ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

영역 모니터링 질의 처리를 위한 공간 분할 기법☆

A Spatial Split Method for Processing of Region Monitoring Queries

정 재 우[1] 정 하 림[1] 김 응 모[1]*
Jaewoo Chung HaRim Jung Ung-Mo Kim

요 약

본 논문은 영역 모니터링 질의를 효율적으로 처리하는 기법에 대해서 다룬다. 기존의 영역 모니터링 질의 처리를 위해서 사용된 중앙 집중식 기법은 이동 객체가 서버에 주기적으로 자신의 위치 업데이트를 전송하고, 서버가 질의 결과를 지속적으로 업데이트 한다고 가정한다. 그러나 이러한 가정은 많은 양의 위치 데이터 전송으로 인해 시스템 성능을 크게 저하시킨다. 최근, 영역 모니터링 질의 처리를 위한 몇 가지 분산 기법들이 제안되었다. 분산 기법에서 서버는 각 이동 객체에게 i) 작업 공간의 서브 공간인 상주 도메인과 ii) 몇 개의 인접 질의 영역을 할당한다. 각 이동 객체는 상주 도메인을 벗어나거나 질의 영역의 경계를 가로지를 경우에만 서버에게 자신의 위치를 전송한다. 상주 도메인 및 인접 질의 영역을 이동 객체에 할당하기 위해서 서버는 작업 공간을 반복적으로 동일하게 반으로 분할하여 생성되는 질의 색인 구조를 사용한다. 하지만 이와 같은 색인 구조는 불필요한 분할이 발생하게 되므로 시스템의 성능 저하를 발생시킨다. 본 논문에서는 불필요한 분할을 줄이기 위해서 적응 분할 기법을 제안한다. 적응 분할 기법은 i) 질의 영역과 결과 서브 공간의 공간적 관계와 ii) 질의 영역의 분포를 고려하여 동적으로 작업 공간을 분할한다. 본 논문에서는 기존의 색인 구조인 QR-tree에 본 논문에서 제안한 새로운 분할 기법을 적용하였으며, 시뮬레이션을 통해 제안 된 분할 기법의 효율성을 검증했다.

☞ 주제어 : 영역 모니터링 질의, 색인, 공간 분할 기법

ABSTRACT

This paper addresses the problem of efficient processing of region monitoring queries. The centralized methods used for existing region monitoring query processing assumes that the mobile object periodically sends location-updates to the server and the server continues to update the query results. However, a large amount of location updates seriously degrade the system performance. Recently, some distributed methods have been proposed for region monitoring query processing. In the distributed methods, the server allocates to all objects i) a resident domain that is a subspace of the workspace, and ii) a number of nearby query regions. All moving objects send location updates to the server only when they leave the resident domain or cross the boundary of the query region. In order to allocate the resident domain to the moving object along with the nearby query region, we use a query index structure that is constructed by splitting the workspace recursively into equal halves. However, However, the above index structure causes unnecessary division, resulting in deterioration of system performance. In this paper, we propose an adaptive split method to reduce unnecessary splitting. The workspace splitting is dynamically allocated i) considering the spatial relationship between the query region and the resultant subspace, and ii) the distribution of the query region. We proposed an enhanced QR-tree with a new splitting method. Through a set of simulations, we verify the efficiency of the proposed split methods.

☞ keyword : region monitoring query, indexing, spatial split method

1. 서 론

스마트 기기의 대중화를 통해서 GPS(Global Positioning System)를 기반으로 하는 위치 기반 서비스(Location Base Service)가 사용자들에게 꾸준히 관심을 받고 있다.

위치 기반 서비스는 사용자의 위치를 기반으로 주변의 가치 있는 정보를 제공하는데, 모바일 광고 서비스나 위험 알림 서비스 등이 위치 기반 서비스의 대표적인 예이다. 위치 기반 서비스의 소비가 증가하면서 사용자들에게 효과적으로 정보를 제공하기 위한 다양한 연구가 진행

1 College of Information and Communication Engineering, Sungkyunkwan University., Suwon, 16419, Korea
* Corresponding author (umkim@skku.edu)
[Received 27 August 2017, Reviewed 24 September 2017, Accepted 23 November 2017]
☆ This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2015R1D1A1A01057238, NRF-2016R1D1A1B03931098)

중이다. 그 중 영역 모니터링 질의(Region Monitoring Query)는 중요한 역할을 담당한다. 예를 들어, 아동 모니터링 서비스를 제공한다고 가정했을 때, 부모는 특정 영역을 안전(위험) 영역으로 할당해서 자녀가 영역의 안(밖)으로 이동하면 알림을 받을 수 있다. 또 한, 여러 업체들이 모바일 광고를 전송할 영역을 할당해서 영역 밖에 있는 고객에게 불필요한 광고를 전송하지 않음으로써, 통신 비용 절약이 가능하다.

본 논문에서 우리는 영역 모니터링 질의의 효율적인 처리 기법에 대해서 연구한다. 이동 객체(moving object) 집합 O에 대한 영역 모니터링 질의 q는 주어진 공간 질의 영역(spatial query region) q.R에 현재 위치하는 이동 객체 Ó (⊆ O) 를 반환하고 이동 객체 Ó가 q.R의 안과 밖으로 이동할 때 계속해서 업데이트를 해준다.

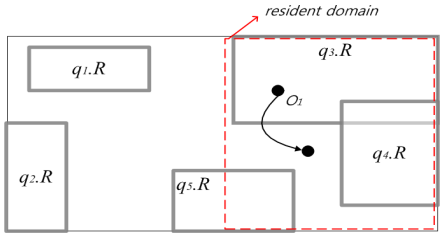
이러한 이동 객체에 대한 영역 모니터링 질의 처리는 크게 두 가지 범주로 구분을 할 수 있는데 질의 영역이 정적인 경우와 동적인 경우이다[1]. 본 논문은 정적인 질의 영역에 대해서 다룬다. 기존에 주로 사용 된 정적 영역 모니터링질의 처리는 이동 객체가 무선 네트워크를 통해서 서버에게 위치 업데이트를 주기적으로 전송하고, 서버는 질의의 결과를 계속해서 갱신하는 방식이었다.

하지만 모바일 시장의 성장과 함께 위치 업데이트는 과거와 비할 수 없이 거대한 양이 빠르게 발생하고 있으며, 이는 서버에 병목 현상을 일으키고 전체적인 수행 능력을 저하시킨다는 문제점과 이동기기의 급격한 배터리 소모를 발생시킨다.

영역 모니터링 질의를 효율적으로 처리하기 위해서는 무선 통신 비용을 최소화하고, 서버의 CPU 비용을 최소화해야 한다. Monitoring query management(MQM)[2]와 Query region-tree(QRT)[3]는 위의 문제를 해결하기 위한 분산 기법으로, 이동 객체의 연산능력을 이용해서 서버의 작업량과 통신비용을 감소시킨다.

일반적으로 서버는 각각의 이동 객체 o에게 전체 작업 공간에서 사각형 형태의 서브 공간인 상주 도메인(resident domain)을 할당하고 o의 상주 도메인에 겹쳐있는 모든 질의 영역을 할당한다. o에 할당되는 상주 도메인의 크기는 o의 연산 능력인 o.Cap에 의해서 결정된다. o.Cap=n 일 때, o에게 할당 된 상주 도메인은 반드시 o를 포함해야 하며 최대 n개의 공간 질의가 포함 될 수 있다.

예를 들어, 그림 1에서 o_1.Cap의 연산 능력이 3이라고 했을 때, o_1은 점선으로 이루어진 상주 도메인과 상주 도메인에 겹쳐있는 공간 질의 영역(q_3.R, q_4.R, q_5.R)을 할당받게 된다. o_1이 상주 도메인에 할당 된 질의 영역 밖으로 나가거나 상주 도메인에서 나가게 된다면, 질의 결과를 갱신하거나 새롭게 상주 도메인을 할당해주는 작업이 이루어진다. 즉, 그림 1에서와 같이 o_1은 질의 영역 q_3.R에서 빠져나왔기 때문에 서버는 질의 결과를 갱신하게 된다.



(그림 1) 상주 도메인의 예
(Figure 1) An example of resident domain

MQM과 QRT는 이동 객체의 이동을 주기적으로 갱신하지 않아도 되기 때문에 위치 업데이트 스트림을 획기적으로 줄일 수 있다. 이동 객체에 인접한 질의 영역과 함께 상주 도메인을 할당하기 위해서 MQM과 QRT는 BP-tree (binary partitioning tree)와 QR-tree (query region tree)를 사용하는데, 전체 작업 공간을 반복적으로 분할하여 나누어진 서브 공간들이 트리의 노드에 해당하게 된다. 그리고 각각 서브 공간과 겹치는 질의 영역의 정보에 대해서 저장하게 된다.

하지만 위의 두 가지 색인 구조는 전체 공간을 동일한 크기의 부분 공간으로 분할하는 중심 분할 기법을 사용하기 때문에 과도한 양의 불필요한 분할을 발생시키게 되며, 다음과 같은 두 가지 이유로 시스템 성능 저하가 일어나게 된다.

- 첫째, 서브 공간의 평균 크기가 작아지게 된다. MQM과 QRT에서 상주 도메인은 트리 노드에 해당하는 서브 공간이기 때문에 각 이동 객체에 할당되는 상주 도메인의 크기가 작아지게 된다. 결과적으로 이동 객체는 새로운 상주 도메인을 할당 받기 위해서 위치 업데이트를 서버에 자주 보내게 된다.
- 둘째, 트리의 높이가 불필요하게 높아지기 때문에 검색 및 삽입, 삭제 작업에 많은 비용과 시간을 필요로 한다. 이는 i) 서버에서의 CPU 비용과 ii) 질의 처리 시간이 길어지게 된다.

본 논문에서는 위와 같은 문제를 해결하기 위해서 불필요한 분할 횟수를 줄이는 새로운 분할 기법을 제안한

다. 새로운 분할 기법은 작업 공간을 동적으로 나누는 방법으로써, i) 결과 서브 공간과 질의 영역 사이의 공간 관계 및 ii) 질의 영역의 분포를 고려한다. 제안 된 분할 기법은 BP-tree와 QR-tree 모두에 적용 될 수 있지만, QR-tree가 BP-tree 보다 우수하기 때문에 본 논문에서는 QR-tree를 기본 색인 구조로 본다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 QRT의 시스템 모델과 QR-tree를 중심으로 관련 연구를 조사한다. 3장에서는 QR-tree가 적용 된 분할 기법에 대해서 자세히 다룰 것이며, 4장에서는 서버의 무선 통신 비용과 CPU 비용 측면에서의 성능평가를 통해 제안 된 분할 기법의 효율성 검증을 한다. 마지막으로 5장에서는 결론을 제시한다.

2. 관련 연구

위치 기반 서비스의 발달과 함께 영역 모니터링 질의(이동 객체에 대한 처리)를 처리하는 많은 연구가 수행되었었다. 이러한 연구들은 질의 영역이 정적인지 동적인지에 따라서 크게 두 가지 범주로 나누어지는데 첫 번째 범주는 이동 객체에 대한 정적 질의 영역에 대해서 다루며[2,3], 그 외의 범주는 이동 객체에 대한 질의 영역 이동에 대해서 다룬다[4,5]. 본 논문은 첫 번째 범주에 대해서 다루고 있기 때문에 정적 질의 영역에 대한 검토 후에 i) QRT의 시스템 모델과 ii) QR-tree에 대해서 자세하게 설명한다.

영역 모니터링 질의는 긴 시간동안 활성 된 상태로 유지 되어야 하며 질의 영역이 정적이기 때문에 질의를 색인(이동 객체 대신)하는 기법을 주로 사용한다. 대표적으로 Prabhakar는 디스크 상주 R-tree를 사용하여 질의를 색인하였고[2] Kalashnikov는 메모리 내의 격자 색인 구조를 사용하였다[4]. 두 가지 방법 모두 이동 객체가 주기적으로 위치 업데이트를 서버로 보낸다는 가정을 갖고 있다. 이 때, 서버는 지속적으로 i) 위치 업데이트 스트림을 수신하고 ii) 영향을 받게 되는 질의를 선택하며 (iii) 결과들을 업데이트하게 된다. 그러나 이동 객체가 많아질수록 거대한 양의 위치 업데이트 스트림이 발생하고 서버의 무선 통신 비용과 CPU 비용이 지속적으로 (영향을 받은 질의를 결정하고 그 결과를 업데이트하기 위해) 크게 증가하게 된다.

이러한 이동 객체가 보내는 위치 업데이트 수를 줄이기 위해서 많은 연구가 진행이 되었고 MQM과 QRT가 제안 되었다. MQM과 QRT는 이동 객체의 기능을 활용하여 서버의 무선 통신 비용과 CPU 비용을 줄이는 것에 목적을 두고 있다. 또 한, 인접한 질의 영역과 적절한 상주 도메인을 이동 객체에 할당하기 위해 MQM과 QRT는 BP-tree와 QR-tree를 각각 사용하는데 본 연구에서는 기본 색인 구조를 QR-tree로 고려하기 때문에 QRT의 시스템 모델을 설명하고 QR-tree를 설명한다.

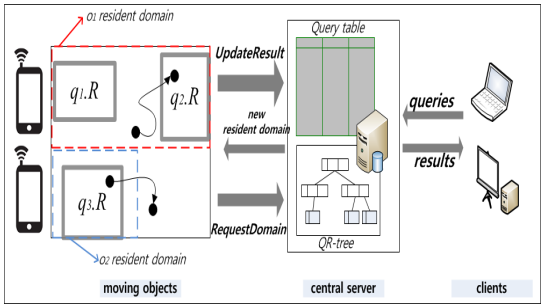
2.1 QRT(Query Region Tree) 시스템 모델

QRT의 중점적인 목표는 상주 도메인의 개념을 활용해서 1장에서 언급 된 두 가지 조건을 만족시키는 것이다. 그림 2는 이동 객체, 클라이언트 및 중앙 서버로 이루어진 시스템 모델의 개요를 보여준다.

- 이동 객체(Moving objects): 각 이동 객체 o는 현재의 위치를 알고 있고 한 번에 질의 영역 개수를 최대로 처리할 수 있는 연산 능력인 O.Cap을 갖고 있다. 이동 객체는 두 가지 유형(RequestDomain, UpdateResult)의 위치 업데이트 메시지를 서버로 전송한다. RequestDomain은 새로운 상주 도메인(새로운 질의 영역을 포함)을 수신하기 위한 메시지이며, UpdateResult는 서버가 질의 결과를 업데이트하도록 하는 메시지이다. 예를 들어, 그림 2의 이동 객체인 o_2는 상주 도메인을 벗어나기 때문에 새로운 상주 도메인을 수신하는 RequestDomain을 서버로 전송하게 된다. 또 한, 그림 2의 이동 객체 o_1은 q_2.R의 질의 영역으로 들어왔기 때문에 UpdateResult 메시지를 전송하게 된다.
- 클라이언트(Client): 각 클라이언트는 서버에 여러 영역 모니터링 질의를 발행하고 무선 또는 유선 네트워크를 통해 서버에서 질의 결과를 지속적으로 받는다. 클라이언트와 이동 객체는 서로 직접 통신하지 않고 서버를 통신 수단으로 활용하게 된다. 클라이언트가 발행한 질의 q는 고유 식별자로 식별되며 그것의 질의 영역인 q.R은 정적인 직사각형 형태이다.
- 중앙 서버(Central server): 서버는 질의 테이블과 QR-tree를 관리하고 다음 세 가지 작업을 수행한다.
  - -영역 모니터링 등록(등록 취소): 클라이언트가 질의 q를 요청할 때, 서버는 q에게 식별자를 할당하고, q를 질의 테이블에 삽입하며, QR-tree에 삽입한다. 그 후 모든 이동 객체에 새로운 질의가 발행 됐다는 것을 브로드캐스팅하게 된다.
  - -도메인 할당: 서버가 이동 객체로부터 RequestDomain 메시지를 받으면 QR-tree를 사용하여 o의 새로운 상

주 도메인을 결정하고 o에게 적합한 상주 도메인을 새로운 질의 영역과 함께 할당한다.

-질의 결과 업데이트: 서버가 이동객체 o에게 UpdateResult 메시지를 수신할 때 서버는 질의 결과를 업데이트해야 한다. 즉, 메시지를 수신 했을 때 질의 결과가 o를 포함하고 있는지 확인하고 포함 된다면 업데이트를 진행하게 된다.



(그림 2) 시스템 모델 개요

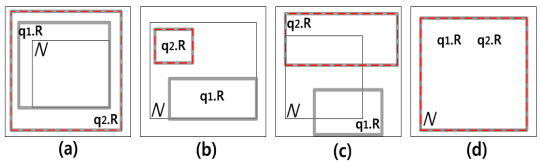
(Figure 2) Overview of system model

2.2 QR(Query Region)-tree

QR-tree는 전체 작업 공간을 같은 크기의 서브 도메인으로 재귀 분할하여 구축되는 질의 색인 구조이다. 루트 노드에 해당하는 질의 영역들이 주어졌을 때, 질의 영역의 수가 분할 임계값 θ보다 클 경우에는 작업 공간은 루트 노드의 자식 노드 N에 해당하는 두 개의 서브 공간으로 분할된다. 이러한 과정은 질의 영역이 모든 도메인에 대해 coverd_by 관계 또는 partially_intersect 관계를 만족하는 질의의 개수가 θ보다 작을 때까지 진행한다. QRT는 질의 영역 q.R과 서브 공간 N 사이의 중첩 관계를 4개의 카테고리로 분류한다.

정의 1. 질의 영역 q.R과 서브 공간(QR-tree 노드) N이 주어졌을 때, 아래와 같은 4가지의 겹침 관계가 존재한다.

- cover 관계(그림 3(a)): (q.R∩N=¬∅)∧(q.R-N=¬∅)∧(N-q.R=¬∅)일 때, q.R은 N을 포함한다.
- cover_by 관계(그림 3(b)): (q.R∩N=¬∅)∧(q.R-N=∅)∧(N-q.R=¬∅)일 때, q.R은 N에 포함된다.
- partially_intersect 관계(그림 3(c)): (q.R∩N=¬∅)∧(q.R-N=¬∅)∧(N-q.R=¬∅)일 때, q.R은 N과 부분적으로 겹침을 의미한다.
- equal 관계(그림 3(d)): (q.R∩N=¬∅)∧(q.R-N=∅)∧(N-q.R=∅)일 때, q.R은 N과 같다.



(그림 3) 겹침 관계 분류

(Figure 3) Classification of the overlap relationship

QR-tree의 단말 노드는 θ개의 질의 식별자를 저장하며, 각 질의 식별자는 질의 테이블의 질의 q를 나타낸다. 비단말 노드들은 두 개의 엔트리 <ptr, N>을 저장한다. 여기서 ptr은 자식 노드에 대한 포인터이고 N은 ptr이 가리키는 자식 노드의 서브 공간이다. 또 한, QR-tree는 아래와 같은 4가지 조건을 만족시켜야한다.

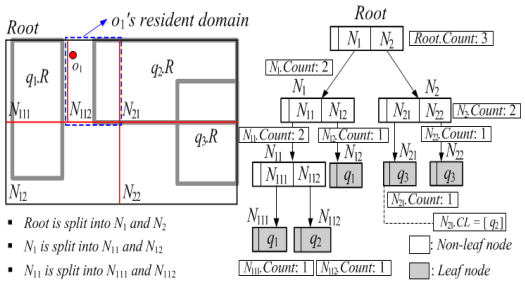
- 질의 q의 질의 식별자 qid는 q.R이 단말 노드 N에 의해 cover_by 되거나 partially_intersect 하는 경우에만 단N에 저장한다. q.R이 N과 겹쳐도 q.R이 N을 cover 하거나 equal하는 경우에는 qid는 N에 저장하지 않는다.
- 질의 q의 질의 식별자 qid는 q.R이 단말 노드들과 partially_intersect 관계일 때, 여러 단말 노드를 중복해서 저장한다.
- 비단말 노드 N에 저장 된 엔트리(ptr,Ń)에서 Ń는 N을 균등하게 절반으로 나눈 서브 공간이다.
- 모든 노드는 변수 Count를 추가로 저장하고 CL(Covering list)라는 특수 리스트와 연관된다.

정의 2: Count: 질의 집합 Q와 QR-tree의 노드 N이 주어지면, N에 저장된 Count 변수 N.Count의 값은 다음과 같이 결정 된다.

$$N.Count = \sum_{q \in Q} Check_{bc}(q.R)$$

$Check_{bc}(q.R)$은 q.R이 N에 의해서 covered_by 또는 partially_intersect하면 1을 반환하고 그렇지 않다면 0을 반환하다.

정의 3: CL(Covering list): 질의 집합 Q와 QR-tree의 노드 N이 주어지면, N의 관련 CL인 N.CL은 질의 영역 q.R이 N을 cover 또는 equal 관계로 모든 질의 q의 질의 식별자 qid 를 저장하는 리스트이다.



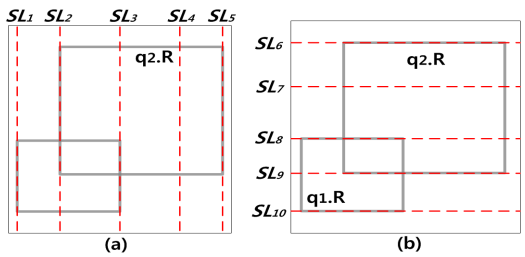
(그림 4) QR-tree의 예시

(Figure 4) An example of QR-tree

θ=1이라 가정하면, 그림 4는 3개의 질의 q1, q2, q3를 인덱싱하는 QR-tree의 예를 보여준다. 그림에서 QR-tree는 중심 분할 기법이 불필요한 분할을 발생시켜서 각 이동 객체에 o에 할당 된 상주 도메인의 크기가 작아지는 것을 볼 수 있다. 예를 들어, 그림4에서 이동 객체 o_1.Cap = 1이라고 가정하면 N_{112}에 해당하는 서브 공간이 상주 도메인으로 할당되고 질의 영역 q_2.R이 N_{112}와 중첩된다. 결과적으로 o_1은 N_{112} 밖으로 쉽게 이동할 수 있고 위치 업데이트(RequestDomain 메시지)를 서버에 전송한다. 또한 중심 분할 기법에 의해서 발생한 불필요한 분할은 QR-tree의 높이가 높아져서 i)서버의 CPU 비용과 ii)질의 처리 시간을 증가 시킨다.

3. 제안된 분할 기법

이번 장에서는 QR-tree에 불필요한 분할을 줄이기 위한 새로운 분할 기법을 제안한다. QR-tree 노드 N을 수직 또는 수평으로 분할하게 된다. 또 한, (N_1^{sl}, N_2^{sl})는 분할 선 SL에 의해 생성 된 서브 공간 쌍을 의미한다. 그림 5(a)와 5(b)는 수직 분할 선(SL_1, SL_2, …, SL_5) 및 수평 분할 선(SL_6, SL_7, …, SL_{10})의 예를 각각 나타낸다.



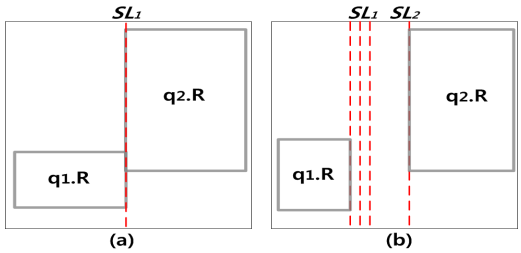
(그림 5) 분할 선의 예시

(Figure 5) Example of splitting lines

3.1 분할 규칙

질의 영역 집합 Q.R의 집합은 N에 covered_by거나 partially_intersects 관계이며, 제안 된 분할 기법은 다음 네 가지 규칙을 기반으로 한다.

- 규칙 (1): N_1^{sl}과 N_2^{sl}는 적어도 하나의 질의 영역 q.R∈Q.R과 4개의 중첩 관계 중 최소한 하나의 관계는 만족해야한다. 그림 5의 분할 선 (SL_1, SL_5, SL_6, SL_{10})은 무효이다. 다음 규칙은 규칙 (1)을 만족하는 분할 선에만 적용 될 수 있다.



(그림 6) 최적 분할 선의 예시

(Figure 6) Example of optimal splitting lines

- 규칙 (2): 분할 선 SL이 $(N_1^{sl}.\text{Count} \leq \theta) \wedge (N_2^{sl}.\text{Count} \leq \theta)$이면, SL을 최적 분할 선으로 간주한다. 최적의 분할 선은 N을 단 한 번만 분할하기 때문에 N에 분할 선이 하나만 있을 경우 최적의 분할 선으로 선택해야한다. θ = 1이라고 가정하면, 그림 6(a)는 하나의 최적 분할 선 SL_1만이 존재함을 보여준다. 이 규칙은 결과 서브 공간의 크기를 가능한 크게 유지할 수 있고 트리의 높이를 낮게 유지 할 수 있기 때문에 전체 시스템 성능을 향상 시킬 수 있다. 만약에 i) 최적 분할 선이 하나가 아니거나(그림 6(b) 참조) ii) 최적 분할 선이 존재하지 않을 때(그림 5(a), 그림 5(b) 참조) 규칙 (3)이 적용 된다.
- 규칙 (3): 질의 영역의 수가 covered_by 또는 partially_intersect 관계일 때, N_1^{sl}과 N_2^{sl}(N_1^{sl}.Count+N_2^{sl}.Count)는 최소화 되어야 한다. N에 두 개 이상의 최적 분할 선이 존재하는 경우에 규칙 (3)을 통해서 새로운 질의가 트리에 삽입 될 경우, N_1과 N_2이 서브 공간으로 나누어질 확률을 줄일 수 있다. 반면에 N에 최적의 분할 선이 없으면 이 규칙은 트리의 높이를 높게 만드는 분할을 줄일 수 있다. θ = 2라고 가정하

면, 그림 7(a)는 N에 2개의 최적 분할선이 있음을 보여준다. $N_1^{sl_1}$.Count+$N_2^{sl_1}$.Count=3이고 $N_1^{sl_2}$.Count+$N_2^{sl_2}$.Count=4기 때문에, SL1은 N의 최적 분할선이 된다. 이후로는 표기를 단순하게 하기 위해서 N_1^{sl}.Count+N_2^{sl}.Count를 SUM^{sl}로 표기한다. 만약, SUM^{sl}이 같은 값을 갖고 있는 경우에는 규칙 (4)가 적용된다. θ = 2로 가정하면 그림 7(b)에서 SUM^{sl}의 최소값이 5인 여러 쌍들 ($N_1^{sl_1}$,$N_2^{sl_1}$), ($N_1^{sl_2}$,$N_2^{sl_2}$), …, ($N_1^{sl_7}$,$N_1^{sl_7}$)이 존재한다.

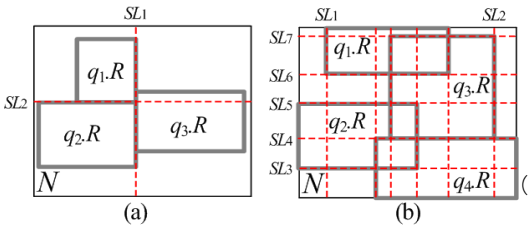


그림 7) 규칙 적용 사례
(Figure 7) Example of applying rule

- 규칙 (4): N_1^{sl}.Area와 N_2^{sl}.Area로 표현 된 N_1^{sl}과 N_2^{sl}영역은 가능한 크기가 같아야한다. 이 규칙에 따르면 N_1^{sl}.Area=N_2^{sl}.Area=$\frac{N.Area}{2}$일 때 최상의 경우이며, N.Area는 N의 면적을 의미한다. 이 규칙은 트리를 균형을 유지하는 것이 목적이다. 우리는 N_1^{sl}.Area와 N_2^{sl}.Area 사이의 유사도 측정을 위한 D^{sl}을 아래와 같이 정의한다.

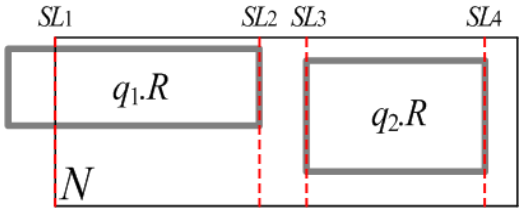
$$D^{sl}=(N_1^{sl}.Area-\frac{N.Area}{2})^2+(N_2^{sl}.Area-\frac{N.Area}{2})^2$$

D^{sl}의 값이 작을수록 N_1^{sl}.Area와 N_2^{sl}.Area의 크기가 유사함을 의미한다. 그림 6(b)에서 θ= 1일 때, N에 한 개 이상의 최적 분할선이 존재하고 SUM^{sl}가 최소값(=2)으로 동일한 셀 수 없는 쌍의 서브 공간이 존재하는 경우, D^{sl}값이 가장 적은 SL_2를 최적 분할선 으로 선택한다. 반면에, 그림 7(b)에서 θ= 2라고 가정하면 N에 최적의 분할선이 존재하지 않고 SUM^{sl}가 동일하게 최소값(=5)인 다수의 서브 공간 쌍이 존재 D^{sl}값이 최소인 SL_5를 선택한다.

3.2 분할 선 검색

이 절에서는 N에서 분할 선을 검색하는 방법을 제시한다. 여기에서는 N의 수직 분할 선을 검색하는 방법에 초점을 맞춘다. N의 왼쪽 끝에서 오른쪽 끝으로 스윕(Sweep)하는 가상의 수직 분할 선 SL이 있다고 가정한다. SL이 질의 영역 q.R ∈ Q.R의 왼쪽 끝에서 오른쪽 가장 끝으로 가로지를 때, SUM^{sl}이 변경 된다.

특히, SL이 q.R의 좌측 끝을 가로지르는 경우, SL에 의해 생성 된 좌측 서브 공간 N_1^{sl}에 의해 covered_by 또는 partially_intersect하는 질의 영역의 수가 증가 한다(N_1^{sl}.Count는 N_1^{sl}.Count + 1이 된다). 반면, SL이 q.R의 우측 끝을 가로지르면 N_1^{sl}.Count는 N_1^{sl}.Count - 1이 된다. 왼쪽 끝과 오른쪽 끝에서 q.R을 만족하는 분할 선을 후보 분할 선이라고 한다. θ= 1이라고 가정했을 때, 그림 8은 후보 분할 선의 예를 보여준다. 그림에서 q1.R의 왼쪽 가장자리가 N을 벗어나므로 N의 왼쪽 가장자리가 후보 분할 선 SL_1이 된다.



(그림 8) 후보 분할 선의 예시
(Figure 8) An example of candidate splitting lines

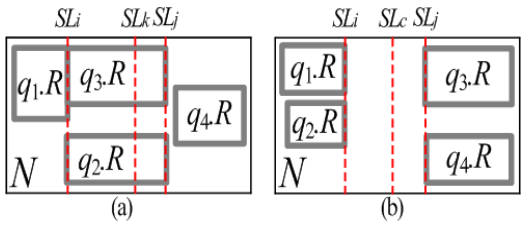
명제 1: 연속하는 두 개의 후보 분할 선 SL_i과 SL_j 사이의 임의의 분할 선 SL_k가 주어지면, $SL_j.x$는 SL_i, SL_k 그리고 SL_j의 x 좌표이며, 다음 조건이 성립한다.

$$(SUM^{sl_i} \le SUM^{sl_k}) \wedge (SUM^{sl_j} \le SUM^{sl_k})$$

증명 1: 각각 SL_i, SL_j 및 SL_k에 의해 생성 된 서브 공간 쌍 ($N_1^{sl_i}$,$N_2^{sl_i}$), ($N_1^{sl_j}$,$N_2^{sl_j}$), ($N_1^{sl_k}$,$N_2^{sl_k}$)가 $N_1^{sl_i}$.Count = p 및 $N_2^{sl_i}$.Count = q라고하자. $N_1^{sl_i}$.Count = p + α, $N_2^{sl_i}$.Count = q - β, $N_1^{sl_k}$.Count = p + α, $N_2^{sl_k}$.Count = q 일 때, 여기서 α와 β는 SL_i가 왼쪽 끝에서 만나는 질의 영역의 수를 나타낸다. SL_j는 우측 끝에서 만나는 질의 영역의 수를 각각 나타낸다. 왜냐하면 SUM^{sl_i}= p+q,

SUM^{sl_j}= p + α+ q-β, SUM^{sl_k}= p + α+ q이기 때문에 명제 1은 즉시 성립한다.

θ= 3이라고 가정했을 때, 그림 9(a)는 두 개의 후보 분할 선 (SLi, SLj)과 그 사이에 있는 분할 선 (SLk)를 보여준다. 그림에서 p = 1, q = 3, α= 2 및 β= 2이기 때문에 SUM^{sl_i}=4, SUM^{sl_j}=4 및 SUM^{sl_k}=6이다. 명제 1에 기초하여, N의 후보 분할 선들과 N의 중앙에 위치한 분할 선 (그림 9(b)의 SLc), 즉 SLc가 규칙 (1), (2), (3), (4)를 모두 만족한다. 우리는 N의 분할 선과 중심 분할 선을 검색하기 위해 SweepSpace라는 간단한 스위핑 알고리즘을 사용한다. SweepSpace는 N을 왼쪽에서 오른쪽으로 스윕하여 수직 분할 선 (중앙 분할 선 포함)을 검색하고, 아래에서 위로 N을 스윕하여 수평 분할 선 (가운데 분할 선 포함)을 검색한다. 수직 후보 분할 선을 검색 할 때 SweepSpace는 규칙 (1)을 적용하여 가장 왼쪽과 가장 오른쪽의 줄을 제거합니다. 유사하게, 수평 후보 분할 선을 검색 할 때, 가장 아래 선과 가장 위의 선을 제거한다.



(그림 9) SLi, SLj, SLk, SLc 의 예시

(Figure 9) Examples of SLi, SLj, SLk,and SLc

3.3 최적의 분할 선 탐색

이 절에서는 후보 분할 선들 중에서 최적의 분할 선을 찾기 위한 알고리즘을 제안한다.

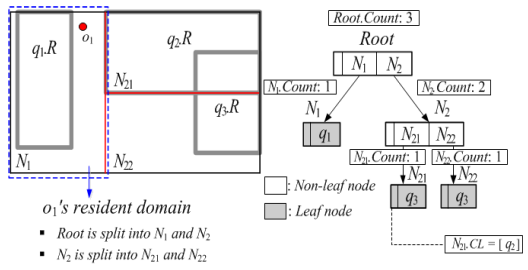
각 후보 분할 선 SL에 대해, 현재 최상의 분할 선 BestSL이 없다면, SL은 현재의 BestSL이 된다(2-3). 그렇지 않다면 알고리즘은 SL이 최적의 분할 선인지 확인하고 SL이 최적의 분할 선이고 현재의 BestSL이 최적의 분할 선이 아닐 때, SL은 규칙 (2)를 적용하여 현재 BestSL이 된다(5-7). SL과 현재 BestSL이 모두 최적의 분할 선인 경우 알고리즘은 규칙 (3)과 규칙 (4)를 적용하여 SL이 현재 BestSL보다 우수한 지 확인한다. 이 경우 SL이 현재 BestSL이 된다(9-13). 반면, SL과 현재 BestSL이 최적의 분할 선이 아니라면, 알고리즘은 규칙 (3)과 규칙 (4)을 적용하여 비교하고, 필요한 경우 BestSL을 업데이트한다 (16-20). 위와 같은 알고리즘으로 최적의 분할 선을 찾고 BestSL(21)을 반환하게 된다.

그림 10은 제안 된 분할 기법을 적용한 QR-tree의 예를 보여준다. 그림과 같이 개선 된 QR-tree의 분할 횟수는 그림 4의 QR-tree의 수보다 적기 때문에, 각 이동 객체 o에 할당 된 상주 도메인의 평균 크기가 훨씬 커진다. 예를 들어, 그림 10을 θ= 1이라 가정 했을 때, N1에 상응하는 서브 공간이 상주 도메인으로 할당되고 질의 영역 q1.R이 N1과 중첩된다. 또 한, 강화 된 QR-tree의 높이는 그림 4의 QR-tree 높이보다 낮다는 것을 알 수 있다.

Algorithm 1. Finding the best splitting line

Input *N*: a QR-tree node, SL*list*: a list of candidate splitting lines (SLs)
Output *Best*SL: the best splitting line among SLs
Procedure

```
1: for each SL ∈ SLlist do
2:     if BestSL is null then
3:         BestSL ← SL;
4:     else
5:       if (N1^sl.Count ≤ θ) and (N2^sl.Count ≤ θ) then
6:
7:         if (N1^bestsl.Count > θ) or (N2^bestsl.Count > θ) then
8:           BestSL ← SL;
9:         else
10:          if SUM^sl < SUM^bestsl then
11:            BestSL ← SL;
12:          else if SUM^sl = SUM^bestsl then
13:            if D^sl < D^bsetsl then
14:
15:              BestSL ← SL;
16:     else   //(N1^sl.Count > θ) or (N2^sl.Count > θ)
17:
18:       if (N1^bestsl.Count > θ) or (N2^bestsl.Count > θ) then
19:         if SUM^sl < SUM^bestsl then
20:           BestSL ← SL;
21:         else if SUM^sl = SUM^bestsl then
              if D^sl < D^bestsl then
                BestSL ← SL;
    return BestSL
```



(그림 10) 강화 된 QR-tree 의 예시

(Figure 10) An Example of the enhanced QR-tree

4. 성능 평가

이 장에서는 제안 된 분할 기법을 적용한 강화 된 QR-tree의 성능을 평가하고 MQR에서 사용 된 BP-tree 및 기존의QR-tree와 서버에서의 무선 통신 비용 및 CPU 비용에 대해서 실험을 진행한다. 서버의 CPU 비용은 서버가 영역 모니터링 질의를 처리하는 데 소요되는 CPU 시간을 기준으로 측정했다. 무선 통신 비용은 (i) 이동 객체에서 서버로 전송 된 메시지의 수와 (ii) 서버에서 이동 객체로 브로드캐스팅 되는 메시지 수의 합으로 측정되었다. 본 시뮬레이션은 Inter Core i7-3770, 8GB RAM, 리눅스 시스템에서 시행하였다.

4.1 시뮬레이션 설정

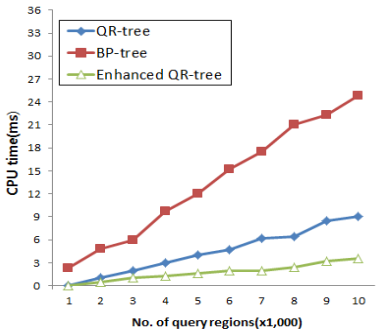
시뮬레이션을 위해서 작업 영역 (50km × 50km 정사각형)에 질의 영역을 일정하게 배치하였고 랜덤 웨이 포인트 모델[6]에 따라 움직이는 이동 객체를 생성했다. 랜덤 웨이 포인트 모델에서 각 객체 o는 현재 위치에서 임의의 목적지로 0에서 최대 속도까지 균일하게 분산 된 일정한 속도로 이동한다. 목적지에 도착하면 일정 기간 동안 고정되어 있고, 이 기간이 만료되면 o가 새로운 목적지를 선택해서 동일한 과정을 반복한다. BP-tree, QR-tree, 강화 된 QR-tree에 한 번에 로드 되어 처리 할 수 있는 질의 영역의 수(o.Cap)을 [25,100] 범위에서 무작위로 선택되게 하였고 25로 설정 되었다.

(표 1) 시뮬레이션 매개변수
(Table 1) Simulation Parameters

Simulation parameter	Value used(Default)
질의 영역 수	1,000~10,000(5,000)
이동 객체 수	10,000~100,000(50,000)

표 1은 시뮬레이션에 사용 된 매개 변수와 기본 값을 나타냈으며, 각 시뮬레이션에서 하나의 매개변수를 변화시키며 측정하였다. 각 시뮬레이션은 1,00번의 시뮬레이션 시간 단계를 실행하여 측정하였다.

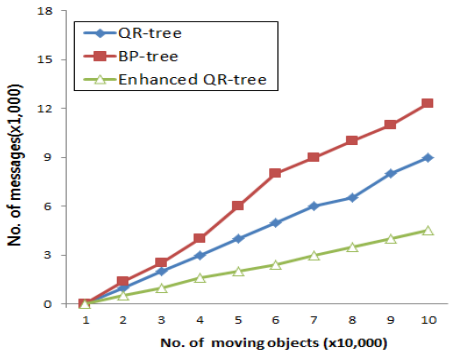
1) 질의 영역 수의 영향 : 첫 번째 시뮬레이션에서 질의 영역의 수를 1,000에서 10,000으로 변경하고 서버에서의 CPU 비용 및 무선 통신 비용에 대한 질의 수의 영향을 조사하였다. 이 시뮬레이션은 질의 영역의 수와 관련하여 향상된 QR-tree의 확장성을 보여준다.



(그림 11) CPU 시간 vs 질의 개수
(Figure 11) CPU time vs number of query regions

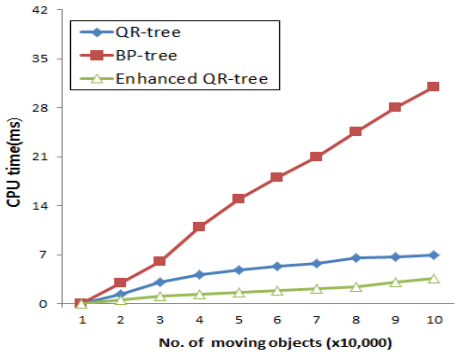
그림 11은 서버가 질의 처리에 사용하는 CPU 시간에 대한 질의 영역 수의 영향을 보여준다. 그림에서 볼 수 있듯이 향상된 QR-tree는 BP-tree 및 QR-tree보다 훨씬 뛰어난 것을 알 수 있다. 이것은 향상된 QR-tree가 서버가 이동 개체에 훨씬 더 큰 상주 도메인을 할당하기 때문이다. 결과적으로, 서버는 상주 도메인을 결정하는 빈도를 줄여서 이동 객체가 상주 도메인을 벗어나는 경우를 줄일 수 있다.

그림 12는 이동 객체와 서버 간에 통신되는 총 메시지 수에 대한 질의 영역의 영향을 보여준다. 질의 영역 수가 증가하면 모든 트리의 성능이 저하 된다. 하지만 향상된 QR-tree는 서버가 더 큰 상주 도메인을 할당 할 수 있게 해주기 때문에 BP-tree와 QR-tree보다 좋은 성능을 보여준다. 이는 이동 객체가 새로운 상주 도메인을 받기 위해 서버에 전송하는 RequestDomain 메시지의 수를 감소시키며, 서버가 새로운 상주 도메인을 이동 객체에 할당할 때 발생하는 통신 오버 헤드를 줄일 수 있다.



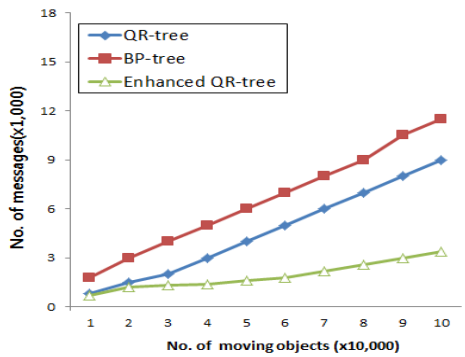
(그림 12) 메시지 vs 질의 개수
(Figure 12) Messages vs. number of query regions

2) 이동 객체 수의 영향 : 위 시뮬레이션에서, 이동 객체의 수를 10,000에서 100,000 개로 증가시켜, 이동 객체가 BP-tree, QR-tree 및 향상된 QR-tree의 성능에 미치는 영향을 확인했다.



(그림 13) CPU vs 이동 객체 수
(Figure 13) CPU vs. number of moving objects

그림 13, 14에 보여 지는 것과 같이, 이동 객체의 수가 증가하면, 서버와 이동 객체 사이에서 송신되는 CPU 시간 및 메시지의 수가 증가한다. 그러나 모든 경우에서 향상된 QR-tree는 첫 번째 시뮬레이션의 설명에서 언급 된 이유들 때문에 BP-tree 및 QR-tree보다 성능이 우수한 것을 알 수 있다. 다시 말해, 확장 된 QR-tree는 거대한 양의 이동 객체에 대한 영역 모니터링 질의를 처리 할 때, BP-tree 및 QR-tree 보다 효과적이다. 본 시뮬레이션은 이동 객체에 더 큰 상주 도메인을 할당 할 수 있으면 서버의 CPU 비용과 무선 통신 비용을 크게 줄일 수 있었다.



(그림 14) 메시지 vs. 이동 객체의 수
(Figure 14) Messages vs. number of moving objects

5. 결 론

본 논문에서 영역 모니터링 질의를 효율적으로 처리하는 문제를 다루었다. 본 연구의 목적은 지리적으로 분산 된 이동 객체가 주어졌을 때, 이동 객체와 관련 된 여러 가지 질의를 평가하게 함으로써 서버의 무선 통신 비용과 CPU 비용을 최소화시키고 질의 결과를 최신 상태로 유지시키는 것이었다. 본 논문에서는 상주 도메인 개념을 활용하였고 그 결과, 서브 공간과 질의 영역 간의 공간적 관계와 질의 영역의 분포에 적응하는 새로운 작업 공간 분할 기법을 제안해냈다. 제안 된 분할 기법을 적용하여 향상된 QR-tree라고 부르는 QR-tree를 구축했다. 우리는 시뮬레이션을 수행하고 향상된 QR-tree가 기존의 인덱싱 구조보다 뛰어난 성능을 보여주는 것을 확인했고 제안 된 분할 기법의 유효성을 입증했다.

참고문헌(Reference)

[1] S. Ilarri, E. Mena, and A. Illarramendi, “Location-Dependent Query Processing: Where We Are and Where We Are Heading,” ACM Computing Surveys, 42 (3), pp. 1-73, 2010.
http://dx.doi.org/10.1145/1670679.1670682

[2] Y. Cai, K. A. Hua, G. Cao, and T. Xu, “Real-Time Processing of Range- Monitoring Queries in Heterogeneous Mobile Databases,” IEEE Trans. on Mobile Computing, 5(7), pp. 931-942, 2006.
http://dx.doi.org/10.1109/TMC.2006.105

[3] H. Jung, Y. S. Kim, and Y. D. Chung,, “QR-tree: An efficient and scalable method for evaluation of continuous range queries,” Information Sciences, 274, pp. 156-176, 2014.
https://doi.org/10.1016/j.ins.2014.02.061

[4] M. A. Cheema, L. Brankovic, X. LIN, W. Zhang, and W. Wang, “Continuous Monitoring of Distance-Based Range Queries,” IEEE Trans. on Knowledge and Data Engineering, vol. 23, no. 8, pp.1182-1199, Aug. 2011.
http://dx.doi.org/10.1109/TKDE.2010.246

[5] B. Gedik, and L. Liu, “Mobieyes: A Distributed Location Monitoring Service Using Moving Location Queries,” IEEE Trans. on Mobile Computing, vol. 5,

no. 10, pp. 1384-1402, Oct. 2006.
http://dx.doi.org/10.1109/TMC.2006.153
[6] H. Jung, M. Song, H. Y. Youn, and U. M. Kim, "Evaluation of Content-Matched Range Monitoring Queries over Moving Objects in Mobile Computing Environments," Sensors, vol. 15, no. 9, pp. 156-176, Sept. 2015. http://dx.doi.org/10.3390/s150924143

◐ 저 자 소 개 ◑

정 재 우(Jaewoo Chung)
2015년 을지대학교 의료IT마케팅학과(학사)
2016년~현재 성균관대학교 대학원 전자전기컴퓨터공학학과(석사과정)
관심분야 : Database, Spatial Database, Location-based Services
E-mail : cjw0828@skku.edu

정 하 림(HaRim Jung)
2004년 광운대학교 컴퓨터 소프트웨어전공 (학사)
2007년 고려대학교 대학원 컴퓨터학과 (석사)
2012년 고려대학교 대학원 컴퓨터·전파통신공학과(박사)
2012년~2013년 성균관대학교 정보통신대학 박사후 연구원
2013년~현재 성균관대학교 정보통신대학 연구교수
관심분야 : Mobile/Pervasive Databases, Spatio-temporal Database, Location Based Information System 등
E-mail : harim3826@gmail.com

김 응 모(Ung-Mo Kim)
1981년 성균관대학교 수학과 (학사)
1986년 Old Dominion University 전산학과 (석사)
1990년 Northwestern University 전산학과 (박사)
1990년~현재 성균관대학교 컴퓨터공학과 교수
관심분야 : Database, Database, Spatial Database, Location-based Services, Big Data 등
E-mail : umkim@skku.edu