고정 그리드를 이용한 병렬 공간 조인을 위한 비용 모델

(Cost Model for Parallel Spatial Joins using Fixed Grids)

김 진 덕 † 홍 봉 희 ††
(JinDeog Kim) (BongHee Hong)

요 약 공간 데이타베이스에서 가장 비용이 큰 공간 연산자는 공간 조인이다. 공간 조인은 두개의 데이타 집합으로부터 공간적인 조건을 만족하는 두 객체 쌍의 집합을 구하는 것이다. 지난 수년동안 공간 조인의 순차 수행 시간은 많이 향상되었지만, 그 응답시간은 사용자의 요구를 만족시키지 못하고 있다. 그래서 공간조인의 병렬 수행에 대한 연구가 자연스럽게 대두되고 있다. 공간 데이타베이스 관리 시스템에서 공간 데이타의 관리의 용이성 및 부분 지역 검색의 효율성 등을 위해 고정 크기의 격자 구조를 갖는 고정 그리드를 이용할 수 있다. 그러나 지금까지 고정 그리드를 이용한 공간조인의 병렬 처리에 관한 연구는 거의 없다.

이 논문에서는 고정 그리드를 이용한 병렬 공간 조인 알고리즘의 성능을 예측하는 비용 모델을 제시하였는데, 이는 최소 경계 사각형(Minimum Bounding Rectangle : MBR)의 비교 횟수, 디스크 접근 횟수, 메시지 전송 횟수 등을 근거로 하였다. 실제 데이타 및 인위 데이타 집합을 이용한 실험은 제안한 비용 모델이 정확함을 보여주었다. 이 비용 모델은 복합 공간 질의의 비용을 예측할 필요가 있는 공간 질의 최적화를 위한 유용한 도구가 될 것으로 기대된다.

Abstract The most expensive spatial operation in patial databases is a spatial join which computes a combined table of which tuple consists of two tuples of the two tables satisfying a spatial predicate. Although the execution time of sequential processing of a spatial join has been so far considerably improved, the response time is not tolerable because of not meeting the requirements of interactive users. It is usually appropriate to use parallel processing to improve the performance of spatial join processing. In spatial databases, the fixed grids which consist of the regularly partitioned cells can be employed due to ease of managing spatial objects as well as simplicity in searching for a query region. However, the previous works on the spatial joins have not studied the parallel processing of spatial joins using fixed grids.

This paper has presented an analytical cost model that estimates the comparative performance of a parallel spatial join algorithm based on the fixed grids in terms of the number of MBR comparisons, disk accesses, and message passing. Several experiments on the synthetic and real datasets show that the proposed analytical model is very accurate. This cost model is also expected to used for implementing a very important DBMS component, called the query processsing optimizer.

1. 서 론

공간 데이타베이스에서 주로 사용되는 공간 연산은 영역 질의와 공간 조인으로서 공간 데이타 처리 시에 많은 연산이 요구되며, 연산의 복잡도(complexity)가 매우 큰 연산자이다[1]. 특히 공간 조인은 많은 처리 시간이 필요한 연산이다. 그러므로 빠른 질의 처리를 요구하는 응용에서 공간 조인의 효율적 처리방안이 필요하다. 그래서 공간 색인을 이용한 공간 조인과 병렬 처리에 관한 연구에 많은 관심이 집중되고 있다.

공간 데이타베이스 시스템에서 공간 데이타의 효율적인 관리를 위해 전체 지도를 여러 개의 레이어(layer)와 여러 개의 타일(tile) 개념을 도입하여 처리하는 경우가

† 정 회 원 : 동의대학교 컴퓨터공학과 교수
jdk@dongeui.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
bhhong@hyowon.cc.pusan.ac.kr
논문접수 : 2001년 1월 27일
심사완료 : 2001년 6월 26일

상당부분에 달한다[4]. 이 경우 한 지역에 대해 주제별로 동일한 형태(점, 선, 면)와 특성(도로, 가옥, 등)을 가지는 객체들끼리 분류하여 여러 개의 레이어로 구성한다. 또한 전체 대상 지역을 작은 단위 면적으로 분할하여 관리할 때 각각의 작은 면적이 나타내는 지도를 타일이라 한다. 일반적으로 타일은 고정 크기를 갖는 사각형으로 구성되며, 각 타일내의 데이타 관리를 위해서도 고정 크기의 격자 구조를 갖는 경우가 많다. 그러므로 공간 데이타베이스 관리 시스템에서 동일한 크기의 그리드 구조를 갖는 고정 그리드를 이용할 수도 있음을 의미한다.

그러나 아직까지 전술한 고정 그리드를 이용한 공간 조인의 병렬 처리에 관한 연구는 없었다. 지금까지의 공간 조인의 병렬 처리에 관한 연구는 주로 Quad tree와 R*-tree를 이용한 공간 조인[10,12]에 집중되었다. 또한 공간 조인의 비용모델에 관한 각종 관련 연구는 주로 R-tree를 이용한 공간 조인의 비용 추정[20,22]에 집중되어 있으며 버퍼의 크기를 고려하지 않았다. 그리고 프로세서수, 메시지 전송 횟수와 같은 병렬 처리 환경을 전혀 고려하지 않았다.

이 논문에서는 점 데이타 만을 다루는 공간 색인인 기존 그리드 파일을 확장하여 크기를 갖는 다각형 데이타에 대해 색인을 구축하는 방법 및 공간 조인의 병렬 처리 알고리즘을 참조하여, 그 병렬 공간 조인 알고리즘들의 성능을 예측하기 위한 비용모델을 제시한다. 이는 각 알고리즘들의 MBR(Minimum Bounding Rectangle : 최소 경계 사각형) 비교 연산 횟수, 디스크 접근 횟수, 메시지 전송횟수 등을 추정한다. 이 논문에서 제시한 비용모델은 데이타의 개수, 데이타의 밀도(density)와 같은 간단한 데이타 집합의 특성만을 이용하여 각 연산 횟수를 추정하며 다양한 프로세서수와 버퍼의 크기를 고려한다. 또한 R*-tree와의 성능 비교를 통해 고정 그리드를 이용한 공간 색인을 구축한 경우 병렬 공간 조인에 있어서 좋은 성능을 보임을 입증하고자 한다.

성능 평가를 위한 실험 데이타는 Sequoia 2000 벤치마크 데이타(실제 데이타)[13,23]와 [24]에서 제공하는 공간 조인 벤치마크 데이타를 사용한다. 실험은 병렬 공간 조인 알고리즘의 각 단계별 연산 횟수를 비용 모델로 추정한 연산횟수와 비교 평가 한다. 또한 R*-tree를 이용한 병렬 공간 조인 알고리즘과의 성능 비교도 수행한다. 이와 같은 실험 평가를 통해 제시한 비용 모델의 정확성을 검증하리라 본다. 또한 고정 그리드를 이용한 병렬 공간 조인 알고리즘이 공간 데이타베이스 엔진 설계에 적절히 활용될 수 있고, 비용 모델 또한 공간 질의 최적화에 유용할 것이다.

이 논문의 구성은 다음과 같다. 제 2장에서는 공간 조인의 비용 모델에 관련 연구에 대해 살펴보고, 이 논문에서 사용하는 공간조인의 정의와 고정 그리드 색인을 구축하는 방법 및 공간 조인의 병렬 처리 기법을 자세히 설명한다. 제 3장에서는 공간 조인 알고리즘의 비용 모델을 제시하고, 제 4장에서는 다양한 실험평가의 결과를 살펴보고, 비용모델의 정확성 및 R*-tree와의 비교 평가를 수행한다. 제 5장에서 결론을 맺는다.

2. 관련 연구

지금까지의 공간 조인에 관한 연구를 대분류하면 우선 조인 색인에 관한 연구[5], Z-order를 적용한 연구[6], 해쉬 조인에 관한 연구[7,8], 변환 기법을 활용한 공간 조인에 관한 연구[9] 등 매우 다양하다. 그렇지만 최근에 연구된 대부분의 공간 조인 알고리즘들은 조인의 대상이 되는 두 데이타 집합이 공간 색인화된 경우에 집중되고 있다. 주로 연구된 공간 색인 방법으로는 R-tree 및 R*-tree[2,3,10,11], PMR Quad tree[12], Grid File[13], Seeded tree[14], 그리고 Filter tree[15] 등이 있다. 또한 공간 데이타 집합 중 한 쪽 또는 두 쪽 모두 공간 색인이 생성되어 있지 않은 경우의 공간 조인 기법에 관한 연구[14,15,16,17,18]도 활발하게 진행되었다. 그리고 최근 다중 프로세서를 이용한 병렬 공간 조인[10,12,13]에 관한 연구도 증가하고 있다. 또한 공간 조인의 성능을 미리 예측하는 비용모델을 제시한 연구[1,18,19,20,21,22]도 진행되었다.

관련 연구 [1]에서는 일반 트리(generalization tree)를 이용한 알고리즘과 비용 추정을 위한 모델을 제시하였다. 일반 트리는 PART-OF관계를 이용한 트리 구조의 공간 색인 구조이다. 이 연구에서는 Nested Loop, 일반 트리, 조인 색인을 이용한 공간 조인을 각각 수행하여 제시한 비용 모델을 평가하였고, 다양한 실험을 통해 일반 트리의 성능을 검증하였다.

관련 연구[19,22]에서는 R-tree를 이용한 공간 조인 알고리즘의 디스크 접근 비용 모델을 제시하고, 실험 평가를 통해 그 우수성을 입증하였다. 그러나 이들 연구에서는 버퍼를 이용했을 경우의 구체적인 비용 추정 모델이 제시되지 않았다.

관련 연구 [20]은 3개 이상의 데이타 집합에 대해 공간 조인 연산을 수행하는 다중 공간 조인 질의의 처리를 위한 알고리즘과 선택률(Selectivity)을 기반으로 하는 비용 모델을 제시하였다. 하부 공간 색인 구조는 R-tree이고, 비용 추정은 노드 접근 횟수와 CPU 연산 시간을 근거로 하였다. 이 연구는 관련 연구 [21], [22]

로부터 유도되었다.

이 논문에서 사용하는 공간 조인이란 두 공간 데이타 집합 R과 S에서 R의 i번째 컬럼과 S의 j번째 컬럼이 공간 속성일 때 두 객체 집합의 cartesian product중에서 공간적인 조건(spatial predicate)을 만족하는 두 객체 쌍의 객체 식별자(id) 집합을 구하는 것이다[2]. 이 논문에서 제안하는 비용 모델이 적용하는 공간적인 조건은 교차 관계이다. 기본적으로 크기를 갖는 선, 면을 공간 데이타로 갖는 레이어들간의 공간 조인만을 대상으로 한다. 그리고 객체의 MBR의 중첩 확률로 판단 가능한 교차, 포함, 인접 관계 등은 이 논문에서 제안한 비용 모델이 적용가능하지만, 기타 다각형 간의 피포함 관계 및 점 레이어와 선 또는 면으로 구성된 레이어간의 공간 조인일 경우 이 논문에서 제안한 비용 모델로 비용 추정은 불가능하다.

이 논문에서 비용 모델을 제안을 위해 도입하는 병렬 공간 조인 알고리즘은 관련 연구 26에서 제시된 것으로서 간단히 ***PSJ_MA***로 명명한다. 다중할당 고정 그리드를 공간 색인으로 이용하는 PSJ_MA 알고리즘은 다음과 같은 특성이 있다.

• **다중 할당** : 먼저, 데이타 집합 공간을 일정한 면적을 갖도록 X,Y 좌표축으로 N등분(fixed grid)한다. 각 분할된 버켓을 그리드 셀이라 한다. 이 때 각 그리드 셀들은 균일한 크기(*regular extent*)를 가지며, 각 그리드 셀간에는 겹침이 존재하지 않는다. 그 뒤, 데이타 집합의 객체들을 객체의 일부분이라도 포함하는 여러 그리드 셀에 중복 참조(reference)된다.

• **단일 조인** : PSJ_MA 알고리즘은 그림 1처럼 같은 위치의 두 그리드 셀 쌍(R_i, S_i)에 대해 공간 조인을 수행한다. PSJ_MA알고리즘에서 하나의 태스크는 같은 위치의 셀 쌍의 공간 조인 작업을 의미한다. 병렬 수행을 위해 각 태스크가 하나의 프로세서에 할당된다.

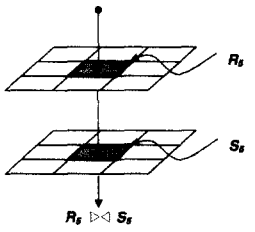


그림 1 단일 조인

PSJ_MA 알고리즘은 단일 조인이므로 여과 단계에서 각 그리드 셀은 오직 한번씩만 검색하면 된다. 따라서 공간 색인의 검색 범위가 좁혀지는 장점이 있다. 그러나 이 방식의 단점은 데이타를 중복해서 참조하기 때문에 조인의 결과가 중복된다[13]는 것이다.

PSJ_MA 알고리즘은 여과 단계를 수행한 후 생성된 후보 객체쌍의 중복을 정제 단계를 수행하기 전에 제거한다[26]. 왜냐하면 일반적으로 정제 단계의 비용이 매우 크기 때문에 중복된 정제 단계를 피하기 위해 정제 단계를 수행하기 전에 중복된 후보 객체쌍을 제거하는 것이다.

일반적으로 공간 조인의 결과로 나온 객체 식별자의 쌍의 개수가 상당히 크기 때문에 중복제거에 많은 시간이 소요된다. 이 논문에서는 중복 제거를 위해 Hyper-Cube Network형태로 구성된 프로세서들간의 Sort Merge방법[8]을 이용하여 병렬성을 높였다. 이 방법은 프로세서의 개수가 P개일 때 $\text{Log}_2\ P$ 단계만에 중복이 제거된다.

알고리즘 1은 동적 태스크 할당 기법을 이용한 PSJ_MA알고리즘을 나타낸 것이다. 이 알고리즘은 병렬 여과

알고리즘 1 PSJ_MA

```
INPUT : indices R,S(multi-assignment) & datafiles Dr, Ds
OUTPUT : A set of pairs of object_IDs
NUM of processors : N (1:MASTER, 2~N:SLAVE)
OPERATION :
01   // Filter Phase
02 IF( MASTER PROCESSOR )
03 {   Task_List1 = Generates tasks for Filter Phase
04     Sort(Task_List1) desc order by Size
05     For( i = 1 ~ Num_of_Tasks1 ) {
06         RM:Receive_Signal_of_Task(Any_Slave, Proc_Id)
07         SM : Send_Task (Proc_Id, Task_List1[i]) }
08     For(i=2~N) SM : Signal_End_of_Filter(Proc[i], NULL)
09 }
10 ELSE  /* SLAVE PROCESSORS */
11 {   While(1)
12     {   SM : Request_Signal_of_Task(Master, Slave_Id)
13         RM : Receive_Task(MASTER, task)
14         If( task = NULL ) break
15         Candidate_Pairs += Execute_Filter_Phase(task)
16 }   }
17 Elimination of Duplicated Candidates( )
18 // Refinement Phase
19 IF( MASTER PROCESSOR )
20 {   Task_List2 = Generate tasks for Refinement Phase
21     For( i = 1 ~ Num_of_Task2 ) {
22         RM : Receive_Singal_of_Task(Any_Slave, Proc_Id)
23         SM : Send_Task(PROC_ID, Task_List2[i]) }
24     For(i=2~N)SM :Signal_End_of_Refine(Proc[i], NULL)
25     For(i=2~N)RM : Collect_Result(Any_Slave, Results)
26 }
27 ELSE  /* SLAVE PROCESSORS */
28 {   while(1)
29     {   SM : Request_Signal_of_Task(Master, Slave_Id)
30         RM : Receive_Task(Master, task)
31         If( task = NULL ) break
32         Results += Execute_Refinement_Phase(task, Dr, Ds)
33     }
34     SM : Send_Result_of_Join(MASTER, Results)
35 }   // End of PSJ_MA
```

(Line 02~17 in Algorithm1) 및 병렬 정제(Line19~34 in Algorithm1)의 2단계로 수행된다. 각 단계는 또한 태스크 생성, 태스크 수행, 결과 취합의 3단계로 세분화 된다. 알고리즘에서 결과의 취합 과정은 두 번(Line17, 25 in Algorithm 1) 존재한다. 알고리즘 1의 SM은 'Send Message'를 의미하고 RM은 'Receive Message'를 의미한다. SM과 RM의 첫 인수(parameter)는 목적 프로세서를 나타내고, 두 번째 인수는 전송되는 데이타를 나타낸다. PSJ_MA 알고리즘의 여과 단계에서 디스크 접근 횟수를 줄이기 위해 버퍼를 사용한다. 버퍼의 한 슬롯(slot)의 크기는 1Kbytes이고 전체 버퍼 크기는 2K부터 256K까지 다양하게 적용할 수 있다. 그리고 페이지 교체(page replacement)는 LRU(Least Recently Used) 방식을 택한다.

3. 병렬 공간 조인 알고리즘의 비용 모델

이 장에서는 PSJ_MA 알고리즘의 비용을 예측하기 위한 비용 모델을 제안한다. 예측하고자 하는 비용은 조인 알고리즘의 여과 단계의 MBR 비교 연산 횟수, 디스크 접근 횟수, 결과의 중복 제거를 위한 메시지 전송량 등이다. 표 1은 앞으로 사용할 심볼들을 정리한 것이다.

표 1 비용 모델을 위한 심볼과 의미

Symbol	Meaning
dim	차원(2라고 가정)
D	데이타 집합의 밀도
D_k	차원 k에서의 데이타 집합의 밀도
N	그리드 해상도
P	프로세서의 수
M	데이타 집합의 객체의 수
M_i	i번째 셀의 객체의 수
M_{MA}	다중 할당 고정 그리드에 의해 색인된 객체의 수
B	버퍼의 페이지의 수
S_k	차원 k에서 각 객체의 평균 크기
G_k	차원 k에서 각 그리드의 평균 크기(extent)
G_i	i번째 그리드 셀이 차지하는 디스크 페이지 수
MC_{MA}	PSJ_MA에서 여과 단계의 MBR 비교 횟수
DA_{MA}	PSJ_MA에서 여과 단계의 디스크 접근 횟수
MP_{el}	PSJ_MA에서 후보 객체쌍의 중복 제거를 위한 메시지 전송 횟수

위의 심볼 들 중 몇 가지 사항이 조합될 수도 있다. 예를 들어 $M_{MA,R}$은 다중할당 고정 그리드에서 데이타 집합 R의 객체의 개수를 의미한다. 그리고 $M_{MA,R,i}$는 다중할당 고정 그리드에서 데이타 집합 R의 i번째 그리드 셀이 포함하고 있는 객체의 개수를 의미한다.

데이타 공간의 차원을 dim이라 할 때 단위 작업 공간은 $WS=[0,1]^{dim}$로 가정[21]한다. 각 데이타 집합은 M_R, M_S개의 객체를 갖고, D_R, D_S의 밀도를 갖는다. 이 장에서의 비용 모델은 가능한 기본적인 데이타 특성(M, D)만을 이용해 표현하고자 한다.

표 1의 '그리드 해상도'란 데이타집합 공간에서 X와 Y축의 그리드 분할선의 개수를 의미한다. 예를 들어 데이타 집합 R의 그리드 해상도가 16이라면 R의 데이타 집합 공간이 X축으로 16개의 그리드 분할선이 존재하고, Y축으로도 16개의 그리드 분할선이 존재하여 전체 그리드 셀의 개수는 256개가 되는 것이다. 이 논문에서는 X, Y축의 분할선의 개수는 항상 동일하다고 가정하며, 두 개의 데이타 집합의 그리드 해상도는 동일하다고 가정한다.

표 1의 '밀도'는 다음과 같은 의미를 갖는다. 밀도란 전체 작업 공간 중에서 객체가 차지하는 면적의 비율을 말한다. 즉, M개의 공간 객체의 전체 면적을 A라하고, 데이타 집합 공간을 B라 할 때 밀도는 A/B가 된다.

앞으로 전개할 모든 수식은 다음과 같은 기본원리[19]를 근간으로 한다. 전체 데이타 영역은 11의 윈도우를 갖는다고 가정한다.

두개의 윈도우 W_1과 W_2가 있다. 각각은 $x_1 \times y_1$와 $x_2 \times y_2$의 면적을 갖는다. 이 때, 두 윈도우가 겹칠 확률은 $\min[(x_1+x_2)\times(y_1+y_2), 1]$이다. 단, x_1, y_1, x_2, y_2 값은 1로 정규화 된 값이다.

3.1 여과 단계의 MBR 비교 횟수

PSJ_MA 알고리즘에서 MBR 비교 횟수는 다음과 같이 정의 된다.

$$MC_{MA} = \sum_{i=1}^{N^{dim}} (M_{MA,R,i} * M_{MA,S,i}) \quad (1)$$

수식 1의 $M_{MA,R,i}$와 $M_{MA,S,i}$는 각각 다중 할당 고정 그리드로 색인 했을 경우에 참조되는 객체의 개수이다. 다시말해 한 그리드의 범위 G_{MA}를 가진 N^{dim}개의 그리드 셀에 대해, 평균 객체의 크기 S를 가진 M_i개 객체가 다중할당되었을 경우의 참조되는 평균 객체의 개수는 다음과 같이 표현된다[21,22].

$$\begin{aligned} M_{MA,i} &= intersect(N^{\dim}, S, G_{MA}) \cdot M_i \\ &= N^{\dim} \prod_{k=1}^{\dim} (S_k + G_{MA,k}) \cdot M_i \end{aligned} \quad (2)$$

수식 2에서 단일 할당일 때의 한 셀의 평균 객체(M_i), 차원 k에서 각 객체의 평균 크기(S_k), 그리드 해

상도가 N일 때 차원 k에서의 한 그리드 셀의 범위($G_{MA,k}$)는 각각 다음과 같이 수식화 할 수 있다.

$$M_i = \frac{M}{N^{\dim}} \quad S_k = \left(\frac{D}{M}\right)^{1/\dim} \quad G_{MA,k} = N^{-1} \tag{3}$$

수식 1을 기준으로 수식 2,3을 조합하면, 수식 MC_{MA}를 데이타 특성 M, D 만을 이용하여 다음과 같이 나타낼 수 있다.

$$MC_{MA} = \sum_{i=1}^{N^{\dim}} \left[\left(M_{R,i} \prod_{k=1}^{\dim} \left(\left(\frac{D_{R,k}}{M_R} \right)^{\frac{1}{\dim}} + N^{-1} \right) N^{\dim} \right) * \left(M_{S,i} \prod_{k=1}^{\dim} \left(\left(\frac{D_{S,k}}{M_S} \right)^{\frac{1}{\dim}} + N^{-1} \right) N^{\dim} \right) \right] \tag{4}$$

3.2 여과 단계의 디스크 접근 횟수

버퍼의 페이지 개수를 B라 하고, 하나의 페이지는 1 Kbytes라고 가정한다. 그러므로 각 페이지에 들어갈 객체의 엔트리 정보는 50개이다. 왜냐하면 하나의 객체 엔트리는 20 바이트(MBR 정보 : 16, Pointer 정보 : 4)이기 때문이다. 그러므로 i번째 그리드 셀이 차지하는 페이지수는 다음과 같다.

$$|G_i| = \left\lceil \frac{M_{MA,i}}{50} \right\rceil \tag{5}$$

우선, 하나의 태스크를 수행할 때 페이지 수가 작은 그리드를 가진 데이타 집합이 공간 조인의 기준 데이타 집합으로 가정한다. 즉 $|G_{R,i}| < |G_{S,i}|$이면 데이타 집합 R이 기준 데이타 집합이 되는 것이다. R이 기준 데이타 집합일 때, 우선 기준 데이타 집합의 페이지 중 버퍼에 한꺼번에 넣을 수 개수(ρ)는 다음과 같이 표현된다.

$$\rho = \min\left[(B-1), |G_{R,i}|\right] \tag{6}$$

그리고 기준 데이타 집합 R은 한번에 ρ 페이지씩 버퍼에 적재되므로 전체 $\left\lceil \frac{|G_{R,i}|}{\rho} \right\rceil$ 횟수의 적재가 필요하다. 따라서 하나의 태스크 수행을 위한 디스크 페이지 접근 횟수($DA_{MA,i}$)는 수식 7과 같이 표현된다. 수식 7에서 R_Page 및 S_Page는 각각 R_i, S_i 셀의 페이지 개수이며, Num_R_Group은 R_i 페이지가 버퍼에 적재되는 횟수를 의미한다.

$$DA_{MA,i} = \{S\_Page * Num\_R\_Group + R\_Page\} = |G_{S,i}| * \left\lceil \frac{|G_{R,i}|}{\rho} \right\rceil + |G_{R,i}| \tag{7}$$

수식 5,6,7을 조합하면, 전체 디스크 페이지 접근 횟수는 다음과 같이 데이타 특성 M과 D로만 표현할 수 있다.

$$DA_{MA} = \sum_{i=1}^{N^{\dim}} DA_{MA,i} = \sum_{i=1}^{N^{\dim}} \left\{ \left\lceil \frac{M_{MA,S,i}}{50} \right\rceil * \left\lceil \frac{\left\lceil \frac{M_{MA,R,i}}{50} \right\rceil}{\min\left[(B-1), \left\lceil \frac{M_{MA,R,i}}{50} \right\rceil\right]} \right\rceil + \left\lceil \frac{M_{MA,R,i}}{50} \right\rceil \right\} \tag{8}$$

3.3 후보 객체쌍의 중복제거를 위한 메시지 전송 횟수

PSJ_MA 알고리즘은 다중 할당이므로 객체의 중복이 발생하고, 이에 따라 여과 단계 후 후보 객체쌍의 중복이 생긴다. 따라서 이들의 제거를 위한 과정에서 많은 메시지 전송량이 필요하다. 프로세서의 수가 P이고, 중복 제거 전의 후보 객체쌍을 CP_{MA} 라 하고, 중복 제거 후의 후보 객체 쌍을 CP_{SA}라 하자. CP_{SA}와 CP_{MA}를 도출하는 수식은 다음과 같다.

$$CP_{SA} = M_R * M_S \prod_{i=1}^{\dim} (S_{R,i} + S_{S,i}) \tag{9}$$

$$CP_{MA} = M_{MA,R} * M_{MA,S} \prod_{i=1}^{\dim} (S_{R,i} + S_{S,i}) \tag{10}$$

전술한 Sort Merge 방법이 요구하는 단계는 $Log_2 P$ 이다. 단계별 메시지 전송량 $MP_{el}(i)$은 다음과 같은 수식으로부터 유도된다. 이 때 유효(Valid) 프로세서수는 메시지 전송 프로세서수의 2배가 된다.

$MP_{el}(i)$={(CP_{SA}+현재 단계의 중복량)/유효 프로세서수 }*전송 프로세서수

=(CP_{SA}+현재 단계의 중복량)/2 (11)

다음은 프로세서의 수가 8개일 때 각 단계별로 중복량, 전송 프로세서 수 등을 풀이한 것이다.

1) 최초 중복량 CP_{red}=(CP_{MA} CP_{SA})

2) 1단계 전송

-전송전 중복 제거량(자체 제거량) : $\frac{CP_{red}}{P}$

-중복된 후보 객체쌍수$CP(1)$

=이전중복량-제거량 :

$$CP_{red} - \frac{CP_{red}}{P} = \frac{CP_{red}(P-1)}{P}$$

3) 2단계 전송

-전송전 중복 제거량 :

이전 중복량/(이전단계의 유효프로세서수-1)=

$$\frac{CP_{red}(P-1)}{P} * \frac{1}{P-1} = \frac{CP_{red}}{P}$$

-중복된 후보 객체쌍수 $CP(2)$:

$$= \frac{CP_{red}(P-1)}{P} - \frac{CP_{red}}{P} = \frac{CP_{red}(P-2)}{P}$$

4) 3단계 전송

-전송전 중복 제거량 :

$$\frac{CP_{red}(P-2)}{P} * \frac{1}{P/2-1} = \frac{2CP_{red}}{P}$$

-중복된 후보 객체쌍수 $CP(3)$:

$$= \frac{CP_{red}(P-2)}{P} - \frac{2CP_{red}}{P} = \frac{CP_{red}(P-2^2)}{P}$$

이상과 같이 각 단계별로 제거량, 전송 프로세서수, 중복량에는 일정한 규칙이 존재한다. 각 단계별 남아 있는 중복량은 다음과 같이 일반화 할 수 있다.

$$CP(i) = \frac{CP_{red}(P-2^{i-1})}{P} \text{ 단, } i>0 \qquad (12)$$

수식11를 기준으로 수식 9, 10, 12을 조합하면 전체 단계의 전송량은 다음과 같이 표현된다.

$$MP_{el} = \sum_{i=1}^{\lceil Log_2 P \rceil} MP_{el}(i) = \sum_{i=1}^{\lceil Log_2 P \rceil} CP_{SA} + \frac{CP_{red}(P-2^{i-1})}{P}{2}$$

$$= \sum_{i=1}^{\lceil Log_2 P \rceil} \frac{P \cdot CP_{MA} - 2^{i-1}(CP_{MA} - CP_{SA})}{2P} \qquad (13)$$

지금까지 제안한 비용 모델은 공간 데이타의 균일 분포를 가정하였다. 이 논문에서는 불균일 분포의 공간 데이타를 효율적으로 지원하기 위해, 위 수식에 밀도 표면(density surface)[22]을 적용하고자 한다. 즉, 제안한 비용 모델의 데이타 균일 분포 가정을 전체 작업 공간에서 여러 개의 소규모 지역 공간으로 변화시키는 것이다. 이는 실제 데이타의 밀도를 여러 개의 지역 밀도로 대체를 통해 이루어진다. 그리고 밀도 표면으로부터 지역공간의 객체의 개수(M')를 다음과 같이 유도한다.

$M'=(D'/D)*M$(단, D'는 actual(local) density이고,

D 는 global density)

지금까지의 비용 모델은 교차관계만을 고려하였다. 그렇지만 자주 사용되는 공간조건 중 포함 관계 및 인접을 검색하는 공간 조인일 경우에는 제시한 기본 원리를 다음과 같이 일부 변경함으로써 제안한 비용 모델의 적용이 가능하리라 판단된다.

두개의 윈도우 W_1과 W_2가 있다. 각각은 $x_1 \times y_1$와 $x_2 \times y_2$의 면적을 갖는다. 이 때, W_1이 W_2를 포함할 확률은 $\max\left[\frac{|X_1|-|X_2|}{1-|X_2|},0\right] \bullet \max\left[\frac{|Y_1|-|Y_2|}{1-|Y_2|},0\right]$이다. 또한 W_1와 W_2이 인접할 확률은 $\min[(x_1+x_2+\delta)\times(y_1+y_2+\delta), 1]$이다. 단, x_1, y_1, x_2, y_2, δ 값은 1로 정규화된 값이며, δ 값은 인접 거리이다.

4. 실험 평가

이 논문에서 MIMD(Multiple Instruction Multiple Data)구조의 IBM SP2를 이용하여 전술한 병렬 공간 조인 알고리즘의 성능과 비용 모델의 정확성을 평가한다. 실험 데이타로 실제 데이타 및 인위 데이타의 두 가지 유형을 사용하고, 프로세서의 수는 16개까지, 그리드 해상도는 16부터 256까지, 버퍼의 크기는 2Kbytes부터 256Kbytes까지 다양하게 실험하였다. 실험 결과의 기술은 여과 단계에 소요되는 MBR 비교 연산 횟수, 디스크 접근 횟수, 메시지 전송 횟수를 살펴보고 비용 모델에 의한 추정치와도 비교 평가하고자 한다. 그리고 R^*-tree와의 성능 비교 결과도 기술하고자 한다.

4.1 실험 데이타

이 논문에서 사용하는 실험 데이타는 두 가지 유형이다. 유형 1은 Sequoia 2000 Benchmark 데이타[23]로서 실제 데이타(real data)이고 불균일 분포 상태의 다각형 정보이다. 각 다각형을 구성하는 점들의 좌표 값까지 가지고 있다. 유형 2는 Scholl[24]이 공간 조인의 Benchmarking용으로 Home Page를 통해 제공하는 인위 데이타(synthetic data) 집합이다. 데이타는 Gaussian 분포 상태이며, 각 다각형의 MBR 정보만 가지고 있다. 이 논문에서는 각 데이타를 간단히 Sequoia 데이타, Scholl 데이타라 명명한다. 표 2에 각 데이타의 특성(M,D)를 정리하였다.

표 2 Sequoia 데이타의 특성

	M(R)	M(S)	D(R)	D(S)
Sequoia	41814	37793	0.39	0.42
Scholl	100000	95000	0.195	0.2

4.2 비용 모델의 성능 평가

이 논문에서 사용한 조인 대상 데이타는 균일 분포[13,23]가 아니므로 비용모델의 추정치 산정을 위해 32×32 밀도 표면[22]을 이용하였다. 실험에 사용된 버퍼의 페이지 단위는 1Kbytes이고, 비용 추정 시 사용된 프로세서의 수는 8개이다.

4.2.1 MBR 비교 연산 횟수(MC)

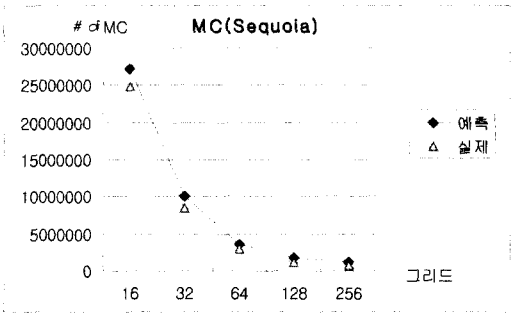
그림 2는 각 데이타에 대해 병렬 공간 조인 알고리즘의 여과 단계에서 MBR 비교 연산 횟수의 실제 수행값

과 비용 모델 추정치를 그래프화 한 것이다. 결과를 다음과 같이 요약할 수 있다.

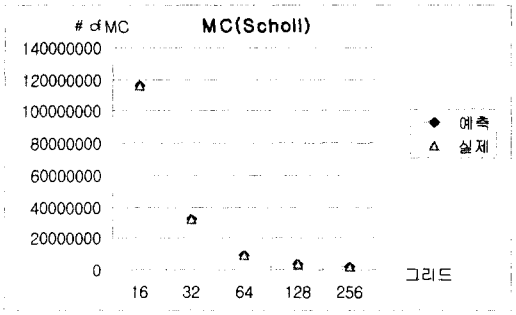
첫째, 추정치의 오차는 전체적으로 1~9% 정도로 정확한 것으로 실험결과 나타났다. 그리고 추정치의 오차는 Scholl 데이타의 경우가 훨씬 작았다. 그 이유는 Gaussian 분포인 Scholl 데이타와 달리Sequoia데이타의 경우는 불균일 분포이고, 객체의 크기 또한 매우 가변적이기 때문인 것으로 판단된다.

둘째, MC값은 데이타 집합의 개수에 비례하므로, Sequoia 데이타에 비해 Scholl 데이타를 이용한 조인의 MBR 비교 연산의 횟수에서 월등히 많은 수를 보여 주고 있다.

셋째, 그리드 해상도가 증가할수록 MC값은 감소함을 알 수 있다. 이는 PSJ_MA 알고리즘이 단일조인이므로 그리드 해상도가 증가할수록 탐색영역이 줄어들기 때문이다. 그러나 그리드 해상도가 증가할수록 객체의 중복률이 증가하므로 MC값은 완만하게 감소한다. 그렇지만 다음절에 설명되는 바와 같이 그리드 해상도가 증가할수록 디스크 접근 횟수는 증가하는 양상을 보인다.



(a) Sequoia 데이타의 MBR 비교 연산 횟수



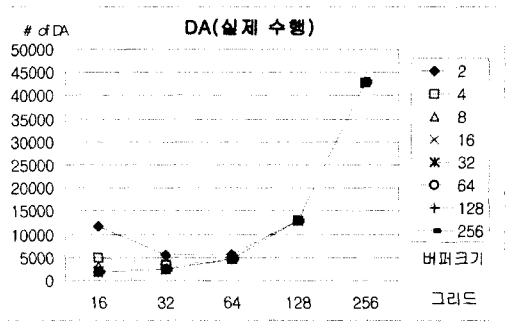
(b) Scholl 데이타의 MBR 비교 연산 횟수

그림 2 MBR 비교 연산 회수의 성능 비교

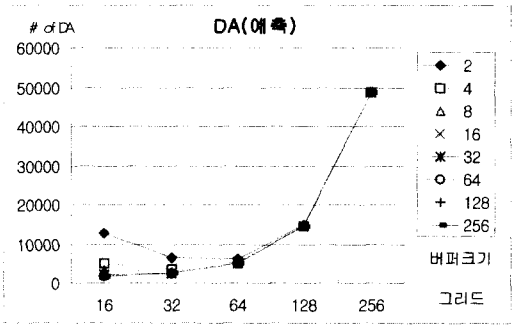
4.2.2 페이지 단위 디스크 검색 횟수(DA)

(1) Sequoia 데이타

그림 3은 Sequoia데이타에 대해 각 버퍼 크기별로 실제 수행 시 디스크 검색 횟수와 비용 모델의 추정에 의한 디스크 검색 횟수를 비교 평가하기 위한 그래프이다. 그림 3의 (a)는 실제 수행 시 디스크 접근 횟수를 표현하였고, (b)는 비용 모델에 의한 추정치를 표현하고 있다. 그림 3은 다음과 같은 몇 가지 사항으로 요약할 수 있다. 우선, 모든 경우에서 실제 수행치와 비용 모델 추정치의 값이 오차 10% 이내로 일치하였다. 그리고 그리드 해상도 128 이상에서는 버퍼 효과가 거의 없음을 알 수 있다. 그리고 그리드 해상도가 적을 경우, 버퍼의 크기가 충분히 크다면 디스크 접근 횟수를 상당히 줄일 수 있다. 그리고 그리드 해상도가 증가할수록 DA값은 증가함을 알 수 있다. 그러므로 MC값과 DA값은 Trade-Off관계에 있다.



(a) 실제 수행 시 디스크 접근 횟수



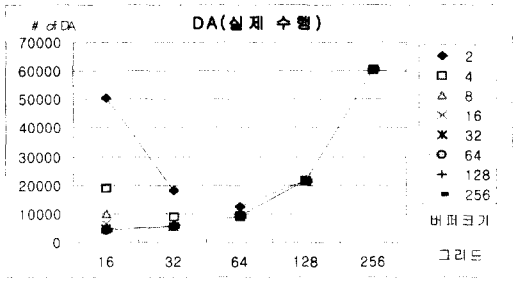
(b) 비용 모델에 의한 추정 시 디스크 접근 횟수

그림 3 디스크 접근 횟수(Sequoia)

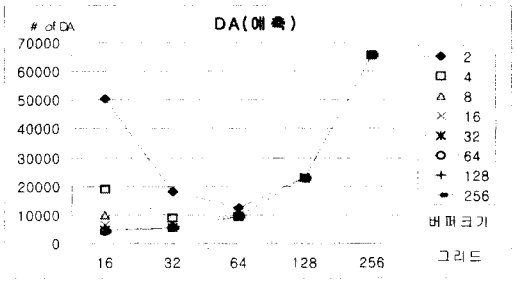
(2) Scholl 데이타

그림 4는 Scholl데이타에 대해 각 버퍼 크기별로 실

제 수행 시 디스크 검색 횟수와 비용 모델의 추정에 의한 디스크 검색 횟수를 비교 평가하기 위한 그래프이다. 그림 4는 Sequoia 데이타를 이용한 공간 조인 알고리즘의 디스크 접근 횟수 그래프와 비슷하지만, 오차율에서는 보다 낮았다. 그 이유는 Sequoia 데이타가 불균일 분포임에 반해 Scholl 데이타는 Gaussian 분포이므로 그리드 해상도가 커도 밀도 표면과의 차이값이 크지 않기 때문이다[22].



(a) 실제 수행 시 디스크 접근 횟수



(b) 비용 모델에 의한 추정 시 디스크 접근 횟수

그림 4 디스크 접근 횟수(Scholl)

(3) Sequoia 데이타와 Scholl 데이타의 비교

Sequoia 데이타를 이용한 공간 조인 알고리즘의 디스크 접근 횟수와 Scholl 데이타를 이용한 공간 조인 알고리즘의 디스크 접근 횟수는 다음과 특성이 있다.

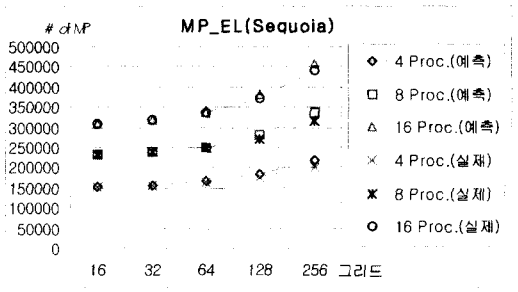
첫째, 데이타 집합을 구성하는 객체의 개수가 Scholl 데이타가 크다. 그래서 그리드 별 평균 디스크 접근 횟수가 최소화되는 지점이 Scholl인 경우 그리드 해상도 64와 128사이에 있고, Sequoia는 그리드 해상도 32와 64 사이에 있다. 이는 곧 최적의 그리드 해상도는 하나의 그리드에 포함되는 평균 객체의 수가 하나의 디스크 페이지에 포함되는 객체의 개수에 근접할 때이고, 그 그리드 해상도에서 최적의 조인 성능을 보임을 간접적으로 확인할 수 있다.

둘째, 데이타 밀도가 큰 Sequoia 데이타가 그리드 해상도가 클 경우 보다 많은 디스크 접근 횟수를 가짐을 그래프를 통해 알 수 있다. 데이타 밀도가 크다는 것은 하나의 객체가 차지하는 면적이 크다는 것이고, 이런 경우 PSJ_MA에서는 데이타의 중복이 많이 발생해 디스크 접근 횟수가 늘어난 것으로 판단된다.

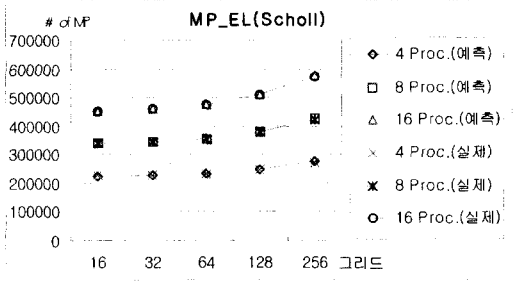
셋째, 불균일 분포 상태가 심한 Sequoia 데이타보다는 Scholl 데이타를 이용한 경우 오차율이 상대적으로 낮았다. 그 이유는 밀도 표면을 32X32로 하였기 때문이다.

4.2.3 후보 객체쌍의 중복 제거를 위한 메시지 전송 횟수(MP_{el})

그림 5는 후보 객체쌍의 중복 제거를 위한 메시지 전송 횟수 비용 수식의 추정치를 그래프화 한 것이다. 적용한 프로세서의 수는 4, 8, 16개이다. 비교 평가 결과 다음과 같은 특징이 있음을 알 수 있다. 프로세서의 수가 많을수록 MP_{el}의 값은 크다. 이것은 비용 모델 수식에서도 나타난 바와 같이Merge 단계의 증가에 기인한다. 그리드 해상도가 높을수록 MP_{el}의 값은 증가한다. 이는 객체 중복률의 증가에 기인한다. 그리고 Sequoia에 비해 Scholl 데이타의 경우 오차율이 거의 0에 가까울 정도로 비용 모델의 예측이 정확함을 알 수 있다.



(a) Sequoia 데이타



(b) Scholl 데이타

그림 5 후보 객체쌍의 중복 제거를 위한 메시지 전송 횟수

이 논문에서 제시한 비용모델들은 정규분포를 보이는 Scholl 데이타인 경우에 보다 정확한 예측이 가능하였다. 그 이유는 Sequioa 데이타는 불균일 분포인데 반해 비용 예측을 위한 밀도 표면은 32X32로 하였기 때문이다. 그럼에도 불구하고 불균일 분포를 보이는 실제데이타에 대해서도 비교적 정확한 예측이 가능함을 보여주었다.

4.3 R*-tree와의 비교

이 절에서는 PSJ_MA 알고리즘과 R*-tree와의 성능을 여과 단계의 MBR비교 횟수와 디스크 접근 횟수 측면에서 평가하고자 한다. 실험 평가를 위한 R*-tree Node의 크기는 1Kbytes이며, Top-Down 트리 매칭(Tree Matching)기법[2]을 사용하였다. 실험은 Sequoia 데이타에 대해 적용하였다.

그림 6은 Sequoia 데이타를 이용했을 경우 PSJ_MA와 R*-tree이용한 공간 조인에서의 MBR 비교 횟수를 그래프로 나타낸 것이다. PSJ_MA 알고리즘이 MBR 비교 횟수에서 대부분 좋은 성능을 보여 주었다. 고정 그리드는 불균일 분포 데이타에서 성능의 저하가 있음은 주지의 사실이다. 그러나 그림 6에서 보는 바와 같이 실제 데이타를 적용했을 경우에도 우려했던 바와 달리 PSJ_MA 알고리즘이 좋은 성능을 보였다. 이는 다음절에 제시하는 노드 이용률에서도 비슷한 결과를 보여준다.

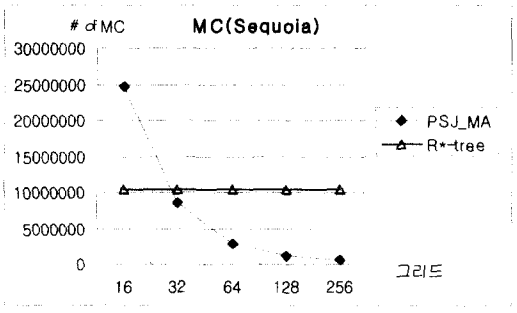


그림 6 Sequoia-MBR 비교 횟수(R*-tree와의 비교)

그림 7은 Sequoia 데이타를 이용할 경우 각각 PSJ_MA와 R*-tree 공간 조인 알고리즘의 여과 단계의 디스크 접근 횟수를 비교한 그래프이다. 실험 결과 R*-tree를 이용한 공간 조인에 비해 PSJ_MA의 여과 단계의 디스크 접근 횟수는 비교적 좋은 성능을 보았다. 그렇지만 그리드 해상도가 32인 경우에는 항상 PSJ_MA의 성능이 우수하며, 특히 적은 버퍼의 사용으로도 뛰어난 성능을 발휘하였다. 이는 Sequoia 데이타인 경우 최적의 그리드 해상도가 32 근처임을 알 수 있다. 최적의 그리드 해상도에 대해서는 다음절에서 설명하고자 한다.

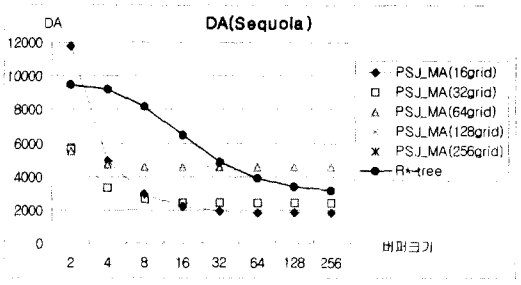
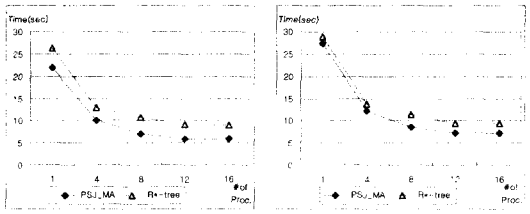


그림 7 Sequoia-디스크 접근 횟수(R*-tree와의 비교)

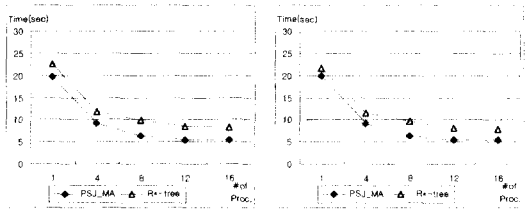
다음은 이 논문의 공간 조인 알고리즘과 R*-tree를 이용한 공간 조인 알고리즘의 수행시간을 비교하고자 한다. 한가지 미리 밝혀둘 것은 고정 그리드와 R*-tree는 그 구조가 전혀 다르고, 병렬 공간 조인 알고리즘 또한 구현상에 있어 많은 차이가 있다. 그리고 최적화 방법 또한 전혀 다르다. 그래서 직접적인 수행시간의 비교는 어렵다. 그래서 이 논문에서는 각 알고리즘에서만 적용가능한 최적화 기법은 제외하였다. 다만, R*-tree의 노드의 크기와 그리드 해상도의 Page 크기는 1K bytes로 동일하게 적용하였다.

그림 8은 PSJ_MA와 R*-tree를 이용한 공간 조인에서 여과 단계의 수행시간을 각 버퍼 크기별로 표현한 것이다. 그리드 해상도는 32일 때, 실험 결과 모든 버퍼에서 PSJ_MA의 수행 성능이 좋았다. 그리고 버퍼의 크기가 8일 때 PSJ_MA의 성대적인 성능이 가장 좋았다. 이것은 그림 8의 디스크 접근 횟수에서도 잘 나타난다. 즉, 이 논문의 PSJ_MA알고리즘은 보다 적은 버퍼를 사용하면서도 좋은 성능을 보임을 알 수 있다.

결론적으로 이 논문에서 적용한 병렬 공간 조인 알고리즘은 균일 분포 데이타에 대해 적합한 고정 그리드 공간 색인이 불균일 분포의 실제 데이타나 선택률이 굉장히 높은 데이타 집합을 경우라 할지라도 최적의 그리드 해상도에서는 PSJ_MA가 좋은 성능을 보였다. 그러므로 PSJ_MA를 이용할 경우 다음절에 설명되는 바와 같이 최적의 그리드 해상도를 선택했을 경우 R*-tree를 이용한 공간 조인 알고리즘에 비해 MBR 비교 연산 횟수와 디스크 접근 횟수 측면에서 우수한 성능을 보일 것으로 판단된다.



(a) Buffer Size : 2K (b) Buffer Size : 8K



(c) Buffer Size : 32K (d) Buffer Size : 128K

그림 8 PSJ_MA, R*-tree 여과 단계의 수행 시간 비교

4.3.1 공간 이용률(Storage Utilization)의 비교

표 3은 이 논문에서 사용하는 공간 색인(다중할당)의 셀 이용률과 R*-tree의 노드 이용률을 대비해서 보여준다. 표 3에서 그리드 해상도가 128이상일 경우는 셀 이용률이 현격하게 낮아져 제외하였다. 일반적으로 고정 그리드는 불균일 분포 데이타를 사용할 경우 셀의 이용률이 매우 떨어진다. 그러나 이 논문에서 사용한 공간 색인은 생각하던 바와 같이 심각한 수준이 아님을 알 수 있다. 더욱이 R*-tree는 트리 구조의 중간 노드가 존재하므로 같은 이용률일 경우에는 색인의 크기가 보다 증가한다.

표 3 공간 이용률 비교

		Sequoia	Scholl
고정그리드	16 grid	90%	94%
	32 grid	71%	80%
	64 grid	42%	54%
R*-tree		72%	72%

그러나 표 3에서 한가지 주의할 것은 고정 그리드에서 셀의 이용률의 증가와 공간 조인의 성능은 반드시 비례하지는 않는다는 것이다. 셀의 이용률이 높을 경우는 Overflow 페이지가 많을 확률이 높다. 반면, 셀의 이용률이 낮을 경우 공간 색인의 크기가 증가한다. 즉, 공간 조인 시 전자일 경우에는 MBR 비교 연산의 횟수가 증가하는 반면, 후자일 경우에는 디스크 접근 횟수가 증가하게 된다.

그러므로 MBR 비교 연산 횟수와 디스크 접근 횟수는 Trade-Off관계에 있다고 볼 수 있다. 그런 측면에서 셀의 이용률은 적정선을 유지할 경우 공간 조인에서 최고의 성능을 보일 것이다. 앞에서 제시한 실험 결과를 종합해 볼 때 표 3에서는 셀의 이용률이 R*-tree와 같이 평균 70%정도일 때 최고의 성능을 보였다. 즉, 셀의 이용률이 평균 70%정도일 때 최적의 그리드 해상도라 하겠다. 최적의 셀 이용률은 불균일 데이타 분포 상태에서는 조금 하향 조정되고, 균일 데이타 분포 상태에서는 상향 조정될 필요가 있다. 왜냐하면 불균일 분포 상태는 Overflow가 발생할 빈도가 높아지기 때문이다. 데이타의 분포 상태는 밀도의 표준 편차로 나타낼 수 있다. 표 3에서 Sequoia 데이타는 32가 최적의 그리드 해상도이고, Scholl 데이타는 32와 64사이가 최적의 그리드 해상도이다.

셀의 이용률은 데이타 집합내의 객체의 개수에 비례하고, 그리드 해상도에 반비례한다. 그러므로 셀 이용률은 아래 수식과 같이 표현된다. 수식에서 M_{MA}는 다중할당으로 색인되었을 때의 객체의 개수를 의미하고, N은 그리드 해상도이고, 50은 한 페이지에 들어갈 객체의 Entry 수이다.

5. 결론 및 향후 연구 과제

이 논문에서는 고정 그리드를 이용한 병렬 공간 조인 알고리즘에서 여과 단계의 각종 비용을 추정하는 비용 모델을 제시하였다. 그리고 실제 데이타 및 인위 데이타를 이용한 실험을 통해 제안한 병렬 공간 조인 알고리즘의 비용 모델의 성능을 검증하였다.

이 논문에서 제안한 비용 모델은 여과 단계의 MBR 비교 연산 횟수, 페이지 단위 디스크 접근 횟수, 대응 그리드 결정을 위한 MBR 비교 연산 횟수, 후보 객체쌍의 중복 제거를 위한 메시지 전송 횟수 등을 추정하는 수식이다. 이들 비용 모델을 통해 그리드 해상도가 매우 크고, 버퍼가 매우 큰 경우를 제외하고는 오차율 10% 이내에 해당하는 정확한 예측이 가능하였다. 또한 R*-tree를 이용한 공간 조인 알고리즘과의 비교 평가에서 이 논문의 병렬 공간 조인 알고리즘이 적은 버퍼를 사용하고도 좋은 성능을 보임을 알 수 있었다.

또한 MBR 비교 연산 횟수와 디스크 접근 횟수는 Trade-Off관계가 있음을 실험 결과를 통해 알 수 있었다. 그러므로 데이타 집합의 객체의 개수에 따라 최적의

성능을 보이는 그리드 해상도가 존재함을 확인하였다.

이 논문의 연구 결과는 최근 활발히 진행되고 있는 병렬 공간 데이타 베이스 엔진의 하부 구조 설계 및 공간 연산자 설계에 잘 활용되리라 본다. 특히 일반적으로 전체 도면을 균등한 간격을 갖는 그리드로 분할하여 데이타를 관리하는 시스템에서는 이 논문에서 제시한 병렬 공간 조인 알고리즘이 그대로 적용 가능하다. 또한 비용이 매우 큰 연산자인 공간 조인에 관한 비용을 예측해야 하는 공간 질의 처리 최적화기를 설계함에 있어 이 논문에서 제시한 비용모델을 매우 유용하리라 본다.

참 고 문 헌

[1] O. Gnther, *Efficient Computation of Spatial Joins*, Proc. of Int. Conf. on Data Engineering, pp. 50-59, 1993

[2] T. Brinkhoff, H.P. Kriegel, R.Schneider, B. Seeger, *Efficient Processing of Spatial Joins Using R-trees*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 237-246, 1993

[3] T. Brinkhoff, H.P. Kriegel, R.Schneider, B. Seeger, *Multi-Step Processing of Spatial Joins*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 197-208, 1994

[4] R. Laurini, D. Thompson, *Fundamentals of Spatial Information Systems*, Academic Press, 1992

[5] D. Rotem, *Spatial Join Indices*, Proc. of Int. Conf. on Data Engineering, pp. 500-509, 1991

[6] J. A. Orestein, *Redundancy in spatial databases*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 294-305, 1989

[7] M.L. Lo, C.V. Ravishankar, *Spatial Hash-Joins*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 247-258, 1996

[8] J.M. Patel, D.J. Dewitt, *Partition based spatial merge join*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 259-270, 1996

[9] J.W. Song, K.Y. Whang, Y.K.Lee, M.J.Lee, S.W. Kim, *Spatial Join Processing Using Corner Trans-* 688-695, 1999

[10] T. Brinkhoff, H.P. Kriegel, B. Seeger, *Parallel Processing of Spatial Joins Using R-trees*, Proc. of Int. Conf. on Data Engineering, pp. 258-265, 1996

[11] Y.W. Huang, N. Jing, E. A. Rundensteiner, *Spatial Joins using R-tree : Breadth-first traversal with global optimizations*, Proc. of Int. Conf. on VLDB, pp. 396-405, 1997

[12] E.G. Hoel, H. Samet, *Data-Parallel Spatial Join Algorithms*, Proc. of Int. Conf. on Parallel Processing, pp. 227-234, 1994

[13] X. Zhou, D. J. Abel, David Truffet, *Data Partitioning for Parallel Spatial Join Processing*, Proc. of Int. Conf. on SSD, pp. 178-196, 1997

[14] M.L. Lo, C.V. Ravishankar, *Spatial Joins Using Seeded Trees*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 209-220, 1994

[15] N. Koudas, K.C. Sevcik, *Size Separation Spatial Join*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 324-335, 1997

[16] W.G. Aref, H. Samet, *Cascaded Spatial Join Algorithms with Spatially Sorted Output*, Proc. of Int. Conf. on ACM GIS, pp. 17-24, 1997

[17] L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, J.S. Vitter, *Scalable Sweeping Based Spatial Join*, Proc. of Int. Conf. on VLDB, pp. 570-581, 1998

[18] N. Mamoulis, D. Papadias, *Integration of Spatial Join Algorithms for Processing Multiple Inputs*, Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 1-12, 1999

[19] Y.W. Huang, N. Jing, E. A. Rundensteiner, *A Cost Model for Estimating the Performance of Spatial Joins Using R-trees*, Proc. of Int. Conf. on SSDBM, pp. 30-38, 1997

[20] D. Papadias, N. Mamoulis, Y. Theodoridis, *Processing and Optimization of Multiway Spatial Joins Using R-trees*, Proc. of Int. Conf. on PODS, pp. 44-55, 1999

[21] Y. Theodoridis, T. Sellis, *A Model for the Prediction of R-tree Performance*, Proc. of Int. Symp. on ACM PODS, pp. 161-171, 1996.

[22] Y. Theodoridis, E. Stefanakis, T. Sellis, *Cost Models for Join Queries in Spatial Databases*, Proc. of Int. Conf. on Data Engineering, pp.476-483, 1998.

[23] http://epoch.cs.berkeley.edu:8000/sequoia/benchmark/polygon/, Sequoia 2000 FTP server home page

[24] http://www.enst.fr/~bdtest/sigbench/index.html, Spatial Join Benchmarking home page

[25] D.J. DeWitt, *DIRECT-A Multiprocessor Organization for Supporting Relational Database Management System*, IEEE Trans. on Computers, pp. 395-406, 1979

[26] J.D. Kim, B.H. Hong, *Parallel Spatial Join Algorithms using Grid Files*, Proc. of Int. Symp. on DANTE'99, pp. 127-135, 1999.

김 진 덕

1993년 2월 부산대학교 컴퓨터공학과 졸업(학사). 1995년 2월 부산대학교 대학원 컴퓨터공학과 졸업(석사). 2000년 8월 부산대학교 대학원 컴퓨터공학과 졸업(박사) 1998년 3월 ~ 2001년 2월 부산정보대학 정보통신계열 전임강사. 2001년 3월 ~ 현재 동의대학교 컴퓨터공학과 전임강사. 관심분야는 객체지향 DB, 지리정보시스템, 공간 질의어, 공간 색인

홍 봉 희

정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조