이동체의 효율적인 색인을 위한 재삽입 정책

박상진° 김주성 김유성
인하대학교 대학원 정보통신공학과
22031523°@inhaian.net, 22051477@inhaian.net, yskim@inha.ac.kr

A Reinsertion Policy for Efficient Indexing of Frequent Moving Objects

Sangjin Park° , Joosung Kim , Yoosung Kim
Dept. of Information and Communication Engineering, Inha University

요 약

무선 통신, 유비쿼터스 컴퓨팅의 발달과 인터넷 환경이 널리 보급되면서 이동체의 색인을 위한 연구가 활발히 진행되었다. MV3R-tree 알고리즘은 과거 정보의 검색을 위한 Timestamp 질의와 Interval 질의 모두에 좋은 성능을 보여 주는 색인 기법이다. 그러나 기존의 인덱싱 기법인 3D R-tree에 비해 인덱스의 크기가 1.5배 크고, 삽입 시 재배치 비용이 2배정도 크다. 빈번하게 이동하는 이동체의 정보를 저장한다면 삽입 비용의 증가로 인하여 효율성은 현저하게 떨어진다. 본 논문에서는 노드 삽입 과정에서 아주 작은 변화나 반복적인 움직임 같은 불필요한 이동체의 거리 변화의 정보를 제한하는 방법을 제안한다. 정보를 제한할 임계치 값은 이동체들의 과거 정보를 통해 결정한다. 불필요한 정보를 제거하여 노드 공간의 낭비와 재배치를 횟수를 줄인다.

1. 서론

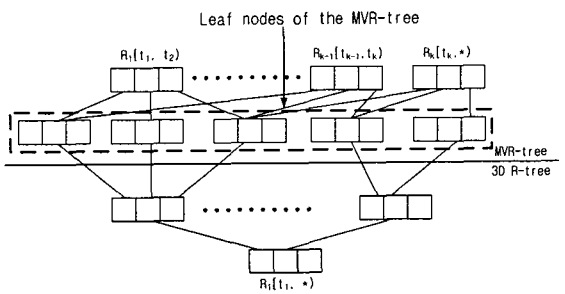
와이어리스, 모바일 컴퓨팅 기술의 발달과 인터넷 환경이 일반화되면서 위치 추적 시스템 기술과 서비스 분야에 대한 관심이 증가했다[1][2]. 예를 들어, 실내 환경에서 이동체의 검색이 가능할 경우에 문서의 이동정보나 휴대폰 같은 모바일 장비를 사용하는 사람들의 이동 정보를 유용하게 사용 할 수 있다. 이런 필요성에 발맞추어 이동체의 정보를 효율적으로 다루기 위한 인덱싱 기법이 연구되어 왔다.

인덱싱 기법에 대한 연구는 세부적으로 과거 정보에 대한 검색, 이동체의 궤적을 검색하여 미래의 이동체의 위치에 대한 예측 등의 질의로 분류된다[3][4][5]. 과거 정보의 검색을 위한 Timestamp 질의(시간의 정보)와 Interval 질의(공간의 정보) 모두에 좋은 성능을 보여주는 MV3R-tree를 색인 기법으로 널리 쓴다[6].

MV3R-tree는 모든 질의에서 좋은 성능을 보여주지만, MVR-tree와 3D R-tree의 결합한 형태이기 때문에 트리의 재배치 시 두개의 구조를 변경해야 한다. MV3R-tree의 재배치 분할 비용이 증가하고 중복되는 데이터로 노드 공간이 낭비된다. 본 논문에서는 문제점을 해결하기 위해 한정된 fanout(노드의 레코드 수)을 가진 MV3R-tree의 데이터의 재배치 시 중복을 줄이고 불필요한 움직임을 제한할 수 있는 재배치 알고리즘을 제안한다.

2. 관련 연구

2.1 MV3R-tree

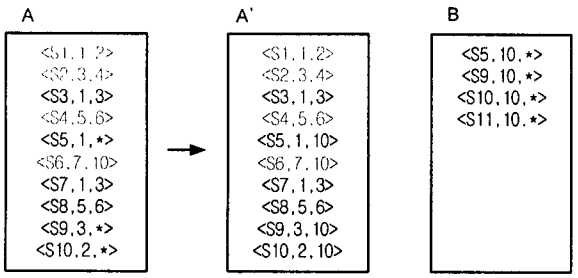


[그림 1] MV3R-tree의 구조

B+ tree를 기반으로 하여 2개의 색인 구조(3D R-tree와 MVR-tree)를 결합시킨 MV3R-tree 방법에 대하여 설명한다. [그림 1]은 MV3R의 구조를 보여준다.

MV3R-tree 노드의 하나의 레코드는 <S, T_{start}, T_{end},

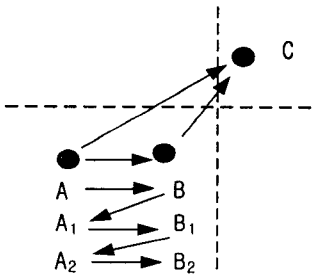
pointer>으로 구성된다. 여기에서 공간 MBR (Minimum Bounding Rectangle, 최소 경계 사각형)을 나타내고, T_{start}, T_{end}는 각 객체의 지속시간의 시작점과 끝점을 나타낸다. *는 객체의 지속 시간의 끝나지 않고 진행하고 있을 때 사용한다. 즉, 객체가 현재까지의 위치나 모양 정보를 변경하지 않았음을 의미한다. pointer는 루트 노드에서 실제 레코드와 일치하는 키 값을 저장하고 있는 노드를 가리킨다.[그림 2]는 루트 노드 <R1, 1, 10, A>가 가리키는 A 노드의 삽입 과정이다.



[그림 2] MV3R tree의 삽입 과정

MVR-tree의 말단 노드를 이용하여 3D R-tree를 구성한다. 이렇게 만들어진 MVR-tree는 timestamp 질의를 처리하고, 3D R-tree는 interval 질의를 처리한다.

2.2. MV3R-tree의 삽입 시 문제점



[그림 3] 이동체의 위치 변화

기존의 MV3R-tree 알고리즘은 이동체의 움직임이 있을 때, [그림 3]에서와 같이 A -> B -> A1 -> B1 -> A2 -> B2 -> C 순으로 변화된 이동 정보가 저장 된다. 이 경우 불필요한 이동정보 A1, B1, A2, B2 등의 불필요한 작은 변화의 이동 정보까지 저장된다. 예를 들면, 오피스 내에서 문서가 자료수정을 위해 옆 동료와 연속으로 문서를 교환하는 경우이다. 저장된 불필요한 정보들로 인하여 [그림 2]에서와 같이 노드의 fanout 10개가 꽉 찬 상태에서 또 하나의 정보 <S11,10,*>가 들어오면 A'와 B로 재배치 나누어지면서 끝나지 않은 *를 갖는 S5, S9, S10의 레코드 값이 복사된다.

MV3R-tree는 [그림 3]과 같이 이동체의 잦은 변화가 있는 경우 연속적으로 데이터의 정보를 저장하므로 추적하고자 하는 이동체의 양이 늘어나면 노드의 레코드 낭비가 심해진다. 또한 불필요한 정보들로 인해 노드가 재배치되고 트리의 레벨이 증가한다.

3. 불필요한 정보를 제한하기 위한 재배치 정책

본 논문에서는 임계치 값을 이용한 합병 방법을 통해 빈번하게 변하는 불필요한 정보를 줄이고 시공간 이동체의 정보를 효율적으로 저장하기 위한 재배치 알고리즘을 제안한다. 새로운 재배치 정책을 통한 이상적인 이동체의 이동 경로는 A -> B -> C 순으로 이동 정보를 저장하는 것이다. 잦은 변화를 줄이기 위해 임계치 T 값을 이용한다. 임계치 값이란 삽입 시 불필요한 정보를 필터링하기 위한 값이다.

3.1 임계치 값을 구하기

추적하고자 하는 이동체의 위치 정보 샘플들을 모아 이동체의 거리의 변화량에 따른 값들을 정렬하고 그중에서 거리의 변화량이 작은 N개의 샘플들을 구한다. 샘플 중에서 작은 변화량을 갖는 N개의 평균값은 수식 (1)에 해당한다.

$$U = \frac{1}{N}\sum_{i=1}^{N} S_i \quad \cdots\cdots (1)$$

노드의 레코드의 수는 저장 공간의 민감도(Sensitivity)와 반비례하기 때문에, 레코드의 노드 공간이 중요하지 않다. 그러나 노드의 fanout 수는 제한이 있으므로 기존 논문에서는 fanout의 수를 36개로 실험하였다[6]. σ는 자료가 저장될 수 있는 [노드 공간(fanout) / 10]으로 가정하고 정확한 노드 공간에 따른 비율은 실험을 통하여 결정한다. 임계치 T 값은 σ와 U의 값을 곱하여 결정한다. 이 값은 수식 (2)에 해당한다.

$$T = \sigma \cdot U \quad \cdots\cdots (2)$$

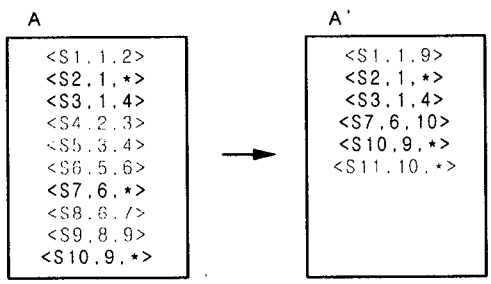
평균 U의 값이 1m이고 노드의 fanout의 수가 30개라면 임계치 T값은 3m이다.

3.2 제안하는 재배치 정책

```
Algorithm Delete_Limit_Critical_Value
(this_node, entry_to_reinsert)
1. list = {all entries below the critical value in list}
2. entry_to_reinsert = the first entry in list
3. remove the first entry
4. for(entries is empty in list )
5.       entry_to_reinsert = entry.value.Tend
6.       remove entries to input entry_to_reinsert
7.return entry_to _reinsert;
```

[표 1] Delete_Limit_Critical_Value Algorithm

[그림 2]와 같이 노드가 꽉 찬 경우 노드의 스플릿이 일어나기 전에 [표 1]의 Delete_Limit_Critical_Value 알고리즘을 이용하여 불필요한 레코드 값을 제거한다. 노드의 저장 공간이 BlockOverFlow(빈 저장 공간이 없는 상태)가 일어나면 스플릿이 일어나기 전에 노드 안에 불필요한 정보가 있는지 검색한다. 그 후에 임계치 이하의 불필요한 정보들을 합병한다. 처음 레코드의 값을 저장하고 다음 레코드 정보가 들어왔을 때 두 번째 노드의 T_{end} 시간으로 레코드의 정보를 수정한다. 이렇게 갱신된 두 번째 레코드는 삭제한다. 알고리즘의 4-6 번 과정을 반복하여 리스트안의 임계치 이하의 레코드 값을 합병한다.



[그림 5] 새로운 재배치 정책

[그림 2]와 [그림 5]를 비교하면 불필요한 정보였던 S4, S5, S6, S8, S9의 이동체 변화 정보가 S1과 하나로 합병하여 낭비된 노드 공간이 줄어든 것을 보여준다. 시간 정보를 갱신하여 이동체의 움직임이 있어다는 정보를 알 수 있다. 또한 이동거리 역시 추측할 수 있다. 예를 들어, S1 레코드의 T_{end} 시간을 다음 정보 S4의 레코드의 데이터와 병합하여 S1 레코드의 변화 정보는 <S1,1,3>으로 변한다. Delete_Limit_Critical_Value 알고리즘을 통하여 임계점 이하의 값이 계속 합병되어 <S1, 1, 9>로 S1의 레코드 정보가 변한다.

임계치 T 값의 크기에 따라 중요한 데이터의 변화 정보를 잃어버릴 수 있다. 그러나 임계치 T 값을 구하기 위한 데이터의 정보를 늘리거나 혹은 데이터가 적을 경우에는 U의 값을 취하는 변화량의 작은 샘플 N의 수를 줄여 T 값에 변화를 주어 다양한 환경에 맞게 적응을 시켜줄 수 있다.

4. 결론 및 향후 연구

기존의 MV3R-tree 색인 알고리즘은 이동체의 변화 정보가 있을 때 마다 이동 정보를 저장하였다. 빈번히 변하는 이동체들의 정보를 저장하면 불필요한 이동체의 정보까지 저장하여 색인 알고리즘의 성능을 저하시킨다. 본 논문에서는 불필요한 노드 공간의 낭비를 줄이고 스플릿을 줄이는 효율적인 재배치 정책을 제안했다. 향후 실험을 통하여 더욱 정확한 임계치 값의 기준을 정하고 새로운 재배치 정책을 도입한 MV3R-tree 알고리즘을 구현할 예정이다.

참고문헌

[1] Rashimi Bajaj, Samantha Lakinda Ranaweera and Dharma P. Agrawal, "GPS: Location Tracking Technology", Computer, pp.92-94, April, 2002.

[2] Jeffrey Hightower and Gaetano Borriello, "A Survey and Taxonomy of Location Systems for Ubiquitous Computing", Technical Report UW-CSE 01-08-02, University of Washington, Department of Computer Science and Engineering.

[3] Nascimento, M. Silva, J. Theodoridis, Towards Historical R-trees. ACM SAC, 1998.

[4] Pfoser, D. Jensen, C. Theodoridis, Y Novel Approaches to the indexing of Moving Object Trajectories. VLDB, 2000.

[5] Saltenis, S. Jensen, C. Leutenegger, S. Lopes, M, Indexing the positions of Continuously Moving Objects. ACM SIGMOD, 2000.

[6] Y. Tao and D. Papadias. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. VLDB, 431~440, 2001.

[7] O. Wolfson, A. P. Sistla, S. Chamberlain and Y. Yesha, "Updating and querying databases that track mobile units", Distributed and Parallel Databases, pp. 257 ~ 387, 1999.