병렬처리 공간자료구조연구

방갑산*
한성대학교 정보시스템공학과

e-mail: ksbang@hansung.ac.kr

A Study on Parallel Spatial Index Structure

Kapsan Bang

Dept. of Information System Engineering, Hansung University

요 약

공간데이터를 관리하는 공간 index structure는 대부분 순차처리를 위한 구조를 가지고 있다. 많은 응용분야에서 방대한 양의 공간 데이터는 보조기억장치(예: disk)에 저장이 되어 사용이 되고 공간 index structure의 operation은 I/O에 대한 의존도가 크므로, I/O operation의 병렬처리는 공간 index structure의 질의반응시간을 현저하게 줄일 수 있다. 본 논문에서는 PPR-tree라는 병렬형 공간 index structure를 제안한다.

1. 서론

대부분의 공간 index structure는 질의연산을 순차적인 방식으로 처리한다. 이러한 방식의 공간 index structure는 참고문헌 [1,2,6]에서 발견할 수 있다.

MXR-tree[5]는 최초의 병렬처리 공간 index structure이다. MXR-tree는 R-tree의 node를 여러 개의 disk에 배치하기 위해 proximity index방식의 heuristic을 사용한다. Proximity index의 목적은 새로이 만들어진 node가 기존의 node들과 함께 검색될 확률이 가장 적은 disk에 새 node를 배치하는 것이다. 이상적인 node의 분배를 위해서는 같은 level에 있는 모든 sibling node들을 proximity index에서 고려해야 하나 이를 위해서는 너무 많은 disk access가 필요하므로 같은 parent node를 가진 sibling node들만을 proximity index계산에 고려한다. 따라서 proximity index heuristic은 경우에 따라서 node distribution에 불균형을 가질 수 있다.

MXR-tree는 R-tree의 구조적인 특성을 그대로 갖고있기 때문에 R-tree의 extra search path와 같은 문제를 갖고 있다. 병렬처리 공간 index structure의 질의연산에서 D(node배치에 사용된 disk의 수)개의 process가 만들어지며 각 process는 동시에 할당된 disk를 search한다. MXR-tree는 하나의 tree를 가진 구조이며 모든 node들은 D개의 disk에 분배된다. 따라서 MXR-tree는 질의연산동안에 D개의 process각각이 search또는 deletion 영역과 overlap하는 intermediate entry를 찾을 때마다 process간에 communication이 필요하다. Process간에 communication에 필요한 시간은 사용되는 disk의 숫자에 비례하여 증가한다. 본 논문에서 제시되는 PPR-tree는 native space indexing과 disjoint space decomposition(같은 level에 있는 index rectangle들이 서로 overlap하는 것을 허용치 않는 방식)을 사용한다.

*본 연구는 2009년도 한성대학교 연구년 지원과제임

2. PPR-tree

효율적인 병렬처리 공간 index structure는 질의 processing에 있어 total disk access의 수가 적어야함은 물론 access되는 node가 disk상에 균일하게 분포되어 있어야 한다. 질의 processing동안에 total disk access의 수를 줄이기 위해서 공간 index structure는 R-tree[4], R^+-tree[7], R*-tree[3]의 단점을 향상시켜야한다. PPR-tree는 data object를 여러 개의 data space에 분배함으로써 intermediate rectangle들 사이의 overlap을 제거하고 leaf node에 존재하는 중복된 entry들을 제거한다. PPR-tree는 native space indexing과 disjoint space decomposition방식[4, 7]을 사용한다.

2.1 PPR-tree의 구조

PPR-tree는 multiple-layer와 multiple-tree의 구조를 가지고 있다. 각 layer는 multiple의 tree들로 구성되어 있고, tree의 개수는 사용되는 disk의 수와 같다. 각 tree는 height-balanced tree이며 같은 layer에 있는 tree들은 같은 height를 갖는다. 각 data space는 하나의 disk상에 위치하는 독립된 하나의 tree와 연관되어있다. 하나의 disk는 하나이상의 tree를 가질 수 있다.

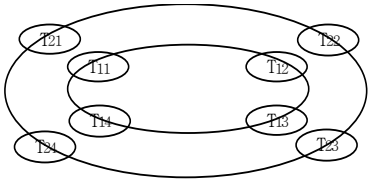

그림1. 2-layer 4-tree로 구성된 PPR-tree의 layout

PPR-tree에서는 각 tree들이 독립적인 구조를 가지고 있기 때문에 질의연산동안에 inter-process communication이 불필요하다. 따라서 모든 disk의 완전한 병렬처리가

가능하다. 그림1은 2개의 layer와 4개의 tree를 가진 PPR-tree의 layout을 보여준다. 각 tree들은 tree ID인 T_{ij}(i는 layer의 번호이고 j는 tree의 번호를 나타낸다. Tree번호는 disk의 번호이기도 하다)에 의해 식별될 수 있다. Disk j는 tree ID T_{ij}(i= 1,2,3,...,L, L은 layer의 수)를 가진 tree들을 포함한다. 예를 들어 그림1에서, disk 1은 tree T_{11}과 T_{21}을 포함한다. 따라서 PPR-tree의 질의 반응시간은 disk access의 최대 수에 비례한다. 만일 P_{ij}가 tree T_{ij}안에 있는 node들을 처리하는데 걸리는 시간이라면 PPR-tree의 질의 반응시간(R)은:

$$R = \max_{j=1...D}(\sum_{i=1}^{L} P_{ij})$$

(D는 disk의 수를, L은 layer의 수를 나타낸다)

2-2. PPR-tree 알고리즘(Relative Crowd Index)

Relative crowd(RC) index는 새로운 엔트리를 삽입하기 위한 트리의 삽입경로에 있는 노드들의 relative crowdness를 사용한다. RC index distribution heuristic은 삽입경로 상에 있는 색인 사각형들의 크기 대한 엔트리의 상대밀도가 낮은 트리를 선택한다. 질의 연산에서 좋은 성능을 갖기 위해서는 PPR-트리는 같은 계층에 있는 트리의 삽입경로에 있는 노드들의 숫자가 거의 유사해야만 한다. 즉, 각 데이터 공간에서 주어진 검색 영역과 중첩하는 색인 사각형의 숫자가 거의 유사해야한다. 왜냐하면, PPR-트리에서는 중첩되지 않은 공간 분할방식이 사용이 되고 하나의 노드는 데이터 공간에 있는 색인 사각형에 의해 표현되기 때문이다. 이러한 조건을 만족시키기 위해서 RC index distribution heuristic은 삽입경로를 따라서 엑세스된 노드(단 root 노드는 제외한다)의 엔트리당 평균 영역크기를 계산하고 그 값들을 합한다. 값이 크면 클수록 그 영역의 엔트리 밀도는 낮은 것이다. RC index distribution heuristic은 가장 낮은 엔트리 밀도를 가진 트리를 선택한다. RC index 값은 다음과 같이 계산될 수 있다:

$$\sum_{j=2}^{h}\left(\frac{M_j \times W^{(j-2)}}{N_j \rightarrow EntryNum}\right)$$

h는 트리의 오더, W는 가중치 상수, N_j는 삽입경로의 오더가 j인 노드, M_j는 N_j에 대한 minimum bounding 사각형의 크기를 나타낸다.

높은 오더를 가진 노드를 나타내는 사각형들은 낮은 오더를 가진 노드를 나타내는 사각형에 의해 완전히 포함이 되므로, 가중치 상수인 W(>1)는 삽입경로에 있는 노드를 나타내는 색인 사각형들의 size를 normalize하기 위해 사용이 된다. 만일 말단 노드가 NEW를 받아들일 수 있는 상태라면 트리의 상태는 말단 노드내의 엔트리의 수에 의해 A(accept)또는 S(accept and split)로 설정된다. 상태가 A 또는 S인 트리들에 대해서 알고리즘은 4가지의 criteria(트리 오더, 트리 상태, index 값, 트리에 속한 엔트리의 수)를 사용을 해서 하나의 트리를 선택을 한다. 이들 4개의 criteria의 적용 우선 순위는:

1. 트리 오더- 최소 오더를 가진 트리를 선택
2. 트리 상태- 상태 A를 가진 트리를 상태 S를 가진 트리에 우선해서 선택
3. index 값- 최대 RC index값을 가진 트리를 선택
4. 트리에 속한 엔트리의 수- 최소 숫자의 엔트리를 가진 트리를 선택

RC index distribution heuristic에서 트리 오더가 index값의 적용보다 우선한다. Index값이 트리 오더보다 우선 적용이 되었을 때, PPR-트리의 질의 성능은 향상이 되나 공간 활용도가 낮아지는 단점이 생긴다. 만일 현재의 계층에 어떤 트리도 새로운 엔트리인 NEW를 받아들일 수 없다면, 현재의 계층을 하위의 계층으로 설정을 하고 선택 프로세스를 반복한다. 만일 하위의 계층이 존재하지 않는다면 새로운 계층이 만들어진다.

3. 결론

PPR-tree는 여러 개의 disk를 사용하여 질의연산에 따르는 disk I/O를 병렬 처리함으로써 질의 processing을 향상시켰다. 공간 data object를 disk상에 균일하게 분배하기 위해서, PPR-tree는 object distribution heuristic을 제시하였다. PPR-tree와 MXR-tree에 대한 성능의 비교에서 PPR-tree는 우수한 질의 response time과 공간활용도를 갖고 있는 것으로 분석된다.

참고문헌

[1] Bang, K. S. and Lu, Huizhu, SMR-tree: an efficient Index Structure for Spatial Databases, Proc. of the 1995 ACM Symposium on applied Computing, Nashville, February, pp. 46-50, 1995.

[2] Bang, K. S. and Lu, Huizhu, An Efficient Index Structure for Spatial Databases, Journal of Database Management, Vol. 7, No. 3, Summer, pp. 3-15, 1996.

[3] Beckmann, N and Kriegel, H. P., The R*-tree: An Efficient and Robust Access Method for Points and Rectangles, ACM SIGMOD, pp. 322-331, 1990.

[4] Güttman, A., R-trees: A Dynamic Index Structure for Spatial Searching, Proc. of the ACM SIGMOD, pp. 47-57, 1984.

[5] Kamel, I. and Faloutsos, C., Parallel R-tree, ACM SIGMOD, pp. 195-204, 1992.

[6] Lomet, D.B., A Review of Recent Work on Multiple attribute Access Methods, ACM SIGMOD Record, Vol. 21, No. 3, pp. 56-63, 1992.

[7] Sellis, T., Roussopoulos, T. and Faloutsos, C., R^+-tree: A Dynamic Index for Multi-dimensional objects, Proc. of the 13th VLBD Conference, pp. 507-518, 1987.