다차원 색인구조 M-트리에서 노드 색인공간의 중첩을 최소화하기 위한 효율적인 분할 알고리즘

임상혁° 김유성
인하대학교 정보통신대학원
c2021026@inhavision.inha.ac.kr yskim@inha.ac.kr

An Efficient Split Algorithm to Minimize the Overlap between Node Index Spaces in Multi-dimensional Indexing Scheme M-tree

Sanghyuk Im° Yoosung Kim
School of Information and Communication Engineering, Inha University

요 약

다차원 색인 기법 M-트리는 노드의 중심점과 객체간의 상대적 거리로 색인을 구성하고, 검색 공간에 포함되는 객체를 액세스하는 기법으로서 노드들은 페이지 단위로 구성되며 하위 엔트리들을 포함할 수 있는 반경, 즉 유사도 거리에 의해 노드의 영역이 표현되어진다. 그러나 이와 같은 노드의 영역 표현에 있어서 노드 색인공간의 중첩으로 인해 질의 시 검색 패스가 증가하고 이로 인해 거리계산과 디스크 입출력의 증가를 야기하는 문제를 갖는다. 본 논문에서는 M-트리에서 문제가 되고 있는 노드 색인 공간의 중첩을 최소화할 수 있는 노드 분할 정책을 제안한다. 기존의 M-트리와는 다르게 노드의 가상 중심점을 계산하여 그것을 라우팅 객체로 만들어 노드를 커버하는 영역을 최소화하고 노드 안의 엔트리를 재분배하여 밀도 높은 노드를 구성 하도록 한다. 제안된 분할 알고리즘의 효율성을 증명하기 위한 실험 결과 색인공간의 중첩이 줄고 이로 인해 거리계산과 디스크 입출력의 횟수가 줄어듦을 보였다.

1. 서 론

최근 데이타베이스 응용에서 멀티미디어 데이타에 대한 내용기반 검색을 위한 유사성에 기반한 검색기법에 대한 관심이 부각되고 있다. 멀티미디어 데이타는 다차원, 대용량의 특징상 기존의 색인 기법과는 다른 다차원 색인 및 검색 기법이 필요하며, SAMs(Spatial Access Methods)와 MAMs(Metric Access Methods)등의 기법들이 연구 되고 있다.[1][2]

이러한 다차원 색인구조에서는 차원이 증가할수록 저장과 검색 효율이 떨어지는 차원의 저주(the curse of dimensionality)문제와 색인공간 중첩의 증가로 인해 거리계산이나 디스크 입출력이 증가하는 문제 등이 해결해야 할 문제점으로 지적되고 있다. 특히 SAM과는 다르게 축 개념을 도입하지 않고 두 객체간의 거리에 기반한 색인 구조인 MAM에서는 노드의 색인공간 중첩으로 인해 질의 시 검색 대상이 될 노드가 많아져 거리계산과 디스크 입출력이 크게 증가 하는 문제를 안고 있다.

본 논문에서는 MAM의 대표적 색인 구조인 M-트리[3]에서 이러한 노드 색인 공간의 중첩을 최소화하는 방법을 제시한다. 기존의 M-트리에서는 라우팅 객체를 하부 노드의 엔트리 중에서 선택함으로써, 노드의 표현에 있어서 불필요하게 색인 공간이 커지고 이로 인해 다른 색인공간과 중첩이 될 확률이 높아지는 문제가 있었다. 이에 대한 해결 방안으로 노드의 중심점을 계산하여 그 중심점을 노드의 가상 라우팅 객체로 만듦으로서 노드 영역을 최소화하고 노드간의 중첩을 감소시킬 수 있는 분할 알고리즘과 노드의 반경을 더욱더 줄이면서 밀도 높은 노드들로 트리를 재구성할 수 있는 엔트리 재분배 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 M-트리에 대해 간략하게 요약하고 3장에서는 기존 분할 방법의 문제점과 색인공간의 중첩을 최소화하기 위해 본 논문에서 제안하는 기법에 대해 논의 한다. 4장에서는 실험을 통해 제안된 기법의 성능을 검증하며 5장에서는 결론을 내리고 향후 연구방향을 제시한다.

2. 관련 연구

M-트리는 객체간의 상대적 거리를 이용하여 주어진 검색 공간을 분할하는 MAM의 기본적인 원리를 따르면서 동적 특성(동적인 삽입과 삭제), 페이지 특성(고정크기 노드), 균형 특성(모든 단말 노드는 같은 레벨에 위치)을 갖는다. 그리고 실제 객체를 저장하는 단말노드(leaf node)와 라우팅 객체를 저장하는 내부노드(internal node)로 이루어진다. 객체들은 참조되는 라우팅 객체들로부터의 거리에 기반 하여 재귀적으로 구성되며 라우팅 객체들은 노드의 분할 시 선택된 객체들로서 이루어진다. M-트리의 노드 종류와 엔트리 구조, 그리고 노드들의 도식적 표현을 각각 표1과 그림1에서 나타내었다. 그림 1에서 볼 수 있듯이 내부노드와 단말노드의 첫 번째 엔트리와 그 부모노드와의 거리($d(O_r, P(O_r))$ 혹은 $d(O_j, P(O_j))$)는 0으로서, 이는 첫 번째 엔트리가 노드의 중심 즉, 라우팅 객체임을 의미한다. 실제로 M-트리에서는 노드안의 엔트리들과 노드의 중심까지의 거리에 따라 노드안의 엔트리들이 정렬되어 저장된다.

표 1 M-트리의 노드종류와 엔트리구조

내부노드 엔트리	O_r	라우팅 객체
	$ptr(T(O_r))$	부트리의 루트를 가르키는 포인터
	$r(O_r)$	O_r의 반지름
	$d(O_r, P(O_r))$	O_r과 O_r의 부모간의 거리
단말노드 엔트리	O_j	데이타 객체
	$oid(O_j)$	객체 식별자
	$d(O_j, P(O_j))$	O_j와 O_j의 부모간의 거리

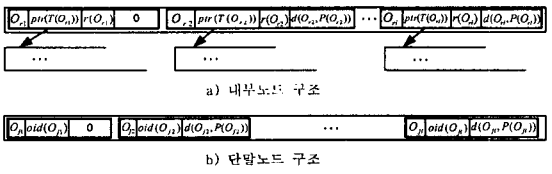

그림 1 M-트리 노드들의 도식적 표현

3. 가상 중심점과 엔트리 재분배를 이용한 분할

기존의 M-트리 구조와 분할 알고리즘은 노드의 엔트리 중 하나를 뽑아 라우팅 객체로 선정하므로 불필요하게 노드의 반경이 커지고 이로 인해 노드 색인 공간의 중첩이 증가하는 문제점이 있다. 이러한 색인 공간의 중첩은 질의 시, 검색 패스의 증가를 가져와 많은 디스크 입출력과 거리계산이 필요하게 된다. 그림 2는 이와 같은 기존 분할 정책의 문제점을 나타내고 있다. 여기서 진급(promotion)객체란 분할 후 노드의 중심, 즉 라우팅 객체가 될 객체를 의미한다.

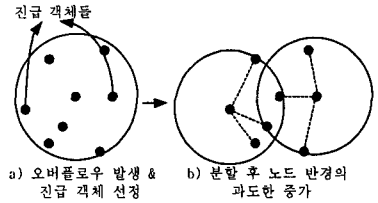

그림 2 기존 분할 정책의 문제점

본 논문에서는 이러한 문제점을 해결하기 위해 노드 안의 엔트리를 라우팅 객체로 선출한 기존의 분할 방법과는 달리 노드 안의 모든 엔트리를 이용하여 가상 중심점(Virtual Center Point:*V_CP*)을 계산한 후 그 중심점을 노드의 라우팅 객체로 선정한다. *V_CP*의 계산을 위해 K-means[4], K-medoids[4]등의 메트릭 공간상의 클러스터링 알고리즘을 이용할 수 있다. 하지만 이러한 알고리즘들은 클러스터 안의 모든 객체에 대한 무게 중심점을 찾으므로 객체들의 밀도가 높은 쪽으로 중심점이 이동하여 클러스터의 반경을 늘리게 되고, 중심점을 찾기 위한 반복 작업으로 인해 복잡도 역시 크다. 본 논문에서는 최대 거리 엔트리 쌍을 이용하여 영역을 커버할 수 있는 최소의 반지름을 갖는 클러스터를 생성한다. 그림 3은 중심점 선출을 위해 본 논문에서 제안하는 클러스터링 방법을 나타낸다.

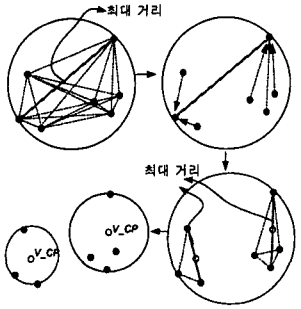

그림3 최대 거리 쌍을 이용한 클러스터링

그림 3에서 볼 수 있듯이 객체간의 거리를 각각 계산하고 최대 거리를 갖는 두 엔트리를 선정한 후 그 엔트리들에 각각 가장 가까운 객체들로 클러스터를 형성하게 된다. 클러스터의 중심점은 평균점이 아닌 클러스터의 엔트리 쌍 중 최대 거리의 엔트리들의 중심점으로 한다. 이와 같은 클러스터링 과정을 이용한 M-트리의 새로운 분할 방법은 그림 4와 같다.

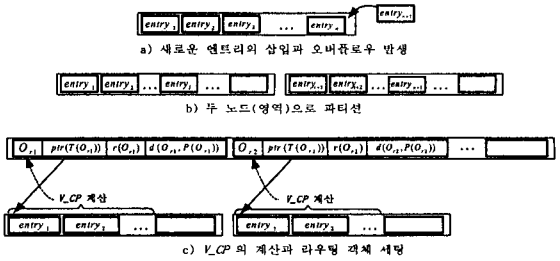

그림 4 제안된 분할 방법

그림 4(a)와 같이 노드에 오버플로우가 발생하면 클러스터링 과정을 통해 두개의 영역(노드)으로 분할을 하게 되고(그림 4(b)), 각각 분할된 영역의 엔트리들을 계산하여 가상의 중심점(*V_CP*)을 생성하며 이를 상위 레벨의 부모 객체로 대체하게 된다.(그림 4(c)) *V_CP*의 개념을 통해 내부 노드 엔트리들의 객체들은 모두 가상 점이 되며 단말 노드를 위한 라우팅 객체 역시 계산에 의해 얻어진 *V_CP*값으로 바뀌게 된다. 이러한 *V_CP*의 도입으로 인해 한 노드의 표현을 위한 최적의 라우팅 객체를 선출 할 수 있으며 이로 인해 기존 분할 방법 보다 더 작은 노드의 반경을 얻을 수 있다.

분할의 후 처리과정으로 보다 밀도 높은 노드를 만들기 위해 노드의 엔트리들을 형제 노드로 재분배 한다. 그림 5(a)와 같이 분할된 노드의 중심에서 가정 먼 거리를 갖는 엔트리의 중심점까지의 거리가 다른 형제 노드의 중심점 보다 멀 때 그 노드에서 엔트리를 삭제한 후 형제 노드에 재 삽입 한다.(그림 5(b)) 여기서 고려해야 할 사항은 재분배로 인한 형제 노드의 연속된 오버플로우를 막기 위해 형제 노드의 현재 엔트리 개수를 고려해야 한다는 것이다.

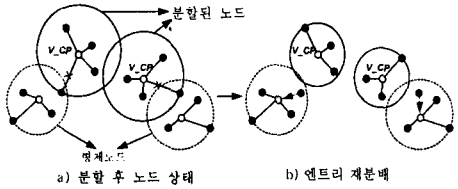

그림 5 엔트리 재분배 과정

재 삽입 될 수 있는 형제 노드 중 현재의 엔트리 개수가 노드가 최대로 수용할 수 있는 엔트리 개수 보다 1이상 작은 노드를 선택하게 된다. 이 기법을 통해 어떠한 노드들의 반경의 증가 없이 분할된 노드의 반경을 줄이고 밀도 높은 노드들로 트리를 재구성할 수 있다.

4. 실험 및 평가

실험을 위한 트리의 구현은 다차원 색인을 위한 GiST 0.9 패키지[5]를 기반으로 한 M-트리[3]를 이용하였다. 실험은 450MHz pentium II PC(주기억 장치 256메가바이트)의 하드웨어, Window2000 Professional 운영체제 환경에서 Visual C++6.0 컴파일러를 사용하여 수행되었으며 페이지 크기는 4K 바이트로 고정 하였다. 실험 데이타는 난수 발생기를 이용하여 10,000개에서 100,000개까지 균등분포를 갖는 [0,1] 범위의 10차원 실수 데이타 집합을 이용하였다. 비교 대상으로서는 기존의 M-트리에서 질의 시 디스크 입출력과 거리계산에서 최고의 효율성을 지닌 m_RAD 진급 정책을 이용한 노드 분할 방법과, 가장 간단한 진급 정책인 RANDOM 정책을 이용한 분할방법, 그리고 본 논문에서 제안한 가상 중심점과 엔트리 재분배를 이용한 New_Split 분할 방법의 성능 비교이다.

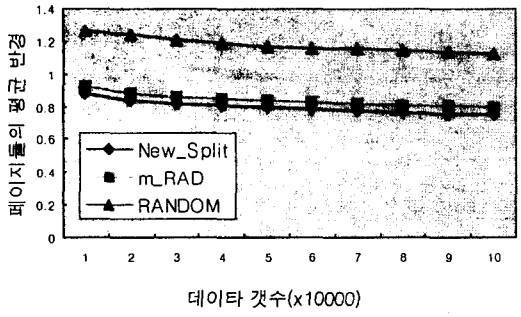

그림 6 생성된 페이지들의 평균 반경

그림 6은 트리가 생성된 후 노드들의 평균 반경을 나타내고 있다. 그래프에서 볼 수 있듯이 본 논문에서 제안하는 New_Split 분할 방법이 평균반경에서 다른 방법들에 비해 작은 것을 알 수 있는데, 이는 *V_CP*와 엔트리 재분배로 인한 노드들의 중첩 감소를 의미한다.

그림 7은 10개의 근접 객체를 찾는 근접 질의를 10회 실시한 응답 시간의 평균치를 나타내며, New_Split 분할 방법이 색인공간 중첩의 감소로 인해 디스크 입출력과 거리계산이 줄어들어 응답 시간이 가장 적게 걸림을 알 수 있다.

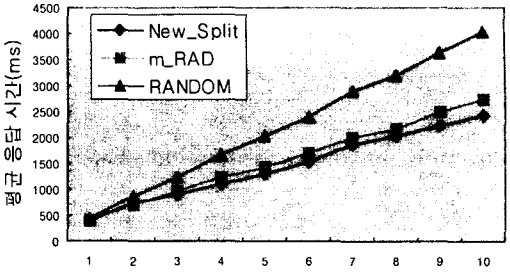

그림 7 근접 질의 시 평균 응답 시간

5. 결론

본 논문에서는 다차원 데이타를 색인하고 질의하기 위한 M-트리를 소개하고 성능 향상을 위해 노드의 색인 공간 중첩을 감소시키는 새로운 분할 방법을 제안하였다. 분할 시, 엔트리간의 최대거리를 이용하여 오버플로우 노드를 분할하고 노드들의 가상 중심점을 계산하여 라우팅 객체로 만듦으로서 노드를 포함하는 반경을 최소화하고 이를 통해 중첩 공간을 감소시킨다. 또한, 분할의 후처리 과정으로서 노드의 엔트리들 중, 중심점에서 가장 먼 엔트리를 형제 노드로 삽입 시키는 엔트리 재분배를 통해 노드의 반경을 더욱 줄이면서 밀도 높은 노드들로 트리를 재구성 할 수 있었다. 기존의 분할 방법들과의 비교 실험을 통해, 트리의 구성과 질의 시 응답 시간 측면에서 제안하는 알고리즘이 성능 향상이 이루어짐을 보였다.

향후 연구방향은 오버플로우 노드의 데이타 객체 분할 시, 분할된 노드의 불균등한 엔트리 개수로 인해 발생할 수 있는 잦은 분할 문제와 노드의 최소 엔트리 개수를 조정하여 트리의 팬아웃을 증가 시키는 알고리즘을 연구하는 것이다.

[참고 문헌]

[1] V. Gaedes and O. Gunther, "Multidimensional access methods,". Technical Report TR-96-043, Computer Science Institute, Berkely, CA, pp36-59, October 1996.

[2] C. Traina Jr., A. Traina, C. Faloutsoss and B. Seeger, "Fast Indexing and Visualization of Metric Data Sets using Slim-Trees," IEEE Transaction on Knowledge and Data Engineering, Vol. 14, No. 2, pp244-259, 2002.

[3] P. Ciaccia, M. Patella and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric spaces," Proc. of VLDB, pp426-435, 1997.

[4] J. Han, M. Kamber, "Data Mining: Concepts and Techniques," Morgan Kaufmann Publishers, pp349-354, 2001.

[5] Joseph M. Hellerstein, Jeffrey F. Naughton and Avi Pfeffer, "Generalized Search Trees for Database Systems," Proc. of VLDB, pp562-573, September 1995.