다중 무선 방송채널에서 kNN 질의 처리를 위한 R-tree 인덱스 스케줄링 기법

(An R-tree Index Scheduling Method for kNN Query Processing in Multiple Wireless Broadcast Channels)

정 의 준 † 정 성 원 ††
(Euijun Jung) (Sungwon Jung)

요 약 본 논문은 다중 무선 방송채널환경에서 R-tree를 이용하여 kNN 질의처리의 효과적인 인덱스 스케줄링 기법에 관한 논문이다. 기존 방식은 kNN질의처리 시 노드를 여러 개 얻어야 할 때 child들이 다중 채널 방송스케줄 상 같은 타임 슬롯에 위치하고 있어 원하는 데이터를 얻기 위해서 다음 사이클로 넘어가 데이터를 얻는 시간이 길어지는 단점이 존재하였다. 제안하는 방법은 방송채널에 인덱스 스케줄링을 하기 전에 kNN을 수행하여 R-tree의 각 노드의 child의 접근빈도를 구한 후 구해진 접근 빈도를 기반으로 인덱스 스케줄링 시 방문이 많이 되어 접근빈도가 높은 child들을 다중채널 상에 직렬로 할당하고 접근이 적게 되는 노드는 병렬로 할당하여 질의처리 시 각 노드의 child들을 탐색할 때 겹치는 부분을 줄여 사용자가 원하는 데이터를 빠르게 얻을 수 있는 인덱스 스케줄링 기법이다.

키워드 : 모바일 데이터베이스, 다중 무선 방송채널, 위치기반서비스 질의, KNN 질의어, R-tree

Abstract This paper proposes an efficient index scheduling technique for kNN query processing in multiple wireless broadcast channel environment. Previous works have to wait for the next cycle if the required child nodes of the same parent node are allocated in the same time slot on multiple channel. Our proposed method computes the access frequencies of each node of R tree at the server before the generation of the R-tree index broadcast schedule. If they have high frequencies, we allocate them serially on the single channel. If they have low frequencies, we allocate them in parallel on the multiple channels. As a result, we can reduce the index node access conflicts and the long broadcast cycle. The performance evaluation shows that our scheme gives the better performance than the existing schemes.

Key words : Mobile databases, Multiple wireless broadcast channels, LBS query, KNN query, R-tree

1. 서 론

최근에 GPS나 네비게이션 기기들에 대한 사용자들의 이용이 폭발적으로 증가하면서 위치기반 서비스(Location Based Service: LBS)에 대한 많은 연구가 진행되고 있다. 기존의 위치기반 서비스에 관한 연구들은 대부분 로컬에서 진행되는 디스크 기반의 연구들이 많았다[1-5]. 하지만 디스크 기반의 위치기반 서비스는 부족한 모바일 리소스에 공간 데이터에 대한 정보를 저장해야 돼서 다양한 서비스에 대한 적용이 힘든 것과 서버로부터의 빠른 업데이트를 하기 힘들다는 단점이 있었다. 방송기법을 위치기반 서비스에 이용하게 되면 클라이언트의 수에 제한이 없기 때문에 불특정 다수의 사용자가 원하는 데이터를 빠르게 얻을 수 있고 서버에 의해서 업데이트 되는 내용이 바로 반영되어 위치기반 서비스를 이용하는 사용자는 항상 최신의 데이터를 얻을 수 있고 다양한 위치기반 서비스에 대한 적용이 용이해진

† 정 회 원 : 서강대학교 컴퓨터공학과
jungej@sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터공학과 교수
jungsung@sogang.ac.kr
논문접수 : 2009년 10월 12일
심사완료 : 2010년 1월 20일

정보과학회논문지: 데이타베이스 제37권 제2호(2010.4)

다. 또한 대용량의 공간 데이터를 제한된 용량을 갖는 모바일 기기에 저장할 필요없이 방송채널에 접속하여 데이터를 얻기 때문에 제한된 용량의 모바일 기기의 단점을 극복할 수 있다.

이런 방송기법의 장점을 이용하여 다수의 사용자들에게 효과적으로 위치기반 서비스를 제공하기 위하여 여러 가지 위치기반 질의처리 연구가 진행 되었다. 그 중 대표적인 위치기반 질의처리 방법으로는 사용자가 지정한 포인트 p로부터 가장 가까운 k개의 위치기반 데이터를 찾게 해주는 kNN쿼리(k Nearest Neighbor)가 있다. 그동안 진행된 kNN 질의처리를 방송환경에 적용시켜 구현한 연구로는 R-tree를 기반으로 한 연구와 힐버트 곡선(Hilbert curve)를 기반으로 한 연구가 진행되어 왔다[6-10]. 방송환경에서 R-tree를 기반으로 한 kNN연구와 힐버트 곡선(Hilbert curve)를 기반으로 한 kNN연구들은 대부분 단일 방송채널 환경에서 진행되어 왔다. 최근에 다중 방송 채널을 이용하여 kNN 질의처리를 하는 연구가 있었다[10]. 다중 채널을 이용함으로써 얻을 수 있는 가장 큰 장점은 여러 채널을 사용함으로써 데이터나 인덱스 정보가 여러 채널에 분산되어 할당됨으로써 전체 방송 스케줄의 길이가 짧아져서 전체 질의처리 시 드는 소요시간을 줄일 수 있게 된다. [10]에서는 다중채널에서 R-tree기반 인덱스를 할당함으로써 kNN 질의 처리 시 드는 시간을 최소화 하고, 할당 시 같은 레벨에 존재하는 노드들의 MBR이 겹치는 부분(overlap)을 없애면서 R-tree기반으로 인덱스를 할당하는 MWIS 인덱스 스케줄링을 제안하였고, 각 R-tree노드의 하위에 존재하는 데이터 개수정보를 이용하여 kNN질의처리를 수행하는 연구를 하였다. 하지만 kNN 질의처리 시 같은 레벨에 존재하는 노드들의 MBR이 겹친다고 해서 겹치는 노드를 모두 탐색해야만 한다는 보장이 없기 때문에 같은 레벨의 노드들의 MBR이 겹치는 부분을 없애는 스케줄링은 질의처리 시 소요되는 시간을 크게 줄이지는 못하였고, R-tree의 변형인 R+-tree같은 경우는 겹치는 부분이 존재하지 않기 때문에 다양한 R-tree의 변형에 적용되기는 힘든 단점이 있었다.

본 논문에서는 R-tree가 다중 방송 채널에 할당 될 때 노드의 child들이 다중채널 상의 같은 타임 슬롯에 위치해서 다음 사이클로 넘어가야 되는 경우를 줄일 수 있게 각 노드의 child들을 직렬로 할당함으로써 다음 사이클로 넘어가는 문제를 줄이는 인덱스 스케줄링 방법을 제안한다. 단, 노드의 child들을 직렬로 할당할 경우 기존 방식 보다 전체 방송 스케줄 길이가 길어 질 수 있기 때문에 다중 채널에 할당하기 전에 kNN 질의처리를 n번 수행 후 child들의 방문된 접근빈도를 구하여 이를 기반으로 각 노드의 child들 중에 접근빈도가 높은 child만을 직렬로 할당하고 접근빈도가 낮은 child들은 병렬할당을 하여 방송 스케줄 길이가 길어지는 문제점을 줄여 전체 kNN질의처리 시간을 줄이는 접근빈도 기반 인덱스 스케줄링 기법을 제안한다.

이후의 본 논문의 구성은 다음과 같다. 2장에서는 접근빈도를 이용한 R-tree 기반 인덱스 스케줄링 알고리즘을 설명한다. 3장에서는 성능 평가를 통해 제안하는 방법을 성능을 증명하고 4장에서는 성능평가에 따른 제안하는 방법에 대한 결론을 내린다.

2. 접근빈도를 이용한 인덱스 스케줄링

2.1 기본 아이디어

제안하는 방법의 인덱스 스케줄링을 설명하기 전에 두 가지 용어 정의가 필요한데 해당 용어들은 다음과 같다. AccessFrequency는 서버 측에서 다중채널에 인덱스를 할당하기 전에 접근빈도가 높은 child들만을 직렬로 할당하기 위한 정보로 사용되게 된다. AccessFrequency는 서버 측에서 다중 방송채널에 인덱스를 할당하기 전에 kNN 질의처리를 실행하여 구해지게 된다. 본 논문에서는 기존 R-tree의 노드 구조에 추가적인 정보를 포함하여 인덱스 스케줄링 시에 따로 정보를 참고할 필요 없이 R-tree를 탐색하는 것만으로도 각 노드의 child의 접근빈도를 얻을 수 있게 하였다. conflict는 노드의 child들이 다중채널 상 같은 타임 슬롯에 위치하여서 한 번에 얻지 못하고 다음 사이클로 넘어가는 현상을 말한다.

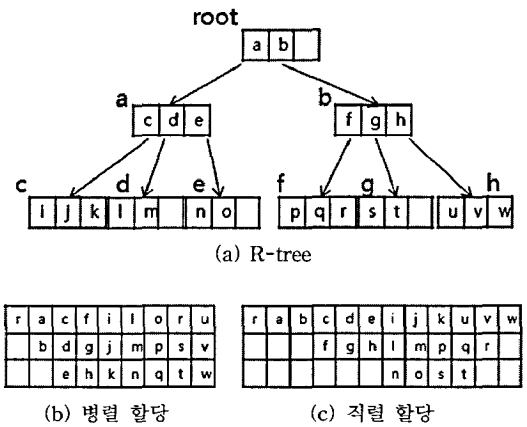
정의 1. 병렬 할당은 각 노드의 child의 개수가 n이고 주어진 다중 방송채널이 c개 일 때, 주어진 채널 개수 c에 맞게 동일한 타임 슬롯의 1번 채널부터 c번 채널까지 채널이 증가하는 순으로 할당되는 것을 말한다. 이 때 R-tree의 부모-자식 관계를 지키기 위해서 앞서 할당한 child의 레벨보다 뒤에 할당한 레벨이 높다면 마지막으로 병렬 할당한 child의 뒤의 타임 슬롯부터 할당이 시작된다.

정의 2. 직렬 할당은 노드의 child 개수가 n개이고 타임 슬롯이 i일 때 동일한 채널 내에 타임 슬롯을 하나씩 증가하며 i번째 타임 슬롯부터 i+n 번째 타임 슬롯까지 차례로 할당하는 것을 말한다. 직렬 할당은 각 노드의 child들만 주어진 채널에서 타임 슬롯 순서만 증가하면서 직렬로 할당하는 것을 의미하는 것이기 때문에 같은 레벨에서 부모가 다른 노드의 child끼리는 병렬 할당이 될 수 있다.

각 노드의 child들을 다중 채널에 직렬로 할당하게 되면 해당 child들을 여러 개 얻어야 할 때 conflict를 줄임으로써 다음 사이클로 넘어감으로써 생기는 레이턴시를 줄일 수 있는 장점을 얻을 수 있다. 하지만 각 노드

의 child들을 모두 직렬로 다중 방송채널에 할당하게 되면 전체 방송 스케줄의 길이가 길어지는 단점이 생기게 된다. 그림 1에서 채널이 3개 주어지고 R-tree가 BFS 순서로 탐색될 때 병렬로 할당한 스케줄(b)와 직렬로 할당한 스케줄(c)가 있다. 그림 1의 (c)와 같이 각 노드의 child를 모두 직렬로 할당하게 되면 병렬로 할당한 (b)와 비교했을 때 보다 방송 스케줄이 늘어나게 되는 단점이 생기게 된다. 본 논문에서는 각 노드의 모든 child들을 직렬로 할당하는 것이 아닌 각 노드의 child들 중 방문되는 빈도가 높은 child만을 직렬로 할당하여 노드의 모든 child들을 직렬로 할당했을 때 방송 스케줄의 길이가 늘어나는 문제점을 보완하려고 한다. 제안하는 방법으로 R-tree기반 인덱스 스케줄링은 크게 두 단계로 나누어진다. 첫 번째 단계는 R-tree의 각 노드의 child들의 접근빈도를 구하기 위해 kNN을 실행하여 접근빈도를 구하는 것이고, 두 번째 단계는 구해진 접근빈도를 기반으로 R-tree를 탐색해 나가면서 serialBase를 기준으로 다중 방송 채널에 할당하는 것이다.

첫 번째 단계인 R-tree의 각 노드의 child들에 접근빈도를 구하기 위해서 서버 측에서 kNN을 주어진 R-tree에 수행하게 된다. kNN 질의처리 시 총 n번을 수행하게 되는데 공간 데이터의 분포에 따라 고르게 n번의 kNN질의처리를 하였다. 본 연구에서 접근빈도 저장을 위한 자료구조를 만들지 않고 기존 R-tree의 각 노드의 child에 접근빈도를 저장하기 위해 AccessFrequency 정보를 추가하였다.



(a) R-tree

(b) 병렬 할당 (c) 직렬 할당

그림 1 R-tree를 병렬로 할당과 직렬로 할당한 경우

2.2 접근빈도를 이용한 인덱스 스케줄링

두 번째 단계는 구해진 접근빈도를 바탕으로 다중 채널에 인덱스를 할당하는 단계이다. 기본적인 R-tree 탐색순서는 BFS순서로 노드를 탐색하며 다중 채널에 할당되고 할당 시 루트노드부터 리프 노드까지 순서대로 탐색이 가능해야 하므로 child 노드는 항상 부모 노드의 레벨보다 뒤에 할당된다[5]. 각 노드의 child들을 다중 방송 채널에 할당하기 위해서 2개의 큐가 필요한데 첫 번째는 직렬로 할당할 child들을 저장하는 serialQueue이고 두 번째는 병렬로 할당할 child들을 저장하는 parallelQueue이다. 추가적으로 R-tree를 BFS순서로 탐색하기 위한 BFSqueue가 사용된다. serialQueue와 parallelQueue에 대한 정의는 다음과 같다.

정의 3. serialQueue는 각 노드의 child 중 child의 접근빈도가 serialBase보다 같거나 높아서 다중 채널에 직렬로 할당되기 위해 삽입되는 큐이다.

정의 4. parallelQueue는 각 노드의 child 중 child의 접근빈도가 serialBase보다 낮아서 다중 채널에 병렬로 할당되기 위해 삽입되는 큐이다.

R-tree의 루트부터 시작하여 할당이 시작되면 노드의 child들의 접근빈도를 확인한다. 만약 접근 빈도가 직렬 판단 기준인 serialBase 보다 높거나 같으면 serialQueue에 삽입하고 serialBase보다 낮으면 parallelQueue에 삽입한다. 본 논문에서는 기본적으로 serialBase를 전체 접근빈도의 평균을 기준으로 하여 직렬에 대한 판단을 기준을 한다.

큐에 삽입이 끝나면 serialQueue와 parallelQueue의 개수에 따라 세 가지 케이스로 나누어지게 되는데 각

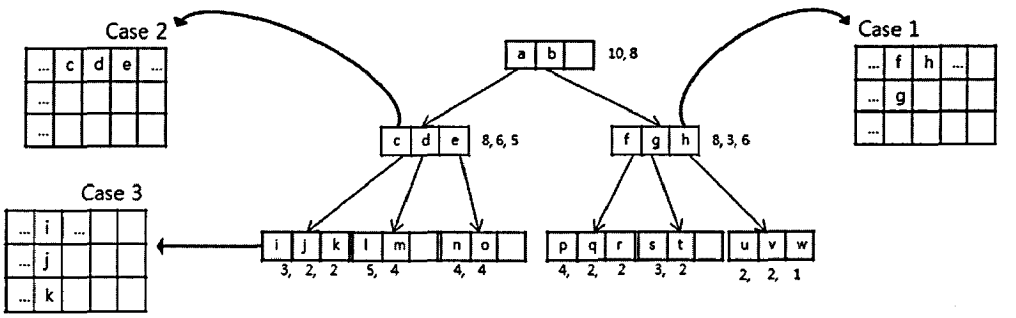


그림 2 총 10번 kNN후 평균접근빈도가 4일 때 각 Case에 따른 할당 예

세 가지 케이스는 다음과 같고 그림 2에 세 가지 케이스에 대한 할당 예가 나와 있다.

Case 1. 직렬로 할당할 *child*와 병렬로 할당할 *child*가 모두 있는 경우: serialQueue의 개수가 2개 이상이고 parallelQueue가 1개 이상이면 직렬, 병렬 두 가지로 모두 구분하여 할당한다.

Case 2. 모두 직렬로 할당하는 경우: serialQueue의 개수가 2개 이상이고 parallelQueue의 개수가 0이라면 같은 병렬로 할당할 개수가 없기 때문에 모두 직렬로 할당한다.

Case 3. 모두 병렬로 할당하는 경우: serialQueue의 개수가 1개 이하라면 해당 노드의 child들은 모두 병렬로 할당한다.

접근빈도를 이용하여 serialBase 기준으로 직렬로 할당할 child가 들어있는 serialQueue와 병렬로 할당할 child가 들어있는 parallelQueue로 나누어지면 실제 다중 방송채널에 할당을 시작하게 된다.

그림 3은 다중채널에 인덱스를 할당하는 SPAF 알고리즘이다. 주어진 다중채널의 개수가 n이라고 할 때 채널의 할당순서는 1번부터 n번까지 오름차순으로, BFS 순서로 노드를 탐색하면서 채널에 할당하게 된다.

```
Input : R-tree with AccessFrequency, serialBase
Output : multi-channel broadcast schedule

Algorithm SPAF-schedule(root, serialBase)
Make new node r that contains root's child information
Allocate r
BFSqueue.add(root);
while ( BFSqueue ≠ ∅ ) do
{   temp = Pop first node from BFSqueue
    if ( type of temp is Intermediate ) then
    { Push temp's child to BFSqueue }
    for ( each child in temp ) do
    {   if ( child's AccessFrequency >= serialBase ) then
        { Push ith child to serialQueue }
        else
        { Push ith child to parallelQueue } }
    if (serialQueue.size >= 2) && (parallelQueue.size >= 1) then
    {   for ( each child in serialQueue ) do
        { FindPosition() Allocate ith child and increase currentTimeslot }
        for ( each child in parallelQueue ) do
        { FindPosition() Allocate ith child and increase currentChannel } }
    else if ( serialQueueSize <= 1 ) then
    {   for each child in both queue do
        { FindPosition()   Allocate child and increase currentChannel } }
    else if (serialQueue.size >= 2) && (parallelQueue.size == 0) then
    {   for ( each child in serialQueue )do
        { FindPosition()   Allocate child and increase currentTimeslot } }
}end while

Algorithm FindPosition()
{  if ( level of last allocated node is lower than level of current node ) then
   {  set currentTimeslot to bigger value between lastCurrentTimeslot
              and lastSerialTimeslot in  last level increase currentTimeslot }
   if ( current child is first child from serialQueue ) then
   { store serialStartTimeslot as currentTimeslot }
   if ( current child is first child and
              there was serial allocation in last queue )then
   { set currentTimeslot as serialStartTimeslot }
   if ( position at currentChannel and currentTimeslot is not empty ) then
   { increase currentChannel }
   if ( currentChannel == chNum ) then
   { increase currentTimeslot and set currentChannel to 1
          find first empty position from currentTimeslot and channel 1 }
}
```

그림 3 SPAF 인덱스 스케줄링 알고리즘

SPAF-schedule 알고리즘이 시작되면 방송채널에서 루트 노드의 child들임을 알 수 있게 하기 위해서 루트 노드의 child정보를 갖고 있는 r을 생성하고 루트노드의 child를 r의 child로 삽입 후 방송채널에 할당한다. 다음 R-tree를 BFS순서로 탐색하기 위해 R-tree의 루트노드를 BFSqueue에 삽입한다. 다음 while 문에서는 BFSqueue가 빌 때까지 while문을 반복하여 실행하게 된다. while 문이 시작되면 BFSqueue의 제일 첫 번째 노드를 꺼내어 temp에 저장한다. temp는 BFSqueue의 노드를 저장하기 위한 임시 노드이다. 만약 temp에 저장된 노드가 중간 노드(intermediate node)라면 하위 child 노드를 가리키고 있으므로 BFS 순서의 탐색을 위해 child 노드를 BFSqueue에 삽입한다. 다음으로 현재 노드를 가리키는 temp 노드의 모든 child를 검사하여 serialBase보다 접근빈도가 크거나 같은 child는 직렬로 할당하기 위해 serialQueue에 삽입하고 접근빈도가 serialBase보다 작은 child는 parallelQueue에 삽입한다. temp의 모든 child가 serialQueue와 parallelQueue에 삽입이 끝난 후에 앞서 설명한 세 가지 경우에 따라서 다중 방송 채널에 할당을 시작한다. 즉, 직렬로 할당할 child와 병렬로 할당할 child가 모두 있는 경우, 직렬로 할당할 child만 있는 경우, 병렬로 할당할 child만 있는 경우로 나누어서 다중 방송채널에 할당하게 된다. 직렬과 병렬 모두 할당할 child가 있는 경우는 직렬로 할당할 child가 저장된 serialQueue를 할당 후에 병렬할당을 수행하게 된다.

직렬로 할당할 child와 병렬로 할당할 child가 정해지면 두 큐의 child들을 다중 방송 채널에 할당하게 된다. 다중채널에서의 할당할 위치의 결정은 FindPosition()함수가 수행한다. FindPostion()함수는 각 노드의 레벨에 따라, 노드의 child의 직렬, 병렬 할당에 따라 다르게 다중 채널에 할당할 적절한 위치를 찾게 된다. 전에 할당했던 노드와 현재 할당하려는 노드의 레벨이 틀리다면 R-tree의 부모-자식 관계를 지키기 위해 현재 타임 슬롯과 이전 레벨 직렬할당을 한 타임 슬롯 중에 큰 타임 슬롯 값이 현재 타임 슬롯으로 세팅되고 1을 증가시킨다. serialQueue의 노드를 직렬로 할당할 때에는 할당을 시작한 타임 슬롯 위치를 기억하고 있다가 다음 할당 시 해당 타임 슬롯부터 할당을 시작하게 된다. 왜냐하면 직렬할당은 타임 슬롯 위치를 1씩 증가하면서 직렬로 할당하게 되는데 이 때 직렬할당의 시작위치를 모른다면 다음 할당할 child가 직렬 할당이 끝난 다음 타임 슬롯에 위치하게 되어 다중 채널을 충분히 활용할 수 없기 때문이다. 병렬로 할당할 때에는 채널의 위치만 1씩 증가하게 되고 n개의 채널이 주어졌을 때 현재 위치가 n이라면 타임 슬롯을 1증가하고 채널번호는 1로 세팅한다.

3. 성능 평가

이 장에서는 기존의 다중 채널에서의 인덱스 스케줄링 방식과 SPAF방식과 비교하여 SPAF방식이 효과적인 스케줄링 방식임을 보일 것이다. 비교대상으로는 다중 채널 방송환경에서 인덱스 스케줄링 기법인 BFS, MWIS 두 가지와 제안하는 SPAF와 비교를 할 것이다.

모든 결과 값은 1000번의 질의처리에 대한 결과의 평균값이며 실제 공간 데이터인 real data는 [11]에서 구한 데이터를 사용하였고 지세 측량 정보가 있는 데이터이다. 실제 공간데이터의 전체 노드 개수는 총 3642 개이다. data item의 개수는 3000개, k의 개수는 4개로 고정하였고, 채널의 개수는 2개부터 30개 까지 변화를 주면서 실험을 진행하였다. 그림 4와 5는 채널 개수에 따른 튜닝타임과 레이턴시의 변화를 보여준다.

튜닝타임에서는 거의 변화가 없는 반면 레이턴시에서는 BFS와 SPAF는 채널의 수가 증가할수록 좋은 성능을 보여주었고 MWIS는 채널 개수가 15개 이상에서는 성능의 변화가 없어 좋지 않은 성능을 보여주고 있다. MWIS는 채널 개수가 15개 이후에서 레이턴시의 변화가 거의 없는데 이는 MWIS는 다중 채널에 인덱스를 할당할 때 겹치는 부분이 없는 스케줄링을 하기 때문에 겹치는 부분이 존재하면 한 타임 슬롯에 할당하는 것이 아닌 다음 타임 슬롯으로 넘어가 할당하게 된다. 이는 한 타임 슬롯에 할당 할 수 있는 인덱스의 개수가 제한적이라는 의미이고 주어진 다중 채널을 활용할 수가 없다는 의미이다. 반면에 BFS와 SPAF는 다중 채널을 충분히 활용하는 인덱스 스케줄링을 하기 때문에 채널의 개수가 늘어날수록 다중 방송채널의 장점인 짧은 방송 사이클로 인해 좋은 레이턴시를 얻을 수 있었다.

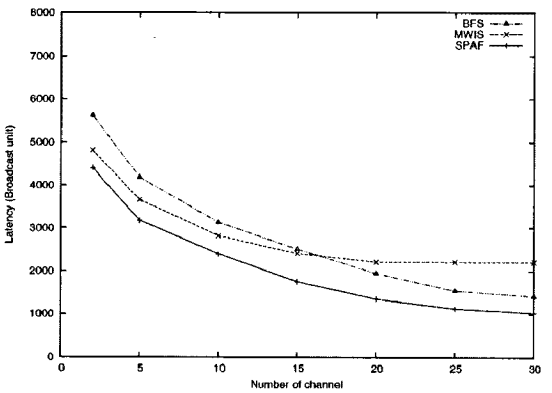


그림 4 채널의 수에 따른 레이턴시

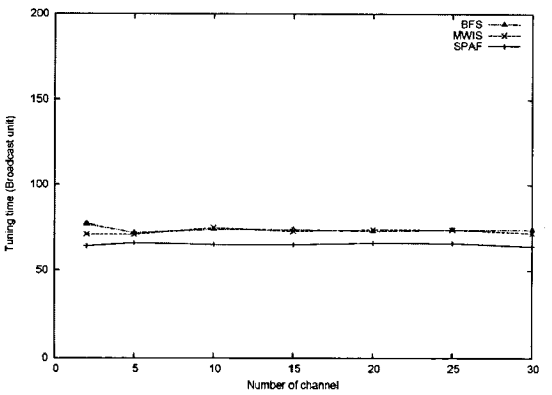


그림 5 채널의 수에 따른 레이턴시

4. 결 론

논문에서는 다중 방송 채널 환경에서 효과적인 kNN 질의처리를 위해서 접근빈도를 이용한 R-tree 기반 인덱스 스케줄링을 제안하였다. kNN질의 처리 특성상 노드의 여러 개의 child를 얻어야 되는 속성과 이런 속성은 상위 노드일수록 빈도가 높다는 특성을 이용하여 다중 채널에 할당될 때 이런 속성이 반영되는 인덱스 스케줄링을 하여 kNN질의처리 시 발생되는 conflict를 줄이는 스케줄링을 하였다. 이런 특징을 스케줄링에 반영하기 위하여 R-tree 각 노드의 접근빈도를 기반으로 전체 접근빈도의 평균보다 높은 빈도수를 가지는 노드의 child만을 직렬로 할당하여 kNN처리시 발생할 수 있는 conflict를 줄이고 평균보다 낮은 빈도수를 가지는 child들은 병렬로 할당하여 방송 스케줄 길이는 크게 늘어나지 않게 하는 SPAF 인덱스 스케줄링을 제안하였다. 제안한 SPAF의 성능을 비교 평가하기 위하여 기존의 R-tree 기반 인덱스 스케줄링 방식인 BFS와 MWIS를 비교하였다. 성능 평과 결과는 SPAF 방식이 다중 채널에서 BFS 방식으로 스케줄링 될 때 kNN 질의처리 시 발생될 수 있는 conflict를 효과적으로 줄여주는 것으로 나타났다. 향후 연구로는 kNN질의처리 외에도 윈도우 질의 처리와 같은 다양한 위치기반 질의처리에 적용될 수 있는 인덱스 스케줄링으로 확장하여 kNN뿐만 아니라 다양한 위치기반 처리를 수행할 수 있도록 확장성을 높이는 것이다.

참 고 문 헌

[1] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'84)*, pp.47-54, 1984.

[2] N. Beckmann and H.-P. Kriegel, "The R*-tree: An efficient and robust access method for points and rectangles," *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*, pp.322-331, 1990.

[3] Nick Roussopoulos, Stephen Kelley, Frederic Vincent, "Nearest Neighber Queries," *Proceedings of the ACM SIGMOD international conference on Management of Data (SIGMOD'96)*, pp.71-79, 1996.
[4] Continuous Nearest Neighbor Monitoring in Road Networks, Kyriakos Mouratidis, Man Lung Yiu, VLDB '06, September 12-15, 2006.
[5] Jun Zhang, Manli Zhu, Dimitris Papadias, "Location-based Spatial Queries," *Proceedings of the ACM SIGMOD international conference on Management of Data (SIGMOD'03)*, pp.443-454, June 2003.
[6] B. Zheng, W.C. Lee, and D.L. Lee, "Spatial Queries in Wireless Broadcast Systems," *Wireless Network, 10(6)*, pp.723-736, December, 2004.
[7] W.C. Lee, B. Zheng, "DSI: A Fully Distributed Spatial Index for Location-based Wireless Broadcast Services," *Proceedings of the 21st International Conference on Data Engineering*, pp. 349-358, 2005.
[8] B. Gedik, A. Singh, L. Liu, "Energy efficient exact kNN search in wireless broadcast environments," *Proceedings of the 12th Annual ACM International Workshop on Geographic Systems*, pp.137-146, 2004.
[9] Chuan-Ming Liu, Shu-Yu Fu, "Effective Protocols for kNN Search on Broadcast Multi-Dimensional Index Trees," *Information Systems 33*, pp.18-35, 2008.
[10] S. Fu, C. Liu, "Broadcast schedules and Query Processing for k Nearest Neighbors search on multi-dimensional index trees in a Multi-channel environment," *Proceedings of the IEEE International Conference on System, Man and Cybernetics*, pp.2646-2651, 2006.
[11] Spatial Datasets in 2D Space, http://www.rtree-portal.org/

정 의 준
2005년 3월~2007년 2월 서강대학교 컴퓨터공학과 학사. 2007년 3월~2009년 6월 서강대학교 컴퓨터공학과 석사. 2009년 7월~현재 동부CNI. 관심분야는 이동통신, 모파일 컴퓨팅 시스템, 모바일 데이터 베이스, 모바일 트랜잭션, 모바일 커머스

정 성 원
1988년 서강대학교 전자계산학 학사. 1990년 M.S. in Computer Science at Michigan State University. 1995년 Ph.D. in Computer Science at Michigan State University. 1997년~2000년 한국전산원 선임연구원. 2000년~현재 서강대학교 컴퓨터학과 교수. 관심분야는 Mobile Databases, LBS, Mobile Computing Systems, Spatial Databases, Telematics, GIS