모바일 지도 서비스를 위한 에이전트 기반의 공간 데이터 캐쉬의 설계 및 구현

임 덕 성[†] · 이 재 호[††] · 홍 봉 희[†††]

요 약

PDA와 같은 무선 단말기에서 지도 데이터에 대한 검색 및 접근을 위해 영역단위의 캐쉬와 R-tree 기반의 공간 색인이 필요하다. 그러나 서버로부터 낮은 저장용량의 무선 단말기에 전송되는 공간 객체는 캐쉬된 공간 객체와 중복되는 문제가 있다. 또한 추가되는 데이터를 저장하고, 효율적인 공간 질의를 위한 색인 재구축 비용은 낮은 컴퓨팅 파워를 가진 무선 단말기에 부하를 준다. 따라서 낮은 컴퓨팅 파워를 가진 무선 단말기의 부하를 분산시키는 방법과 중복 객체의 처리 기법이 필요하다. 이 논문에서는 먼저 캐쉬시 중복 객체에 대한 처리 기법을 분류하고, 클리핑 기법을 사용한 공간 객체 저장과 색인 재구성 방법을 분석한다. 또한, 무선 단말기에 집중된 부하를 분산시키기 위해 색인 구성 및 클리핑 작업을 에이전트에서 처리하는 에이전트 기반 캐쉬 시스템을 제시한다. 그리고, 제시한 시스템을 설계 및 구현하고 성능을 평가한다.

Design and Implementation of the Spatial Data Cache Based on Agents for Providing Mobile Map Services

Duksung Lim[†] · Jaiho Lee[††] · Bonghee Hong[†††]

ABSTRACT

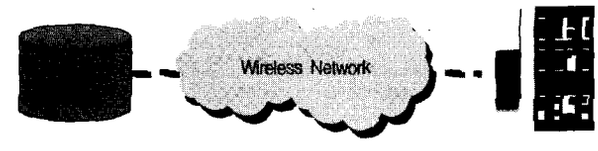
Mobile clients like a PDA need a cache and a spatial index to search and access map data efficiently. When a server transmits spatial objects to a mobile client which has a low storage capacity, some of them can be duplicated in a cache of the mobile client. Moreover, the cost for storing added data in the cache and reconfiguring spatial index is very high in the mobile client with low computing power. The scheme for processing duplicated objects and distributing tasks of the mobile client which has low computing power is needed. In this paper, we classify the method for storing duplicated objects and present the scheme for both caching objects and reconfiguring a spatial index of cached objects using the clipping technique. We propose the caching system based on an agent in order to distribute the overhead of a mobile client as well as to provide efficiently map services. We design and implement it, and evaluate the performance.

키워드 : 모바일 지리정보시스템(Mobile GIS), R-트리(R-tree), 캐쉬(Cache), 클리핑(Clipping)

1. 서 론

최근 PDA와 같은 무선 단말기의 보급이 증가하고 이를 이용한 다양한 정보를 얻고자 하는 요구가 커짐에 따라 무선 네트웍을 이용한 많은 응용들이 개발되고 있다. 특히 이동 환경에서 실시간 지도 정보 서비스, LBS, ITS의 응용을 위해서는 모바일 클라이언트에서 공간 데이터 검색이 필수적이다[6].

(그림 1)과 같이 PDA와 같은 모바일 클라이언트가 무선 통신을 통한 지도 데이터를 송, 수신하는 환경에서 GIS 서버는 질의 처리를 수행하고, PDA는 영역 단위의 지도 요청을 한다. 이와 같은 환경에서 무선 통신의 비용 감소와 질의 응답시간 향상을 위해 캐쉬가 필요하다. 그러나 PDA에서의 캐쉬는 저장 공간의 제약성을 가지므로 검색하는 영역만을 저장해야 하고, 효과적으로 공간 데이터를 검색하기 위해서는 공간 색인이 필요하다.



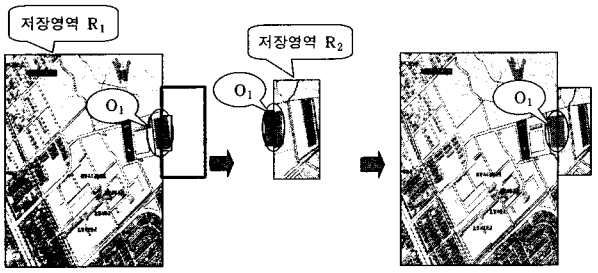
(그림 1) 대상 환경

PDA에서의 공간 색인은 사용자의 요청에 따라 새로운 영역의 삽입과 삭제가 용이한 동적인 구조를 가져야 하고,

† 준 회 원 : 부산대학교 대학원 컴퓨터공학과
†† 정 회 원 : 한국전자통신연구소 연구원
††† 정 회 원 : 부산대학교 컴퓨터공학과 교수
논문접수 : 2002년 3월 8일, 심사완료 : 2002년 10월 23일

효과적인 영역 단위의 데이터 추가 연산을 요구한다. 따라서, 이 논문에서는 공간 색인을 위해 동적인 구조를 가지고 있고 영역 단위의 데이터를 삽입하기 위한 Bulk-Operation을 적용할 수 있는 R-Tree 색인을 기반으로 한다.

PDA에서 효과적으로 공간 데이터를 검색하기 위해서는 캐쉬와 공간 색인이 필요하지만 다음과 같은 문제점들이 존재한다. 첫째, 캐쉬 영역 사이의 중복 객체에 대한 처리이다. PDA는 사용자 요청 데이터만을 저장하기 위해서 질의 영역을 캐쉬의 단위로 사용해야 한다. 사용자의 요청 시 캐쉬되어 있지 않은 영역에 대해서 서버로 질의하여 결과를 전송 받는다. 새로운 영역을 서버로 요청했을 때 이미 캐쉬된 영역과 서버로 질의 영역 사이에는 중복된 객체가 존재할 수 있다. (그림 2)는 중복된 객체가 존재하는 예를 보이고 있다. 저장영역 R_1은 이미 캐쉬된 영역이며 공간 객체 O_1을 이미 저장하고 있다. 그러나 PDA에서 추가영역 R_2를 요청하여 수신한 질의 결과 데이터에도 캐쉬된 영역의 O_1과 동일한 객체 O_1이 존재한다. 따라서 O_1과 같은 중복된 객체에 대한 처리 문제가 발생한다.



(그림 2) 중복 객체 존재

둘째, 고 비용의 색인 구성 연산이 발생한다. 서버로 받아온 질의 영역내의 객체들을 R-Tree 공간 색인에 삽입하는 연산을 수행해야 한다. 이때 PDA의 낮은 컴퓨팅 능력을 고려하면 영역 단위의 많은 객체들을 하나씩 색인에 추가하는 연산은 고 비용의 연산이다. 따라서 적은 비용으로 빠르게 추가 영역을 색인에 삽입하기 위해 Bulk-Insertion [2] 방법을 적용한다. Bulk-Insertion 방법은 추가 영역에 대한 색인 구성과 기존의 색인에 추가하는 연산으로 이루어져 있다. 서버로부터 받아온 추가 영역에 대한 색인을 구성하는 방법으로는 객체를 하나씩 삽입하는 방법과 Bulk-Loading의 방법을 들 수 있다. 연구된 Bulk-Loading의 방법[8-10]으로는 R-Tree를 Bottom-up 방식으로 구성하는 방법과 Top-Down 방식으로 구성하는 방법이 있다. 그러나 이와 같이 추가 영역에 대한 여러 색인 구축 방법들이 있지만 각 방법의 경우 고 비용의 복잡한 연산을 수반하기 때문에 PDA와 같은 낮은 계산 능력을 가진 모바일 클라이언트에서 처리가 어려운 단점을 가진다.

셋째, R-Tree 공간 색인에서 캐쉬 영역을 삭제 시에 고 비용의 연산이 발생한다. 저장 공간이 부족하여 더 이상의 공간 데이터를 수용할 수 없을 경우 교체가 일어난다. 교체 작업 시에 교체 대상 영역 내에 속한 공간 객체들의 삭제 연산을 수행하게 된다. 교체 대상의 캐쉬 영역 내에는 많은 공간 객체들이 존재하고 있다. 이 객체들을 하나씩 삭제를 하게 되면 삭제할 객체의 수에 비례한 연산 비용을 수반한다. 따라서 영역 단위의 빠른 삭제 방법이 필요하다.

이 논문에서는 첫째, 중복 객체를 처리하기 위한 방법으로 중복 저장, 클리핑, 단일 저장 방법으로 분류한다. 중복 저장 방법은 동일한 객체에 대한 중복을 허용하는 방법이다. 클리핑 방법은 질의 영역 외부의 영역을 가지는 공간 객체를 클리핑하여 저장하는 방법이다. 그리고 단일 저장 방법은 질의 결과에서 이미 캐쉬된 공간 객체를 제거하여 PDA에는 하나의 객체만 저장되도록 하는 방법이다. 둘째, PDA에서 중복 객체 처리 비용과 색인 구성 연산 비용을 줄이기 위해 에이전트 개념을 기반으로 한 구조를 제시한다. 즉, 중복 객체 처리시 클리핑 비용과 추가 영역에 대한 색인 구성 비용을 에이전트에 전가 시키는 구조로 서버에 추가적인 부하를 주지 않으면서 PDA의 연산 비용을 감소시키는 구조이다. 셋째, R-tree 공간 색인을 기반으로 한 캐쉬에서 교체시 Bulk-Operation 알고리즘을 제시한다.

이 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴본다. 3장에서는 중복 객체의 처리 방법에 대한 분류하고, 4장에서는 에이전트 개념을 기반으로 한 캐쉬 시스템 구조 및 세부 구성 요소에 대해 설명한다. 5장에서는 구현 및 실험 환경에 대해 설명하며, 마지막으로 6장에서는 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

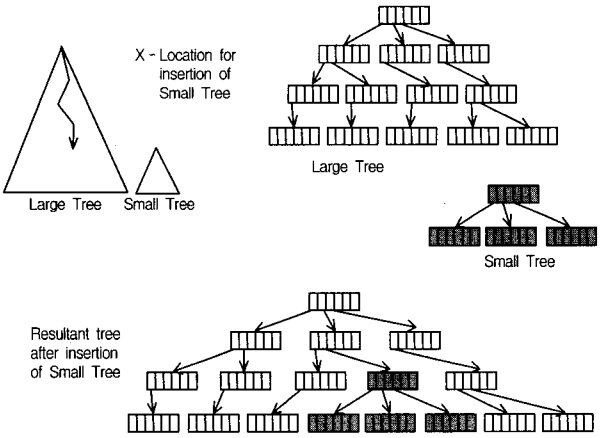
기존의 캐쉬 기법에 대한 연구는 크게 캐쉬 단위와 교체 전략의 두 가지로 분류된다. 첫째, 페이지나 객체, 시맨틱 영역 등의 다양한 캐쉬 단위에 대한 연구가 있다. 둘째, 캐쉬의 적중률을 높이기 위한 캐쉬 교체 전략에 대한 연구가 있다. [4]에서는 캐쉬 단위를 페이지, 객체, 시맨틱 단위로 분류하고 다음과 같이 장단점을 기술한다. 첫째, 페이지 단위의 캐쉬는 데이터 페이지의 클러스터링 정보를 이용하여 디스크 I/O를 줄이기 위한 것으로 클러스터링 상태와 인덱스에 따라 캐쉬의 효율성이 달라진다. 둘째, 객체 단위의 캐쉬는 최대의 융통성을 가지지만 각 객체 단위별로 관리하는 메모리 비용이 크며 데이터를 요청하는 통신 횟수가 많기 때문에 성능이 떨어지게 된다. 셋째, 시맨틱 캐쉬는 클라이언트의 질의에 대한 결과를 단위로 하기 때문에 객체 단위에 비해 관리 비용이 적으며 페이지 단위에 비해 캐쉬

단위 내의 데이터는 의미적으로 연결성을 가질 수 있다. 그러므로 사용자가 요청한 영역 중 저장되어 있지 않은 영역만을 서버로 요청하고 저장하기 위해서는 시맨틱 캐쉬를 이용한 질의에 대한 결과 영역 단위로 캐쉬하는 것이 효율적이다. 시맨틱 캐쉬를 위한 모델로는 [5]의 연구가 있다. 이 연구에서 시맨틱 캐쉬의 질의는 probe query와 remainder query 두 가지로 나뉜다. probe query는 이미 캐쉬된 데이터에 대한 질의이며 remainder query는 서버로 새로운 데이터를 요청하기 위한 질의이다. 시맨틱 캐쉬의 특징은 첫째 클라이언트는 물리적인 페이지나 튜플 대신 캐쉬된 데이터의 의미적인 명세를 유지하고 서버로부터 필요한 데이터를 remainder query로써 기술한다. remainder query는 병렬성과 통신 양을 줄인다. 둘째, 캐쉬의 교체 전략에 사용된 정보는 연관된 데이터들의 집합인 의미적 영역(semantic region)에 대해서 유지된다. 의미적 영역의 사용은 튜플 단위의 캐쉬 방법의 교체 정보 저장에 대한 공간 오버헤드를 피하고 페이지 단위의 캐쉬 방법과는 달리 페이지의 튜플들의 나쁜 클러스트링에 무관하다. 셋째, 유지하고 있는 의미적인 명세에 대한 교체 값 함수는 새로운 모바일 데이터베이스와 같은 응용 뿐만 아니라 전통적인 질의기반의 응용을 위해서도 적용될 수 있다.

캐쉬 교체 전략으로는 데이터의 최근 사용 시간 정보를 기반으로 한 LRU와 MRU 알고리즘과 데이터의 접근 빈도수를 이용한 LFU 기법이 있으며, LRU 기법을 확장하여 사용빈도수를 추가 정보로 이용하는 LRU-K 알고리즘[7]이 있다. 또한, 공간 데이터 캐쉬의 경우 클라이언트/서버 환경에서 클라이언트의 캐쉬 적중률을 높이기 위해 사용자 접근 유형을 고려한 공간 근접성 기반의 알고리즘[14]과 서버에서의 디스크 I/O를 줄이기 위해 질의간 위치 관련성에 따라 질의 스케줄링을 하는 방법[15]이 있다. 그러나, 이러한 기법들은 메모리 기반의 구조이고 컴퓨팅 파워가 낮은 PDA에서 실시간으로 빠르게 처리하기 위해서 비교적 성능이 우수하고 적은 저장 공간과 간단한 연산처리를 하는 LRU 기법을 적용하는 것이 효율적이다.

중복 객체를 처리하기 위한 기법으로 공간 객체 접근 방법[13]은 공간 객체의 효율적 검색을 위한 색인 구성을 위한 방법을 Object Mapping, Object Bounding, Object Duplication, Multiple Layer와 같이 4가지로 분류한다. 그러나 색인을 포함하는 캐쉬의 중복 객체 처리의 경우 Object Mapping, Object Bounding, Multiple Layer 방법은 객체의 단일 저장으로 분류되고, Object Duplication은 방법은 객체의 중복 저장과 클리핑 방법으로 분류된다. 각 처리방법은 3장에서 기술한다.

캐쉬된 공간 데이터의 효과적인 처리 및 검색을 위해서는 공간 색인들 중에서 이 논문에서는 R-Tree 공간 색인[1]을 기반으로 한다. R-Tree 공간 색인의 경우 사용자의 요청에 의해 저장 영역이 동적으로 변화하게 되는데, 해쉬 기반의 공간 색인 구조 보다는 트리 구조를 가지고 있기 때문에 동적으로 확장이 용이하고, 지도 검색시 자주 사용되는 영역 질의의 경우 검색이 효과적이다.



(그림 3) STLT 알고리즘

또한, Bulk-Insertion[2, 3] 기법을 적용하여 객체 단위로 하나씩 삽입하는 것이 아니라 영역 단위의 데이터의 삽입을 할 경우 삽입 속도를 개선할 수 있다. (그림 3)의 STLT 알고리즘[2]의 Bulk-Insertion 기법은 기존의 공간 색인을 Large Tree라 할 경우 추가될 영역의 공간 색인을 Small Tree 형태로 구축하여 Large Tree에 삽입하는 방법이다. STLT를 사용할 경우 객체별 삽입 방법보다 삽입 비용이 낮은 장점을 가진다.

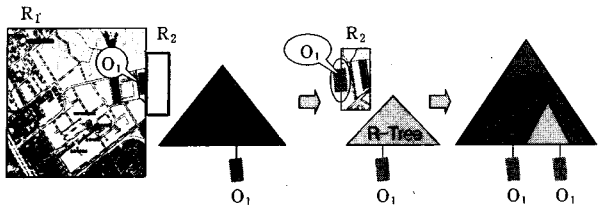
3. 중복 객체 처리

이 장에서는 캐쉬 단위인 질의 영역 사이에 존재하는 중복 객체의 처리 방법들을 분류하고 이를 저장공간, 검색 속도 및 연산 비용에 대해서 기술한다.

3.1 중복 저장

중복 저장 방법은 질의 영역 사이의 중복 객체를 허용하여 동일한 객체를 중복 저장하는 방법이다. (그림 4)(a)는 시간 $t_1(t_1 < t_2 < t_3)$에 PDA가 영역 R_1을 캐쉬하고, R_1에 대한 인덱스를 구성한다. (그림 4)(b)는 시간 t_2에 PDA에서 영역 R_2를 요청하여 추가될 R_2의 데이터와 추가 데이터에 대한 색인을 나타낸다. (그림 4)(c)는 시간 t_3에 PDA에서 R_1과 R_2를 포함한 전체 영역에 대한 색인과 색인에서의 객체 중복을 나타낸다. 이때, 이미 캐쉬된 영역 R_1에는 객체 O_1이 이미 존재하고 있고 추가 영역 R_2에도 객체 O_1이 존재한다. 이러한 중복을 허용하여 저장함으로써 질의영역 R_2가 추가된

후에 객체 O_1이 중복해서 저장되고 색인에 존재하게 된다.



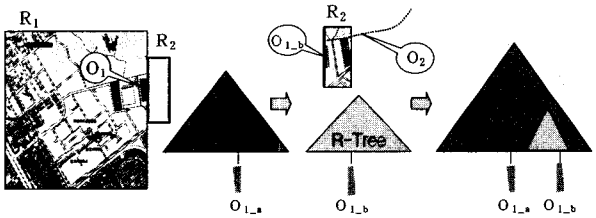
(a) t_1에 캐쉬된 데이터 및 인덱스 (b) t_2에 요청한 데이터 (c) 삽입된 상태 t_3

(그림 4) 중복 저장

중복 저장 방법은 PDA에서 동일한 객체에 대한 중복을 허용함으로써 저장 영역에 대한 객체 검색 속도가 빠르고, 교체시 관리가 간단한 장점을 가지지만 다음과 같은 단점들을 가진다. 첫째, 저장공간이 제한된 PDA에 저장 공간에 대한 활용도를 떨어뜨린다. 둘째, 동일한 객체가 색인에 동시에 존재함으로써 색인의 노드수를 증가시킨다. 셋째, 영역 질의 시 트리의 탐색 경로를 증가시킨다. 넷째, 영역 질의 결과 내에 동일한 객체가 중복 존재하게 된다.

3.2 클리핑

클리핑 방법은 공간 객체의 영역 중 질의 영역내의 모든 데이터와 질의 영역과 교차하는 객체인 경우 질의영역에 포함되는 부분만을 저장하는 방법이다.



(a) t_1에 캐쉬된 데이터 및 인덱스 (b) t_2에 요청한 데이터 (c) 삽입된 상태 t_3

(그림 5) 클리핑

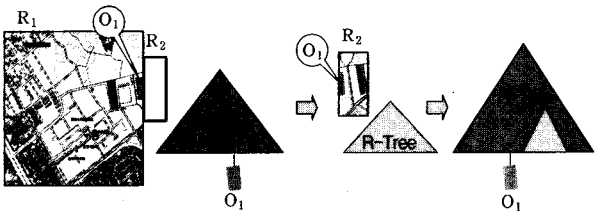
(그림 5)는 클리핑하여 저장하는 방법의 예를 보여 주고 있다. 객체 O_1은 이미 저장된 캐쉬 영역 R_1과 추가 영역 R_2에 모두 속한 객체이다. 질의 영역 R_1에는 객체 O_1이 R_1의 경계 부분으로 클리핑되어 O_{1_a}라는 식별자 형태를 부여 받고 클리핑된 부분의 데이터만이 저장된다. 추가영역 R_2에는 객체 O_1이 O_{1_b}의 형태로 클리핑 되어 저장되고 색인이 구성된다. (그림 5-c)와 같이 두 번의 질의 영역 요청 시 PDA에 캐쉬된 객체 O_1은 O_{1_a}와 O_{1_b}의 형태로 저장되고 색인에 존재한다.

클리핑 방법은 다음과 같은 단점을 가진다. 첫째, 클리핑 연산을 처리해야 하는 연산 비용이 발생한다. 둘째, 동일한 객체가 잘려진 형태로 색인에 중복 존재 함에 따라 색인 공간이 증가 할 수 있다. 중복저장 방법에 비해 저장 공간의 활용도를 높일 수 있는 반면 객체 검색시 공간 색인의 비용는 객체의 분할 정도에 따라 증가하는 단점이 있다.

그러나 클리핑의 큰 장점으로 (그림 5)(b)의 도로 레이어에 속한 공간 객체 O_2와 같이 질의 영역에 객체의 일부가 포함되지만 객체의 대다수 영역이 질의 영역에 포함되지 않을 경우 사용자가 요청한 영역에 속하지 않는 데이터를 제거함으로써 저장 공간의 부하를 줄일 수 있을 뿐아니라 공간 색인에서 사장 영역을 최소화하여 검색 성능을 향상시킬 수 있다.

3.3 단일 저장

단일 저장 방법은 새롭게 추가 되는 캐쉬 영역에서 중복된 객체를 제거하는 방법이다. 즉 추가되는 영역에서 이미 캐쉬된 객체를 제거한 후 색인을 구성하여 기존 색인에 삽입하는 것이다.



(a) t_1에 캐쉬된 데이터 및 인덱스 (b) t_2에 요청한 데이터 (c) 삽입된 상태 t_3

(그림 6) 단일 저장

(그림 6)의 객체 O_1은 이미 저장된 캐쉬 영역 R_1과 추가 영역 R_2에 모두 속한 객체이다. 따라서 단일 저장 방법은 추가 영역의 데이터 중에서 객체 O_1을 제거한 후 색인을 구성하여 기존의 색인에 추가하는 방법이다. 그러나, 이 방법은 다음과 같은 단점을 가진다. 첫째, 질의 결과에서 중복된 객체를 제거하는 연산을 필요로 한다. 둘째, 교체가 발생하여 캐쉬된 영역을 삭제할 경우 PDA에서 삭제를 위한 비용이 크다. (그림 7)에서 R_2가 교체 대상으로 선택되어 삭제될 예정이고, 객체 A와 객체 B가 R_2에 저장되어 있다고 가정하자. 교체를 위해서 R_2의 객체를 색인과 데이터 메모리에서 삭제를 해야 한다. 이때 객체 A와 객체 B는 R_1과 R_3에 교차되어 있는 상태이다. 이 경우 R_2를 삭제하여 객체 A와 B가 삭제 되는 경우, PDA에서 R_1또는 R_3에 질의하면 객체 1과 객체 3은 캐쉬에 존재하지 않기 때문에 다시 서버와 통신을 해서 질의 결과를 전송받아야 한다. 따라서 객체 A와 객체 B는 R_2를 삭제할 경우 삭제 되어서는 안된다. 이와 같이 영역 단위의 삭제 시에 캐쉬된 다른 영역과 겹치는 객체에 대한 처리 연산을 필요로 한다.

(그림 7) 단일 저장시 교체 영역의 삭제

3.4 중복 객체 처리 방법의 선택 및 문제점

위에서 중복 객체를 처리하기 위한 3가지의 방법에 대해서 기술하였다. 이러한 중복 객체에 대한 처리로 인하여 모바일 클라이언트에서 처리할 연산의 부하는 더 가중되었다. 특히 중복 저장의 경우 작은 저장 공간을 가진 PDA에서 사용하기에는 적절하지 못하다. 또한, 단일 저장의 경우 캐쉬 교체시 영역간에 교차하고 있는 객체에 대한 처리 문제가 복잡하다. 따라서 이 논문에서는 모바일 환경에서 공간 객체의 효율적 검색 및 관리를 위해 클리핑 방법을 적용한다.

모바일 클라이언트(M)에서 영역 질의 Q를 서버(S)에 요청하여 질의 결과(QR, QueryResult)를 저장하기 위한 총 비용 $Cost_{total}(Q)$은 다음과 같다.

$$Cost_{total}(Q) = Cost_M(Trans(Q)) + Cost_s(Q) + Cost_s(Trans(QR)) + Cost_M(Save(QR))$$

$Cost_M(Trans(Q))$는 PDA에서 질의를 서버로 전송하는 비용, $Cost_s(Q)$는 서버에서 질의 영역 R에 대한 질의 처리하는 비용, $Cost_s(Trans(QR))$는 서버에서 처리한 질의 결과를 모바일 클라이언트로 전송하는 비용, $Cost_M(Save(QR))$는 모바일 클라이언트에서 질의 결과를 캐쉬에 저장하는 비용이다. 특히 질의 결과를 캐쉬에 저장하기 위한 비용 $Cost_M(Save(QR))$는 클리핑 방법 적용시 먼저 질의 결과 QR에서 영역 R과 교차하는 객체들 $I^R(QR) = \{x : QR \mid Inter\ sec\ t(x, R)\}$을 검색하여야 한다. $I^R(QR)$의 각 객체 $I_i^R(QR)$은 클리핑 과정을 수행하여 영역내의 객체들 $L^R(QR) = \{x : I^R(QR) \mid cLipping(R, x)\}$로 클리핑 된다. 따라서, 모바일 클라이언트에서 저장하는 데이타는 영역 R에 Contain되는 객체들 $C^R(QR) = \{x : QR \mid Contain(R, x)\}$과 $I^R(QR)$중 클리핑된 객체들 $L^R(QR)$을 저장한다. 따라서 모바일 클라이언트에 질의 결과를 저장하기 위한 $Cost_M(Save(QR))$은 다음과 같이 세부적으로 분류된다.

$$Save(QR) = Inter\ sec\ t(R, QR) + Clipping(I^R(QR)) + Save(C^R(QR)) + Save(L^R(QR))$$

클리핑의 경우 질의 결과로부터 교차하는 객체를 찾아야 한다. 이경우 서버에서 이를 처리하는 경우가 있다. 그러나 다수의 모바일 클라이언트가 서버에 질의를 요청하는 경우 서버는 질의 처리와 질의 결과 전송 작업을 수행해야 한다. 따라서, 서버에서 클리핑 및 Intersect 연산을 추가하는 경우 서버의 부하가 높아지게 되어 서버의 처리율이 떨어지게 된다.

모바일 클라이언트에서 캐쉬에 저장된 공간 데이터의 효율적 검색을 위해서는 공간 색인을 필요로 하고, 서버로부터 데이터를 수신할 때마다 공간 색인을 재구성해야 한다. 따라서 서버로부터 전송되는 데이터는 $Cost_M(Save(QR))$외에 색인 재구성 비용 $Cost_M(IR(QR))$이 추가 된다. 관련연구에서 언급한 것과 같이 추가 되는 객체를 하나씩 삽입할 경우 비용이 높기 때문에 Bulk-Operation을 적용한다. 이 경우 색인 재구성 비용은 질의 결과에 대한 Bulk-Loading [8]비용인 $Cost_M(BL(QR))$과 질의 결과에 대한 공간 색인을 현재 캐쉬된 데이터의 색인에 추가시키는 Bulk-Insertion[2] 비용인 $Cost_M(BI(QR))$으로 구성된다. 따라서 모바일 클라이언트가 추가 검색을 지원하기 위해 서버로부터 전송된 데이터를 저장하기 위한 총 비용은 다음과 같다.

$$Save(QR) = Inter\ sec\ t(R, QR) + Clipping(I^R(QR)) + Save(C^R(QR)) + Save(L^R(QR)) + Cost(BL(QR)) + Cost(BI(QR))$$

각 방법의 장, 단점들을 비교해 볼 때 클리핑 연산을 적용하면 클리핑을 통해서 사용자에게 보이지 않는 불필요한 정보를 저장하지 않음으로써 데이터의 양이 줄어 들며 통신양도 줄어 드는 큰 이점을 가질 수 있다. 그러나 클리핑 방법은 모바일 클라이언트의 저장 비용이 클리핑, 색인 재구성을 포함하여 모바일 클라이언트에 많은 부하를 주고 있다. 이 논문에서는 모바일 클라이언트에서 큰 비용의 연산을 분산하기 위해 에이전트를 사용하여 해결하고자 한다. 이를 에이전트와 분산하여 처리할 경우 다음과 같이 비용 분산 효과를 가져 온다.

$$Agent: Inter\ sec\ t(R, QR) + Clipping(I^R(QR)) + Cost(BL(QR))$$

$$MobileClient: Save(C^R(QR)) + Save(L^R(QR)) + Cost(BI(I(BR)))$$

에이전트에 의한 처리 분산으로 발생되는 통신 비용의 경우 질의 결과중 경계와 교차하는 객체들의 클리핑 결과이

므로 질의 결과보다 작거나 같다. 또한, 모바일 클라이언트에서 Bulk-Insertion을 위해 에이전트는 질의결과에 대한 Bulk-loading 작업이 필요하다. 이것은 질의 결과에 대한 색인 구축이므로 구축된 색인의 전송이 추가로 필요하다.

단일 저장 방법의 경우에는 클리핑 방법과 달리 모바일 클라이언트에 저장된 객체와 질의 결과(QR)사이의 중복 데이터를 제거한 결과 QR'을 생성하기 위한 비용 Re*move*(R, QR)과 QR'에 대한 인덱스 생성비용 $Cost(BL(QR))$으로 구성되고 이를 에이전트와 분산 처리할 경우 다음과 같은 비용 분산효과가 있다.

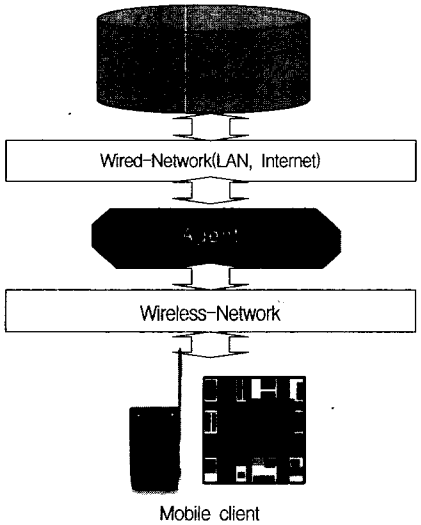
$$Agent : \text{Re}move(R, QR) + Cost(BL(QR'))$$

$$MobileClient : Save(QR) + Cost(BI(QR'))$$

통신 비용의 경우 클리핑방법과 동일하게 질의 결과 데이터보다 작거나 같다. 또한 인덱스의 전송비용도 질의 결과에 대한 인덱스 크기보다 작거나 같다.

4. 에이전트 개념을 기반으로 한 캐쉬 시스템

이 논문에서 제시하는 모바일 환경에서 지도 데이터의 효율적 검색을 지원하는 전체 시스템의 구조는 (그림 8)과 같이 GIS 서버, 에이전트, 모바일 클라이언트로 구성된다. 에이전트와 서버는 유선 네트웍을 통해 연결 되어 있고 에이전트는 모바일 클라이언트와 무선 네트웍을 통해 통신한다. 처리 속도가 상대적으로 느린 모바일 클라이언트의 부하와 무선 네트웍을 통한 통신 양을 줄이기 위한 구조이다.



(그림 8) 시스템 구조

4.1 에이전트

4.1.1 에이전트의 정의 및 기능

에이전트는 많은 의미로 사용되고 있다. 관련 연구[11]에서는 에이전트를 아래와 같이 정의하고 있다.

"Software program that read information stored in agencies and performs some computation based on that information."

이 논문에서의 에이전트는 다음과 같이 정의 한다.

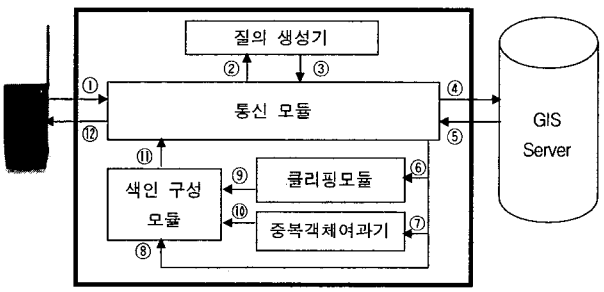
"GIS 서버로부터 공간 데이터를 가져오기 위한 질의를 생성하고 질의의 결과로 가져온 공간 데이터에 대해 PDA에 이미 캐쉬된 공간 객체와 중복된 객체에 대한 처리 연산과 공간 색인을 구성 하는 소프트웨어"

에이전트의 기능은 첫째, 서버로 질의할 질의 영역을 생성한다. 둘째, 질의 결과에 대해 중복 객체를 처리하기 위한 연산으로 클리핑 연산을 수행하거나 중복된 객체를 제거한다. 셋째, R-Tree 공간 색인을 구성하는 작업을 수행한다.

위와 같은 에이전트를 사용하는 목적은 모바일 클라이언트에 집중된 연산의 부하를 감소시키고 컴퓨팅 파워가 우수한 에이전트에서 처리함으로써 모바일 클라이언트의 지도 검색 속도의 향상을 위해서이다. 다시 말하면 문제점에서 언급한 모바일 클라이언트에서 고 비용의 R-Tree 색인 구성 연산과 중복 객체에 대한 처리 연산을 에이전트에서 처리함으로써 모바일 클라이언트의 연산 비용을 분산 시키기 위한 목적이다. 특히 GIS 서버는 다수의 에이전트와 연결할 수 있고, 에이전트는 모바일 클라이언트와 다대일의 관계를 가진다.

4.1.2 에이전트의 구조

에이전트는 (그림 9)와 같이 질의 생성기, 통신 모듈, 클리핑 모듈, 중복 객체 여과기, 색인 구성 모듈로 구성되어 있다. 이 구조는 중복 객체의 3가지 처리 방법에 대한 실험을 위해서 3가지 방법을 모두 수행할 수 있는 구조를 제공한다.



(그림 9) 에이전트 구조

(1) 통신 모듈

모바일 클라이언트와 GIS 서버에 대한 통신 기능을 수행하는 모듈이다. 모바일 클라이언트로부터 사용자 요청 영역과 사용자 요청 영역과 겹치는 캐쉬 영역 정보를 수신하고 생성된 질의를 GIS 서버에 송신하여 질의 결과를 수신한다. 수신된 질의 결과를 처리하여 공간 데이터와 공간 색인

정보를 모바일 클라이언트로 전송한다.

(2) 질의 생성기

모바일 클라이언트에서 받은 사용자 요청 영역과 사용자 요청 영역과 겹치는 캐쉬된 영역들의 정보를 이용하여 PDA에 캐쉬되지 않은 영역을 계산하여 요청할 영역 질의를 생성한다.

(3) 클리핑 모듈

질의 결과 객체 중 질의 영역을 벗어나는 객체에 대해 클리핑 연산[12]을 적용하는 모듈이다.

(4) 중복 객체 여과기

질의 결과 객체 중 이미 모바일 클라이언트에 캐쉬된 객체를 제거하는 기능을 하는 모듈이다. PDA로부터 수신한 사용자 요청 영역과 사용자 요청 영역과 겹치는 캐쉬된 영역 정보를 이용하여 이미 캐쉬된 객체를 여과하는 작업을 수행한다.

(5) 색인 구성 모듈

R-Tree 공간 색인을 구성하는 모듈이다. 서버로부터 수신한 질의 결과 중 클리핑 모듈을 거친 클리핑된 데이터, 중복 객체 여과기를 거쳐 중복 객체를 제거한 데이터에 대해서 R-Tree 공간 색인을 구성한다.

(6) 에이전트의 수행 시나리오

STEP 1, 2 : 모바일 클라이언트에서 사용자 요청 영역, 사용자 요청 영역과 겹치는 캐쉬된 영역 정보를 수신한다.

STEP 3 : 질의 생성기는 서버에 요청할 영역 질의를 생성하여 전달한다.

STEP 4, 5 : 생성된 질의를 서버로 요청하고 결과를 받는다.

STEP 6, 7, 8 : 서버로부터 질의 결과를 수신한다.

STEP 9 : 질의 결과를 클리핑한다(클리핑의 경우).

STEP 10 : 질의 결과 중 중복개체를 제거한다(단일 저장의 경우).

STEP 11, 12 : 구축된 색인 정보와 공간 데이터를 모바일 클라이언트로 전달한다.

4.2 모바일 클라이언트

4.2.1 캐쉬 메커니즘

모바일 클라이언트에서의 캐쉬 단위는 질의 영역이다. 공간 데이터에 대한 캐쉬의 단위는 사용자가 요청한 영역만을 저장하고 캐쉬 되지 않은 영역에 대해서만 질의를 하기 위해서 객체나 페이지 단위가 아닌 질의 영역을 기반으로 한다. 교체 전략으로는 LRU 기법을 기반으로 한다.

질의 영역 단위의 캐쉬를 유지하기 위해서는 캐쉬 영역 정보 관리가 필요하다. 예를 들어 사용자의 영역 질의시 질의 영역의 캐쉬 유무에 대한 검사가 필요하다. 이를 위한 캐쉬 영역 정보 관리에는 색인을 구성하여 유지하는 방법과 단순히 순차적으로 저장하는 방법이 있다. 색인 구성의 경우 캐쉬 요청 영역의 검색시 영역 질의를 수행하기 때문에 영역 질의에 효과적인 R-Tree 계열의 공간 색인을 적용하는 것이 효과적일 것이다. 그러나 PDA에서 캐쉬 영역의 관리를 위해 공간 색인을 유지하는 경우 단점은 다음과 같다. 첫째, 유지 비용이 큰 R-Tree 계열의 공간 색인을 구축하고 유지하는 것은 비효율적이다. 둘째, 캐쉬된 영역의 수가 매우 크지 않는다면 빠른 탐색의 이점보다 트리의 유지 비용이 더욱 커질 것이다. 따라서 이 논문에서는 캐쉬 영역의 관리를 위해서 캐쉬 영역 정보를 순차적 저장을 하고 순차적 검색을 한다. 캐쉬 영역 정보에는 캐쉬 영역의 MBR(Minimum Bounding Rectangle), 교체값 계산을 위한 정보, 데이터의 저장 위치 정보가 저장된다.

모바일 클라이언트에서 캐쉬된 데이터에 대한 효율적인 영역 질의를 위해 R-Tree 공간 색인을 적용한다. 에이전트로부터 전송된 질의 결과는 질의 영역 단위로 추가되고, 질의 영역에 대한 색인 정보는 Bulk-Insertion 연산을 적용하여 기존 색인에 추가된다. 이 경우 질의 영역내의 객체를 하나씩 R-Tree에 삽입하는 방법보다 효율적이다.

저장 공간이 부족할 경우 교체작업이 일어난다. 이때 교체 대상으로 선택된 영역을 삭제하고 메모리를 반환하기 위해서 공간 데이터와 공간 색인의 삭제 연산이 필요하다. 교체 대상 영역의 공간 객체들은 삽입 될 때 질의 영역 단위로 저장하였으므로 빠르게 삭제할 수 있다. 그리고 색인의 삭제 연산은 객체 하나씩 삭제하는 것이 아니라 Bulk-Deletion 연산을 통하여 삭제한다.

Bulk-Deletion 알고리즘은 (그림 10)과 같다. 이 알고리즘은 중복 객체에 대한 처리 방법으로 캐쉬 영역을 클리핑하여 저장하는 것을 전제로 한다.

```
Bulk-Delete ( Node T, Rectangle S )        // S : 삭제될 영역의 MBR
{
    if T != leaf then
        check for each entry E overlap S==TRUE  // 삭제할 영역
                                                // 과 겹치는 하위노드를 찾음
            if E = S || E ⊂ S then   // 하위 노드의 영역
                                     // 이 삭제 영역에 포함될 경우
                Remove E from T ;  // 하위 노드 삭제
                CondenseTree( T )  // 트리 조정
                ShortenTree;
            Else
                Bulk-Delete( E, S )
    Else
        if T = S || T ⊂ S then          // 삭제 영역에 포함될 경우
            Remove T ;                  // 노드 삭제
}
```

```
CondenseTree ( Node T )
{
    CT1 : Set N = T, Set Q, the set of eliminated nodes,
          to be empty
    CT2 : If N is the root, go to CT6,
          otherwise let P be the parent of N, and let En be Ns
          entry in P
    CT3 : If N has fewer than m entries, delete En from P and
          add N to set Q
    CT4 : If N has not been eliminated, adjust EnI to tightly
          contain all entries in N
    CT5 : Set N = P and repeat from CT2
    CT6 : Reinsert all entries of nodes in set Q
          eliminated nodes are re-inserted in tree
          as described in Algorithm Bulk-Insertion
}

ShortenTree
{
          If Root Node has only one child
          Then make child the new Root Node
}
```

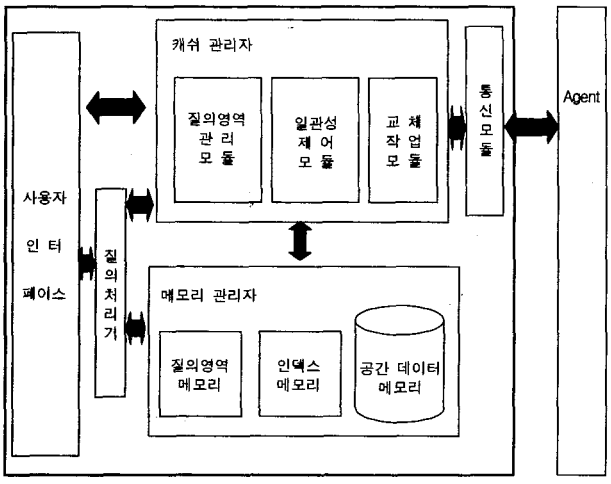
(그림 10) Bulk-Deletion 알고리즘

Bulk-Deletion 알고리즘은 R-Tree의 삭제 알고리즘과 유사하지만 다음과 같은 차이점이 있다. 첫째, R-tree에서 삭제 시 단말 노드의 엔트리만을 찾아 삭제하지만 Bulk-Deletion에서는 중간 노드의 삭제를 허용한다. 둘째, CondenseTree 알고리즘에서 중간 및 단말 노드의 삭제에 의해 underflow가 발생하였을 경우 제거된 노드의 모든 객체들을 재 삽입하는 것이 아니라 제거된 노드를 Bulk-Insertion 알고리즘의 small-tree로 간주하여 Bulk-Insertion 방법으로 재 삽입 연산을 수행하는 것이다.

서버의 공간 객체가 변경될 경우 모바일 클라이언트에 캐쉬된 공간 객체의 일관성 제어가 필요하다. 모바일 클라이언트의 캐쉬의 일관성 유지 기법은 요구시 처리(On-demand)기법과 무효화(invalidation)기법을 적용한다. 요구시 처리 기법은 공간 객체의 변경이 빈번하지 않는 지도 데이터의 특성과 모바일 클라이언트의 단절성의 특성을 고려한 방법으로 모든 클라이언트가 온라인인 것을 요구하는 즉시 변경(immediate update)대신, 모바일 클라이언트에서 일관성 요구가 발생하는 경우에 일관성을 제어하는 방법이다. 무효화 기법은 모바일 클라이언트가 캐쉬한 질의 영역에서 변경된 부분을 무효화시키는 방법으로서 모바일 클라이언트의 일관성 유지의 요청시 캐쉬된 질의 영역의 집합을 전송하여 타임스탬프 이후에 변경된 질의 영역에 대한 응답을 받는다. 변경된 질의 영역은 무효화기법에 따라 색인과 데이터 메모리에서 무효화시킨다.

4.2.2 모바일 클라이언트의 구조

모바일 클라이언트는 사용자 인터페이스, 질의 처리기, 캐쉬 관리자, 메모리 관리자, 통신 모듈로 구성되어 있다.



(그림 11) 모바일 클라이언트의 구조

(1) 사용자 인터페이스

화면상에 지도를 디스플레이하고 사용자의 요청을 받아들이는 모듈이다. 지도의 확대, 축소, 드래깅 방법으로 이용하여 panning 기능을 제공한다.

(2) 질의 처리기

캐쉬 영역에 있는 R-Tree 공간 색인에 대해서 영역 질의를 수행하여 결과를 반환하는 기능을 한다. 캐쉬 관리자의 사용자가 요청한 영역에 대한 질의 요청에 따라 질의 영역과 겹치는 객체를 추출하여 질의 결과를 생성한다. 생성된 질의 결과는 사용자 인터페이스를 통해서 화면상에 보여지게 된다.

(3) 통신 모듈

사용자가 요청한 영역 정보와 캐쉬된 영역 중 사용자가 요청한 영역과 겹치는 영역의 정보를 에이전트에게 송신하고 에이전트로부터 캐쉬되지 않은 영역의 데이터와 공간 색인 정보를 수신한다.

(4) 캐쉬 관리자

1) 질의 영역 관리 모듈

캐쉬된 영역 정보들을 저장하고 관리한다. 서버로 요청할 추가 영역의 질의 생성을 위해 사용자가 요청한 영역과 겹치는 영역 정보를 반환하는 기능을 가진다. 질의 영역은 MBR 형태로 요청되고 결과는 1개이상은 분할된 MBR로서 생성된다. 또한 서버로부터 받아온 질의 영역의 데이터를 공간 데이터 메모리에 저장하고 에이전트에서 이미 구성된 색인 정보를 이용하여 기존의 공간 색인에 추가하는 역할을 한다. 이때 공간 색인에 추가하는 연산은 Bulk-Insertion 기법을 사용한다. 만약 서버로부터 수신한 데이터의 크기 만큼의 빈 메모리 공간이 없는 경우 교체 작업 모듈을 이용하여 필요한 공간 이상의 메모리를 확보한다.

2) 일관성 제어 모듈

모바일 클라이언트에 캐쉬된 공간 객체에 대한 일관성을 제어하는 모듈로서, 질의 영역 관리 모듈로부터 캐쉬된 질의 영역과 캐쉬시 타임스탬프를 에어전트로 전송하는 기능과 에어전트로부터 타임스탬프이후 변경된 질의 영역을 전송받아 변경된 질의 영역을 무효화하는 역할을 한다.

3) 교체 작업 모듈

질의 영역 관리 모듈에서 요구하는 데이터 메모리의 공간을 확보하기 위해 교체 작업을 수행한다. 우선 모든 캐쉬된 영역 중에 LRU 기법을 이용하여 교체 값을 계산한다. 교체 값이 동일한 경우 질의 결과가 큰 영역부터 교체 대상을 선택한 후 교체 영역에 해당하는 정보를 삭제하고, 저장된 교체 영역에 속한 공간 객체들을 공간 데이터 메모리에서 삭제한다. 그리고 인덱스 메모리에서 색인 정보를 제거하여 메모리를 해제하여 필요한 공간 만큼의 메모리가 확보되면 교체 작업을 중단한다. 교체 영역을 제거하기 위해서 Bulk-Deletion 연산을 적용한다. (그림 12)는 이러한 교체 작업 모듈의 시나리오를 나타낸다.

STEP 1 : 캐쉬된 모든 영역에 대해서 교차값 = RP(R)을 구하여 내림차순으로 큐에 삽입한다.
STEP 2 : 큐의 첫번째 영역에 대해서 요구한 크기가 만족할때까지 STEP 3~6 과정을 반복한다.
STEP 3 : 영역의 데이터 메모리를 해제한다.
STEP 4 : Bulk-Delete()를 이용하여 해당 영역을 색인에서 삭제한다.
STEP 5 : 영역 정보를 삭제한다.
STEP 6 : 큐에서 영역을 삭제한다.

(그림 12) 교체 작업 시나리오

(5) 메모리 관리자

1) 질의 영역 메모리

캐쉬된 영역에 대한 정보를 저장하는 메모리 영역이다. (그림 13)과 같이 하나의 영역에 대해서 영역의 MBR, 최근 사용 시간, 영역의 데이터의 저장 위치 정보, 그리고 질의 영역의 캐쉬시 타임스탬프로 구성된다.

MBR	Time	Data Pointer	TimeStamp

(그림 13) 질의 영역 메모리 저장 구조

2) 인덱스 메모리

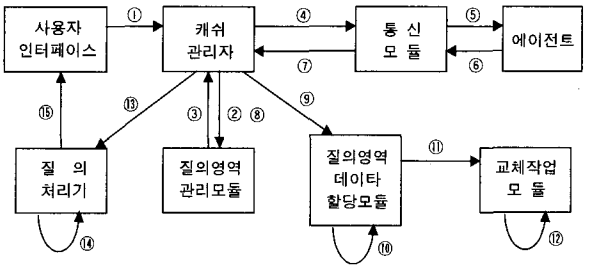
캐쉬된 전체 공간 데이터의 R-Tree 공간 색인 정보를 저장하는 메모리 영역이다.

3) 공간 데이터 메모리

캐쉬된 공간 데이터를 저장하는 메모리 영역이다. 캐쉬 영역 별로 저장되어 있다.

(6) PDA의 수행 시나리오

(그림 14)는 모바일 클라이언트의 구조를 수행 시나리오에 따라 나타낸 그림이다. 수행 시나리오는 다음과 같다.



(그림 14) PDA 수행 시나리오

STEP 1 : 사용자 요청 영역 정보를 전달한다.
STEP 2, 3 : 현재 캐쉬된 영역 중 사용자의 요청 영역과 겹치는 영역 정보를 요청하고, 결과를 전달받는다.
STEP 4, 5 : 사용자 요청 영역과 캐쉬된 영역 중 사용자의 요청 영역과 겹치는 영역 정보를 에이전트에 전달한다.
STEP 6, 7 : 새로운 캐쉬 영역의 데이터와 색인 정보를 전달 받는다.
STEP 8 : 새로운 캐쉬 영역의 영역 정보의 저장한다.
STEP 9 : 새로운 캐쉬 영역의 데이터와 색인 정보의 삽입을 요청한다.
STEP 10 : 데이터 메모리의 유효 공간을 조사한다.
STEP 11 : 저장 공간이 부족할 경우 데이터 메모리 교체를 요청한다.
STEP 12 : 교체 작업을 수행한다.
STEP 13 : 영역 질의를 요청한다.
STEP 14, 15 : 영역 질의를 수행하고 결과를 전달한다.

4.3 GIS 서버

공간 데이터를 저장하고 있는 서버이다. 고유의 인터페이스를 통해 지리 정보를 제공한다. GIS 서버는 공간 데이터에 대한 영역 질의의 결과를 효율적으로 생성하기 위해 공간 데이터의 인덱스를 구성할 수 있다. 그리고 에이전트로부터 요청된 질의 영역에 대해 벡터 형태의 결과를 생성하여 에이전트로 전송한다. 또한 일관성 제어를 위하여 에이전트로부터 전송된 타임스탬프를 가진 영역내에 변경 여부를 검사하여 에이전트로 전송한다.

5. 구현 및 성능 평가

이 장에서는 이 논문에서 제시한 캐쉬 구조를 기반으로 각 모듈의 구현과 구현을 위한 구축 환경을 설명한다. 그리

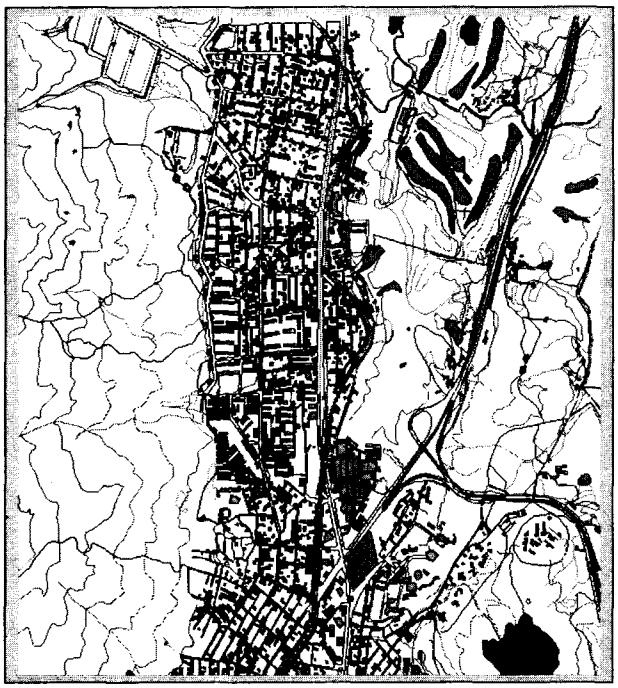
고 구현된 시스템의 동작 예제 및 중복 객체에 대한 처리 방법 3가지 경우에 대한 실험 결과를 비교 분석한다.

5.1 환 경

이 논문의 구현을 위해서 모바일 클라이언트는 Palm IIIc Palm OS 3.5 ROM을 탑재한 에뮬레이터를 사용하였다. 에이전트는 Windows 2000 환경에서 128M Pentium III 866을 사용하였으며, GIS 서버는 Linux 환경 128M Pentium II 266을 사용하는 파일 시스템 기반의 서버를 사용하였다.

(1) 실험 데이터

실험 데이터는 레이어 구조로 분류된 실제 도심의 벡터 지도를 이용하였다. 그 중 데이터의 양이 가장 많은 4가지 레이어에 대해서 실험하였다. (그림 15)는 실험 대상 영역을 보여 준다.



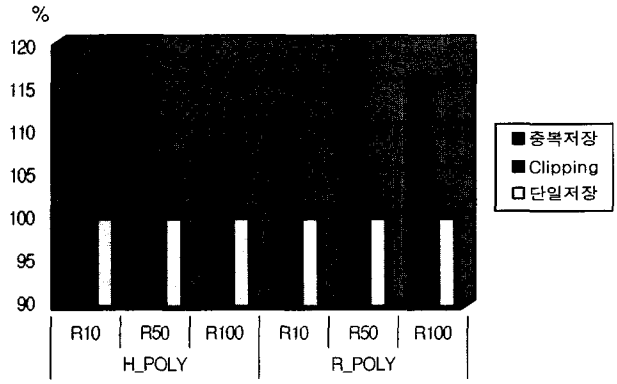
(그림 15) 실험 대상 영역

5.2 성능 평가

이 실험은 모바일 클라이언트에서 캐쉬 영역 사이의 중복 객체 처리의 방법들을 비교 분석하기 위해 각 처리 방법에 대해서 처리 시간과 저장 공간의 비교와 캐쉬의 성능 실험을 한다.

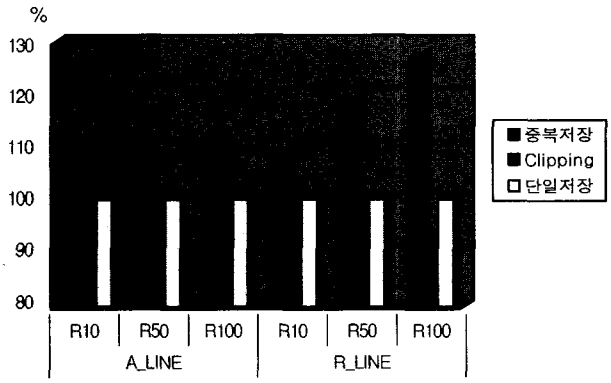
(1) 저장 공간에 대한 실험

모바일 클라이언트에서 영역 요청시 에이전트에서 각 중복 객체의 처리 방법들에 의해 생성되는 데이터양을 측정한다. (그림 16)은 H_POLY와 R_POLY의 면 레이어에 대해 실험한 결과이다. 질의 형태는 가로, 세로가 10m, 50m, 그리고 100m에 대한 영역 질의로서 100회 수행후 평균값을 나타낸다. 단일 저장하는 경우를 기준으로 중복 저장하는 경우와 클리핑할 경우의 데이터의 양의 비율을 그래프로 나타내었다. 면 레이어의 객체들이 단순한 기하 모양을 가지고 있어 클리핑을 했을 때 단일 저장 방법보다 데이터의 양에서 이점이 없다.



(그림 16) 중복 객체 처리 방법들의 데이터 양 비교-면

그러나 (그림 17)은 A_LINE과 R_LINE의 두 선 레이어에 대한 실험 결과를 보면 클리핑을 했을 때 데이터의 양이 중복 저장에 비해 평균 45%, 단일 저장에 비해 60%이하로 현저히 줄어든 것을 볼 수 있다. 중복 저장은 다른 두 방법에 비해 많은 저장 공간을 요구한다는 것을 알 수 있다. 따라서 에이전트에서 전송되는 데이터량의 감소로 인한 전송 비용 및 저장 비용의 효과가 클리핑 방법이 우수함을 입증한다.

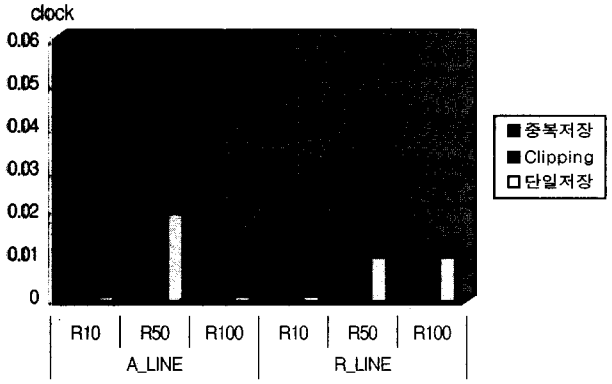


(그림 17) 중복 객체 처리 방법들의 데이터 양 비교-선

(2) 중복 객체 처리 시간에 대한 실험

중복 객체 처리 방법들의 데이터 양을 비교한 실험과 병행하여 각 방법들의 처리 시간을 측정하였다. (그림 18)은 영역별 질의시 중복 객체 처리 방법별 실험 결과 그래프이다. 클리핑의 경우가 다른 방법들에 비해 더 많은 시간이 소요되는 것으로 측정되었지만, 세로축의 측정 단위가 clock

단위이기 때문에 질의 수행에서 에이전트에서의 부하는 아주 미비하다. 따라서 모바일 컴퓨터보다 처리속도가 우수한 에이전트에서 중복 객체를 처리함으로써 중복 객체의 처리 연산은 전체 시스템에 거의 부하를 주지 않는다는 것을 보여준다.



(그림 18) 중복 객체 처리 방법별 처리 시간

즉, 에이전트에서 처리하는 질의 생성, 중복 객체 처리, 패킷 전송 시간의 처리 시간은 모바일 클라이언트의 작업 시간과 비교할 경우 3% 이내의 무시 가능한 시간이었다. 따라서 모바일 클라이언트로 전송되는 데이터 양을 줄이기 위해 클리핑 연산을 수행하고, 이를 에이전트에서 처리함으로서 전체 성능의 감소를 수반하지 않는 장점을 가진다.

5.3 실험 결과 분석

이 실험에서 단순 공간 객체가 아닌 복합 공간 객체의 검색의 경우 클리핑 방법을 사용하여 중복 객체를 처리 경우에 다른 방법에 비해 데이터의 양이 많이 줄어든다는 것을 알 수 있다. 따라서 복합 공간 객체의 경우 클리핑 방법이 통신 비용감소 효과가 있고 모바일 클라이언트의 저장 공간의 활용도 좋아지게 된다. 그리고 처리 시간의 부하는 모바일 클라이언트가 아닌 에이전트가 처리함으로써 전체적인 질의 검색시간을 향상시킬 수 있다. 따라서 도로 레이어와 같이 선 객체가 많은 공간 데이터의 경우 중복 객체 처리를 위해 클리핑을 하는 것이 저장 공간과 처리 시간에 대한 실험 결과를 고려해 볼 경우 효과적인 방법임을 알 수 있다.

6. 결론 및 향후 연구

무선 통신이 가능한 모바일 클라이언트에서 지도 서비스를 하기 위한 에이전트 기반의 캐쉬 시스템 구조를 설계 및 구현하였다. 제한된 저장공간을 고려하여 사용자가 요청한 데이터만을 캐쉬하기 위해 질의 영역 단위의 시맨틱 캐쉬를 적용하였고, 캐쉬된 데이터의 공간 색인에 새로운 캐쉬 영역의 빠른 삽입과 삭제 연산을 위해서 Bulk-Operation을 이용하였다. 또한 컴퓨팅 파워가 낮은 모바일 클라이언트에 집중된 연산을 에이전트를 이용하여 분산 처리하기 위한 구조를 제시하여 질의 응답시간을 단축하였다. 또한, 모바일 클라이언트에서 임의의 영역 요청에 따라 중복 객체를 처리하기 위해 중복 저장, 클리핑, 단일저장 방법에 대한 데이터의 양과 처리 시간을 측정하고 분석을 통해, 복잡한 데이터의 경우 클리핑 연산을 하는 것이 효과적임을 알 수 있었다.

이 논문은 질의 영역 단위를 캐쉬 단위로 하기 때문에 사용자의 질의 패턴에 따라 캐쉬 영역이 작고 복잡해지면 캐쉬 영역 관리 비용이 증가하고 서버로 질의할 영역 계산이 복잡해 지는 제약점을 가진다. 향후 이러한 관리 비용을 줄일 수 있는 캐쉬 구조를 설계하고 캐쉬된 데이터에 대한 보다 효율적인 일관성 유지에 관한 연구가 향후 필요할 것이다. 또한 지역별 관리 에이전트를 두어 다수의 모바일 클라이언트에 서비스 할 수 있는 구조와 에이전트에서의 캐쉬관리 기법에 대한 연구도 필요하다.

참 고 문 헌

[1] A. Guttman, R-Trees : A dynamic index structure for spatial searching, Proceedings of SIGMO, pp.47-57, 1984.

[2] Li Chen, Rupesh Choubey, Elke A. Rundensteiner, "Bulk-insertions into R-Trees Using the Small-Tree-Large-Tree Approach," ACM-GIS pp.161-162, 1998.

[3] Rupesh Choubey, Li Chen, Elke A. Rundensteiner, "GBI : A Generalized R-Tree Bulk-Insertion Strategy," SSD pp.91-108, 1999.

[4] Donald Kossmann, Michael J. Franklin, Gerhard Drasch, Wig Ag, "Cache Investment : Integrating Query Optimization and Distributed Data Placement," ACM Transactions on Database Systems (TODS), Vol.25, Issue.4, pp.517-558, Dec., 2000.

[5] Shaul Dar, Michael J. Franklin, Bjorn T. Jonsson, Divesh Srivastava and Michael Tan, "Semantic Data Caching and Replacement," In Proceedings of the 22nd Very Large Data Bases, 1996.

[6] Qun Ren, Margaret H. Dunham, "Using semantic caching to manage location dependent data in mobile computing," Proceedings of the 6th annual international conference on Mobile computing and networking, pp.210-221, August, 2000.

[7] E. J. ONeil, P. E. ONeil and G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering," In Proceedings of the ACM SIGMOD International Conference on Management of Data, May, 1993.

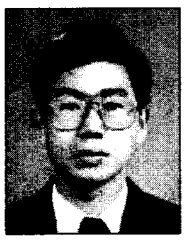
[8] Van den Berchken, Seeger, Widmayer, "A General Approach to Bulk Loading Multidimensional Index Structures," Int Conf. on Very Large Databases, VLDB 1997.
[9] Kamel, I., Faloutsos, C., "On Packing R-Trees," Proc 2nd Int. Conf on Information and Knowledge Management (CIKM), pp.490-499, 1993.
[10] Leutenegger, S. T., Edgington, J., Lopez, M. A., "Efficient Bulk Loading of R-Trees," Univ. of Denver, Technical Report 95-1.
[11] Hsieh-Chang Tu, Miehael L. Lyu and Jieh Hsiang, Agent Technology for Website Browsing and Navigation. Proceeding of 32nd International Conference on System Science 1999.
[12] Tina M. Nicholl , D. T. Lee , Robin A. Nicholl, "An efficient new algorithm for 2-D line clipping : Its development and analysis," ACM SIGGRAPH Computer Graphics, Proceedings of the 14th annual conference on Computer graphics, August, 1987.
[13] Volker Gaede, Oliver Gunther, "Multidimensional Access Methods," ACM Computing Surveys, 30(2), pp.170-231, 1998.
[14] 조대수, 안경환, 홍봉희, "인터넷 지리정보서비스의 성능 개선을 위한 클라이언트 캐쉬 알고리즘", '99 한국데이터베이스 학술대회논문집, 제15권 제1호, 1999.
[15] 서영덕, 박영민, 전봉기, 홍봉희, "디클러스터된 공간 데이터베이스에서 다중 질의의 병렬 처리", 한국정보과학회논문지, 제29권 제1호, pp.44-57, 2002.

임 덕 성

e-mail : dsleem@pusan.ac.kr
1998년 동아대학교 컴퓨터공학과(공학사)
2000년 부산대학교 컴퓨터공학과 (공학석사)
2000년~현재 부산대학교 컴퓨터공학과 박사과정
관심분야 : 모바일 GIS, 이동체 DB, 이동체 색인

이 재 호

e-mail : ljh63359@etri.re.kr
2000년 동아대학교 컴퓨터공학과(공학사)
2002년 부산대학교 컴퓨터공학과 (공학석사)
2002년~현재 한국전자통신연구소 연구원
관심분야 : 이동객체, LBS



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 DB, 이동체 DB, 개방형 GIS