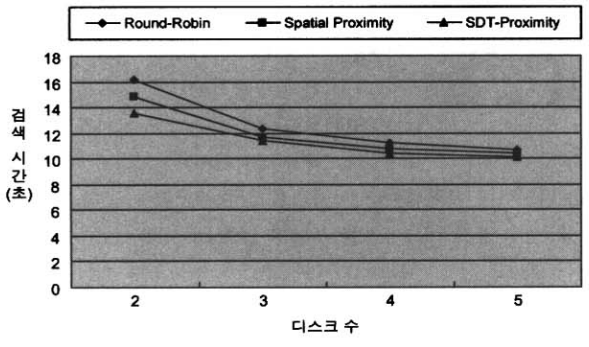
위의 실험 결과를 볼 때 이 논문에서 제안하고 있는 SDT-근접성(Proximity) 방법은 기존의 공간만을 고려한 디클러스터링 정책인 Spatial Proximity 방법보다 평균적으로 5% 정도의 검색 성능의 향상을 보이고 있다. 또한 가장 단순한 디클러스터링 분배 방법인 Round-Robin 방법보다는 10~15% 정도의 성능 향상을 보여주고 있다.



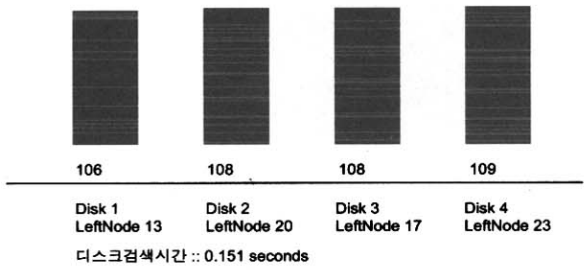
(그림 17) 네트워크 데이터의 5%의 영역 질의



(그림 18) 네트워크 데이터의 10%의 영역 질의



(그림 19) 네트워크 데이터의 20%의 영역 질의



(그림 20) 데이터에 따른 디스크 분포

이에 따른 (그림 20)에서는 질의 결과 데이터의 각 디스크별 분포를 보여주고 있다. 비교적 균등한 데이터의 분포를 나타내고 있다.

6. 결론 및 향후 연구

이 논문에서는 다중 디스크 기반의 서버환경에서 이동체가 자신의 위치를 일정한 주기로 서버에 보고하는 환경을 바탕으로 하고 있다. 이러한 이동체 데이터는 공간 데이터와 시간 데이터를 함께 보고하고 있다. 이러한 환경에서 이동체의 3차원 데이터를 어떻게 적절히 디클러스터링 할 것인가에 대한 새로운 방법을 제시하였다.

기존의 관련 연구를 살펴보면 이동체의 시공간 특징을 고려하지 않고 공간만 고려 하거나 시간 도메인만을 고려하여 디클러스터링을 하였다. 이러한 방법은 이동체 데이터의 시공간 특징을 다 살리지 못하고 디클러스터링 함으로써 특정한 개의 디스크에 데이터가 집중되는 문제점이 발생하였다. 이러한 문제점은 영역 질의 시 하나의 특정 디스크에 계속 접근 하여 낮은 응답 속도와 전체 처리율에 있어서 좋지 못한 결과를 보였다.

이러한 문제점은 데이터의 적절한 분산을 통하여 작은 영역의 질의 시, 한번의 디스크 I/O 와 큰 영역의 질의 시 최소한의 디스크 I/O를 통하여 데이터에 접근하고자 하는 디클러스터링 정책에 맞지 않은 결과를 보였다.

이 논문에서 제안한 SDT-근접성(Proximity) 방법은 이동체 데이터의 시공간 모두를 고려하고 있다. 우선 3 차원 이동체 데이터 중 시간을 제외한 공간 관련성만을 고려하여 근접성이 가장 작은 디스크를 SD(SemiAllocation Disk)로 설정한다. 여기서 설정된 SD(SemiAllocation Disk)와 시간 도메인을 다시 고려하여 근접성을 계산하고 실제 저장 되어야 할 디스크를 결정하는 방법이다. 또한 각 디스크 별 데이터의 양에 따른 적절한 부하 균등화를 통하여 데이터를 분산 함으로써 특정 영역의 질의 시 빠른 응답 시간과 전체 처리율을 향상시킬 수 있었다.

향후 연구로 더 다양한 디클러스터링 정책의 연구와 좀 더 많은 데이터와 실험 요소로 다양한 환경에서 실험을 적용해 보아야 할 것이다.

참 고 문 헌

[1] Sanjiv Behl, Rakesh M. Verma, "Efficient Declustering Techniques for Temporal Access Structures," Proceedings of the 1991 ACM SIGMOD Conference, pp.436-445, 1991.

[2] Bernhard Seeger, Per-Ake Larson, "Multi-Disk B-trees," ACM SIGMOD Conference, pp.436-445, 1991.

[3] Ibrahim Kamel, Christos Faloustsos, "Parallel R-Trees," Proxeedings of the 1992 ACM SIGMOD, pp.195-204, 1992.

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

[5] Peter J. Varman and Rakesh M. Verma, "An Efficient Multiversion Access Structure," IEEE TRANSACTION ON KNOWLEDGE AND DATA ENGINEERING, pp.391-409, 1997.

[6] HV Jagadish, "Analysis of the hilbert curve for representing two-dimensional space," Information Processing Letters, pp.17-22, 1997.

[7] Yannis Theodoridis, Jefferson R. O. Silva, Mario A. Nascimento, "On the Generation of Spatio-temporal Datasets," In Proceedings of the 6th Int'l Symposium on Large Spatial Databases, 1999.

[8] Dieter Proser, Christian S. Jensen, Yannis Theodorisdis, "Novel Approaches to the Indexing of Moving Object Trajectories," In Proc, of the 26th Int, 1 Conference on VLDB, pp.395-406, 2000.

[9] 홍은석, 서영덕, 홍봉희, "이동체 데이터의 근접성을 이용한 시공간 디클러스터링 방법", 한국정보과학회 2003 봄 학술발표논문집, 제30권 제1호, pp.767-769, Apr., 2003.

[10] 홍은석, 서영덕, 홍봉희, "이동체 데이터베이스에서 시공간 근접성을 고려한 디클러스터링정책", 한국정보과학회 2004 가을 학술발표논문집, 제30권 제2호, pp.118-120, Oct., 2003.

서 영 덕

e-mail : ydseo@pusan.ac.kr
1997년 부산대학교 컴퓨터공학과(공학사)
1999년 부산대학교 컴퓨터공학과(공학석사)
2004년 부산대학교 컴퓨터공학과(공학박사)
관심분야 : 지리정보시스템, 모바일 GIS, 병렬데이타베이스, 이동체 색인, 병렬색인

홍 은 석

e-mail : eshong@pusan.ac.kr
2002년 부산대학교 컴퓨터공학과(학사)
2004 년 부산대학교 컴퓨터공학과(석사)
현재 삼성전자 무선사업부 연구원
관심분야 : 데이터베이스, 공간 색인, Embedded OS

홍 봉 희

e-mail : bhhong@pusan.ac.kr
1982년 서울대학교 전자계산기공학과(공학사)
1984년 서울대학교 대학원 전자계산기공학과(공학석사)
1988년 서울대학교 대학원 전자계산기공학과(공학박사)
1988년~현재 부산대학교 공과대학 컴퓨터공학과 교수
관심분야 : 병렬공간 이동체 개방형 DB, DB, GIS