병렬 공간 조인 시 정적 부하 균등화를 위한 작업 생성 및 할당 방법

(Task Creation and Allocation for Static Load Balancing in Parallel Spatial Join)

박 윤 필 † 염 근 혁 ††
(Yun Phil Park) (Keunhyuk Yeom)

요 약 현재의 지리 정보 시스템(GIS : Geographical Information System)은 컴퓨터 응용 시스템의 중요한 분야로서 도시 정보 시스템, 교통 정보 시스템 등에 활용되고 있다. 이들 응용 분야는 대용량의 공간 데이타를 다루기 때문에 효율적인 공간 연산 수행을 위한 기본 연산자를 필요로 한다. 특히, 기본 연산자 중에서 공간 조인은 연산에 참여하는 객체의 수가 증가함에 따라 수행 시간이 지수적으로 증가하는 특성을 가지고 있으므로 빠른 응답 시간을 요구하는 시스템에는 부적합하다. 따라서 이러한 요구 사항을 만족시키기 위해서는 공간 조인의 효율적인 병렬 수행이 필요하다. 본 논문에서는 공간 조인의 효율적인 병렬 수행을 위하여 정적 부하 균등화를 위한 작업 생성 및 할당 방법을 제시한다. 이 방법은 공간 지역성을 고려하여 작업을 생성하고, 비용 모델을 통하여 작업량을 예측하여 표현한 뒤 작업 그래프로 나타낸다. 그리고 생성된 작업 그래프를 그래프 분할 알고리즘을 통하여 균등하게 할당한다. 본 논문에서 제시된 방법은 독일 Parsytec사의 CC16 병렬머쉰에서 실험한 결과로 볼 때, 기존의 정적 할당을 통한 작업 생성 및 할당 방법에 비하여 각 프로세서간의 작업 수행시간의 편차를 줄임으로써 부하 균등화의 효과를 가져온다.

Abstract Recently, a GIS has been applicable to the most important computer applications such as urban information systems and transportation information systems. These applications require spatial operations for an efficient management of a large volume of data. In particular, a spatial join among basic operations has the property that its response time is increased exponentially according to the number of spatial objects included in the operation. Therefore, it is not proper to the systems demanding the fast response time. To satisfy these requirements, the efficient parallel processing of spatial joins has been required. In this paper, the efficient method for creating and allocating tasks to balance statically the load of each processor in a parallel spatial join is presented. A task graph is developed in which a vertex weight is calculated by the cost model I have proposed. Then, it is partitioned through a graph partitioning algorithm. According to the experiments in CC16 parallel machine, our method made an improvement in the static load balance by decreasing the variance of a task execution time on each processor.

1. 서 론

지리 정보 시스템(GIS)은 지리학 분야와 마찬가지로 다양한 분야의 많은 기술들이 결합되어 이루어지므로 정의나 개념, 구현이 다양하다. 초창기에는 컴퓨터 기술 및 전자 지도 제작 방식의 발전 등에 의하여 GIS는 지도 제작 성격이 강하였다. 그러나 정보 기술 분야의 기술이 발달되어 GIS분야에 적용되면서 기존의 GIS 성격과 모습이 달라지게 되었다. 특히 최근에는 새로운 공간 데이타베이스 시스템의 형태로서도 주목받게 되었다.

공간 데이타 베이스 시스템(Spatial DataBase System: SDBS)은 점(point), 선(line), 다각형(polygon)

· 본 연구는 1997년도 한국학술진흥재단 대학부설연구소과제 연구비에 의하여 연구되었음.
† 비 회 원 : 한국증권전산(주) 연구원
sinawee@koscom.co.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
yeom@hyowon.pusan.ac.kr
논문접수 : 2000년 3월 24일
심사완료 : 2001년 5월 25일

등의 기하학적(geometric)타입으로 표현된 공간 데이타를 다룬다. 초창기에는 기존의 데이타 베이스에서 공간 데이타를 처리 할 수 없었으므로 이를 처리하기 위하여 응용 시스템을 파일 시스템(file system)상에 직접 구성하였다. 이러한 방법은 고급 데이타 정의, 질의 유연성, 트랜잭션 관리 등 DBMS의 강력한 이점을 제공하지 못하였다. 다음 단계에서는 관계형(relational) DBMS를 하부구조로 하여 공간 데이타를 처리하기 위한 부분을 추가하여 사용하려는 시도가 있었다. 이는 현재 대부분의 상용 GIS(MapInfo, ARC/INFO, MapBasic 등)가 채택하고 있는 방식으로 비공간 데이타는 DBMS가 담당하고 공간 데이타 처리는 분리된 공간 서브 시스템이 관리하도록 하여, 독립된 두개의 서브 시스템을 통합하여 기능을 구현하고 있다. 그러나 여기서 가장 큰 문제점은 한 개의 질의가 하부에서 두 개로 나누어져서 각각 처리되므로 전체적 최적화가 불가능해지게 되어 작업 처리의 효율성이 떨어진다는 점이다[1]. 즉, 파일 시스템 차원이나 기존 관계형 데이타베이스 시스템을 단순 확장한 차원에서는 한계가 있다. 그러한 이유로 공간 데이타 처리의 요구 기능들을 효율적으로 지원할 수 있는 통합된 공간 데이타 베이스 시스템의 필요성이 대두되었다.

공간 데이타베이스 시스템의 이용 측면에서는 기본적인 연산자들 중 객체를 삽입, 삭제 또는 수정 등의 데이타베이스 갱신 연산보다는 자주 발생하고 사용자 처리 시간에 민감한 점 질의(point query), 창 질의(window query), 공간 조인 연산 등과 같은 기본 공간 연산자에 더 많은 관심이 집중되고 있다. 그리고 공간 연산자 중에서 공간 조인은 두 개 이상의 공간 데이타 집합에 대하여 cartesian product의 부분 집합을 구하는 연산으로 정의된다. 공간 조인의 특징으로는 점 질의, 창 질의와는 달리 다중 질의(multiple scan query)이기 때문에 참여하는 객체의 수가 증가함에 따라 연산 시간이 지수적으로 증가하게 되고 공간 조인 조건(교차, 포함관계 등)을 검사하기 위해 값비싼 비용이 드는 공간 객체들 간의 기하학적 계산(geometric computation)을 수행하게 된다. 현재 이를 효율적으로 처리하기 위한 연구가 국내외에서 다양하게 이루어지고 있다[2-7][18][19].

공간 조인에 관련된 연구들은 크게 단일 프로세서와 다중 프로세서 환경에서의 연구들로 나누어 볼 수 있다. 단일 프로세서에서는 공간 조인의 직렬 수행 시 빠른 응답을 가지기 위한 내용들[2][3][4]이 있었고, 최근에는 다중 프로세스를 통해 공간 조인을 병렬 수행함으로써 성능의 향상을 이루기 위한 연구들[5][6][7][18][19]이 이루어졌다. 본 논문에서는 다중 프로세서 환경에서 공간 조인을 효율적으로 수행하기 위한 작업 생성 및 할당 방법을 제시하여 정적 부하 균등화를 통한 성능 향상을 이루고자한다.

2. 관련연구

지금까지 공간 조인에 관한 연구는 단일 프로세서 환경에서의 연구가 주류를 이루며 많은 성능 향상을 가져왔다. 그러나 실시간에 빠른 응답시간을 요구하는 사용자의 기대를 만족시키지 못하고 있다. 이러한 이유로 최근에는 병렬 컴퓨터 환경에서 성능 향상을 이루기 위한 연구가 국내외적으로 많이 이루어지고 있다.

일반적으로 공간 조인의 병렬 수행을 위해서는 데이타 분할(data declustering)에 대한 연구[8]와 함께 declustered 데이타를 가지고 공간 조인 알고리즘을 고안하게 된다. 대표적인 예로 독일 뮌헨 대학에서는 공간 조인의 병렬 수행(가상 공유 메모리, 공유 디스크 구조)[5]을 위하여 작업 생성, 작업 할당, 정제 단계(기하학적 계산을 통한 공간 객체들간의 교차 여부 검사)와 같은 3가지 단계로 수행한다. 작업 생성 단계에서는 공간 인덱스 구조로 R*-tree[9]를 구성하고, 이를 이용하여 루트 노드로부터 tree matching 과정 시 중간 노드의 MBR(Minium Boundary Rectangle)을 비교하여 서로 교차하는 후보 쌍을 작업으로 만들고, 생성된 작업의 수가 병렬 수행할 프로세서의 수보다 많아질 경우 작업 생성 단계를 마친다. 그리고 생성된 작업들을 프로세서들에게 할당하기 위한 방법으로는 전역 버퍼(global buffer)를 이용한 동적 작업 재 할당 방식을 제시하였다.

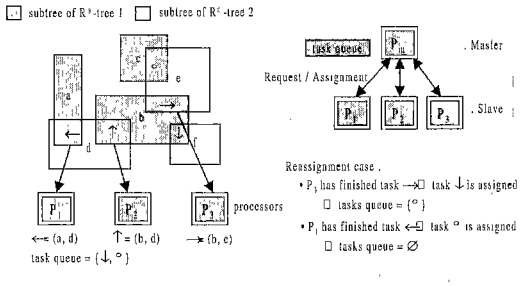


그림 1 동적 작업 재할당 방법[5]

그림 1은 동적 작업 재할당의 예를 보여주는 것으로 Master/Slave 구조로 수행된다. 여기서, Master 프로세서(Pm)는 R*-tree 중간 노드의 MBR이 교차하는 쌍({(a, d), (b, d), (b, e), (b, f), (c, e)})들을 작업({①,②,③,④,⑤})으로 생성한다. 또한, Pm은 생성한 작업들을

Slave 프로세서(P_1, P_2, P_3)에게 하나씩 할당하고, 할당되지 않는 나머지 작업들을 task queue에 저장한다. Slave 프로세서가 할당된 작업을 마치면, P_m에게 작업을 요구하게 되며, 이러한 요구에 의해 P_m은 task queue에서 작업을 꺼내어 Slave 프로세서에게 재할당한다.

전역 버퍼는 공간 인덱스 노드 페이지의 공유를 위하여 사용된다. 이는 각각의 프로세서들이 가진 지역 버퍼를 다른 프로세서들도 자신의 버퍼처럼 접근할 수 있도록 하여 하나의 전역 버퍼처럼 사용하도록 한다. 따라서, 동일한 페이지를 여러 프로세서가 중복하여 디스크 접근을 하는 경우를 없앤다. 예를 들면, 그림 1에서 P_1에게 할당된 작업 ①과 P_2에게 할당된 작업 ②는 중간 노드 d를 공유하고 있으므로 디스크로부터 각각 프로세서(P_1, P_2)가 중간 노드 d를 중복하여 읽어들이는 오버헤드가 존재하지만 전역 버퍼에 의하여, 한번 사용한 노드의 페이지는 다른 프로세서들에게 공유되므로 오버헤드를 없앤다.

공간 조인 병렬 수행 시 작업 할당에 관한 국내 연구로는 비 공유 메모리, 공유 디스크 구조에서 디스크 병목 현상을 해결하기 위한 방법으로 객체 캐쉬를 두어 공간 객체를 공유하였고, 작업 할당 시에는 공간적 지역성을 고려하여 동일 공간 객체에 대하여 여러 프로세서가 중복 디스크 접근 경우를 배제함으로써 수행 시간의 향상을 이루는 방법[6][19]을 제시하였다. 다른 연구[7]에서는 데이타 병렬 처리 작업의 효과적인 균등 분배를 위하여 이웃 공간과의 종속성을 줄이고자 비 겹침 정규 분할 방식의 그리드(grid) 파일을 공간 색인 구조로 이용하였다. 그리고 작업 균등 분배를 위해서 각 그리드 셀에 속한 공간 객체의 수를 기반으로 작업량을 계산하여 준동적 할당을 통하여 작업 균등 분배 알고리즘을 제시하였을 뿐만 아니라 단일/다중할당 공간 색인에서 병렬 공간조인의 성능평가[18]가 이루어졌다.

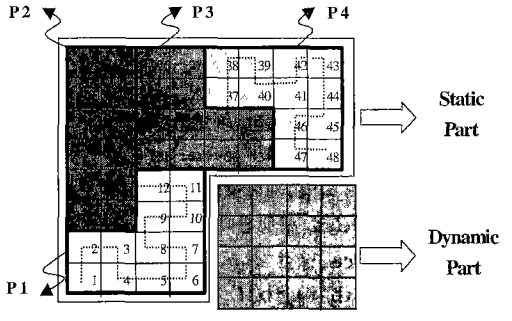


그림 2 준동적 할당의 예[18]

여기서는 단일할당 공간색인으로 R-tree를 사용하였고, 다중할당 공간색인으로는 Quad-tree를 사용하여 평가하였다. 또한 부하 평준화 기법으로 제시된 방법은 그림 2와 같이 Quad-tree를 기반으로 하여 작업량의 75%는 정적기반으로 할당하고 나머지 25%는 동적으로 할당함으로써 기존의 작업 수 기반의 정적 할당 및 동적 할당에만 의존한 방법에 비하여 높은 부하평준화를 이룰 수 있었다.

R-tree 기반의 연구에서는 중간 인덱스 구조에서 작업을 생성하여 병렬화하는 것이 가장 효율적인 방법이지만, 중간 인덱스 구조에서 작업을 생성하기 때문에 부하를 예측할 수 없으므로 병렬 수행시의 부하 균등화 정책이 동적 재 할당에 많이 의존하였다. 따라서 정적 할당에 관련된 연구가 거의 이루어지지 않았다.

3. 정적 부하 균등화를 위한 작업 생성 및 할당 방법

본 논문에서는 공간 조인의 병렬 수행을 크게 작업 생성(task creation), 작업 할당(task assignment), 작업 수행(task execution)의 3가지 단계를 통하여 그림 3와 같이 Master/Slave 구조로 수행한다. Master/Slave 구조는 작업 생성과 할당 단계에서는 Master 프로세서에 의해 직렬 수행되어지고, 작업 수행 단계에서는 Slave 프로세서들에 의해 병렬 수행되어진다.

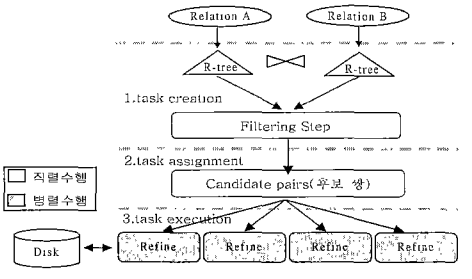


그림 3 공간 조인의 병렬 수행 과정

정적 부하 균등화를 위한 작업 생성 및 할당 단계는 다음과 같이 정리 할 수 있다.

1. 작업 생성 단계 : 정제 단계를 위한 작업을 생성하는 단계로서, 공간 객체의 집합들(Relation A, Relation B)을 공간 인덱스 구조(R-tree)로 구성하고, 루트 노드로부터 MBR을 비교한다. 이러한 과정을 통하여 MBR이 교차하지 않는 쌍은 더 이상의 연산이 필요

하지 않으므로 여과 대상이 된다. 여과 단계를 거쳐 생성된 작업은(본 논문에서 작업은 실제의 공간 데이타를 이용하여 교차 여부 계산을 수행해야하는 공간 객체 후보 쌍으로 간주 함) 작업 그래프로 표현된다.

2. 작업 할당 단계 : 생성된 작업들을 다중 프로세서에게 할당하는 단계로서, 작업 그래프로 표현된 작업을 각각의 프로세서에게 작업량을 균등화와 함께 공간적 지역성을 유지되도록 할당한다. 이때 그래프 분할 알고리즘을 이용하여 작업 그래프를 목적에 맞게 분할하고, 그 정보를 이용하여 Master 프로세서는 메시지 기반으로 분할된 작업들을 Slave 프로세서들에게 전달한다.

3. 작업 수행 단계 : Master 프로세서에 의해 Slave 프로세서들에게 할당된 작업들이 실제로 교차하는지 여부를 가리기 위한 단계로서, 실제의 공간 데이타를 이용하여 기하학적 계산을 통해 공간 조건이 결정된다. 이는 정제 단계라고도 하는데, 디스크 접근, 기하학적 계산과 같은 값비싼 비용의 연산이 필요하므로 각 단계 중에서 가장 많은 시간이 소요되기 때문에 병렬 수행된다.

작업 생성과 할당 시 효율적인 병렬 수행을 위해서 그림 4과 같이 R-tree를 이용한 MBR join, 작업 그래프 생성, 비용 모델(cost model) 적용, 그래프 분할 및 할당 등의 세부적 단계를 거쳐 수행된다.

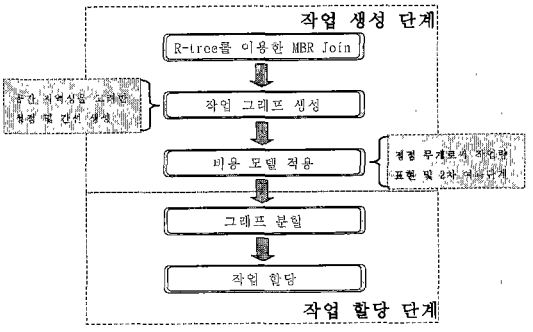


그림 4 작업 생성 및 할당을 위한 단계

3.1 R-tree를 이용한 MBR Join 단계

대용량의 공간 데이타 집합들을 효율적으로 다루기 위해서는 공간 인덱스 구조가 필요한데, 그 중 객체 중심의 영역 분할 방식을 취하고 있는 R-tree[10]가 일반적으로 사용된다. 이는 균형화된 트리이며 동적인 삽입과 삭제가 가능하고 각 노드가 같은 구조를 이루고 있다. 그리고 각 노드마다 최소 경계 사각형(MBR)을 가지고 있기 때문에 다른 색인 구조에 비하여 공간 효율이 좋고 검색 속도가 빠른 특징이 있다.

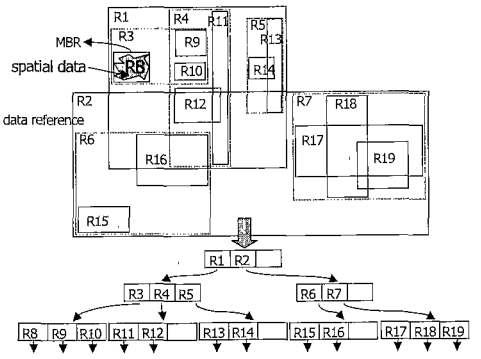


그림 5 공간 인덱스 구조 : R-tree

R-tree의 노드 구조는 그림 5와 같이 나타나며, 비 단말(non-leaf) 노드는 (ref, rect) 형태의 entry를 가지고 있다. 여기서 ref는 자식 노드를 가리키며, rect는 자식 노드의 entry에 있는 모든 객체들의 MBR을 포함하는 최소 영역 사각형(MBR)에 대한 정보를 가지고 있다.

예를 들면, 그림 5에서 루트 레벨의 R1노드는 {R3, R4, R5}를 자식 노드로 가지고 있으며, R1은 자식 노드의 MBR을 모두 포함하는 MBR을 가지게 된다. 또한 R3는 중간 노드로서 {R8, R9, R10}를 자식 노드로 가진다. 여기서 {R8, R9, R10}는 단말(leaf) 노드로서 이들 각각은 실제의 공간 데이타의 저장 장소를 가리키게 된다(즉, R8은 실제 13각형의 공간 데이타가 저장된 장소를 가리키고 있다). 단말 노드는 같은 형태의 entry를 가지고 있지만, ref는 실제의 공간 객체를 참조하고, rect은 공간 객체의 MBR 정보를 가지고 있다.

본 논문에서는 R-tree의 루트 레벨 노드로부터 MBR 교차 여부(MBR join)를 통하여 후보 쌍을 만든다. 여기서 후보 쌍은 단말 노드의 MBR이 교차하는 쌍으로 공간 조인의 조건을 만족할 가능성을 가지고 있으므로 정제 단계를 거쳐야 한다. 이 때 사용되는 탐색 방법으로 가장 보편적으로 사용되는 깊이 우선 탐색 방법(Depth first search)을 사용하며, 탐색 공간을 줄이기 위하여 MBR이 교차되는 영역이 포함된 노드들에 관하여 연산을 수행한다.

3.2 작업 그래프 생성

MBR join 단계에서 생성된 작업들은 실제의 공간 데이타를 이용하여 기하학적 계산(geometric computation : 본 논문에서는 교차 연산에 한함)을 병렬 수행해야 한다. 이때 작업들을 표현하기 위해서, 본 논문에서는 작업 그래프 $G = (V, E)$를 정의한다. V는 정점(vertex)의 집합($v_i \in V$, $1 \le i \le$ 그래프 노드 수의 총

합)으로서, v_i에는 두 R-tree의 단말 노드의 부모 노드 MBR이 교차하는 영역에 포함되는 후보 쌍(하나의 작업으로 간주함)들의 집합으로 정의된다. G는 weighted graph로서 각각의 v_i는 무게 $W(v_i)$를 가지게 된다. 이는 v_i가 공간 지역성이 보존된 작업들의 집합으로 표현되기에 작업량을 표현할 수 있다. E는 단말 노드의 부모 노드를 공유하는 정점들을 연결하는 간선(edge)들의 집합으로 작업들간의 공간 데이타의 공유 가능성을 나타내기 위한 것으로 정의한다.

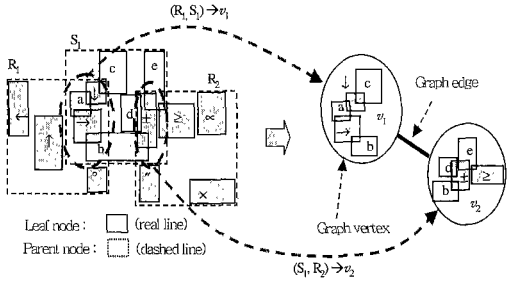


그림 6 작업 그래프 생성

예를 들면, 그림 6에서 그래프 정점(vertex) v_1는 R-tree 단말 노드의 부모 노드 MBR R1, S1이 교차하는 모두 4개의 자식 노드의 쌍(T_k = {(a, ③), (a, ④), (b, ③), (c, ④)} : 정점에 포함된 작업들의 집합, k = 작업 인덱스)을 가지게 된다. 그리고 이 중에서 (a, ③)와 (b, ③)의 경우에는 작업들간에 공간 객체를 공유하기 때문에 만일 서로 다른 프로세서에게 할당되었을 때는 같은 공간 객체(③에 해당되는 공간 객체)에 관하여 디스크를 중복 접근하는 경우가 발생하게 된다. 따라서 이러한 오버헤드를 줄이기 위해서는 공간적 지역성이 보존되어야 한다. 이를 위하여 공간적으로 인접한 작업들을 v_1로 클러스터링[v_1 = {(R_1, S_1)에 속하는 작업들} = $\sum_{k=1}^{n_1} T_k$ (n_1 : v_1에 속한 작업의 총수= 4) = {(a, ③), (a, ④), (b, ③), (c, ④)}]함으로써 공간객체를 공유하는 작업들이 하나의 단위로 처리된다.

그래프 간선은 하나의 공간 객체가 서로 다른 그래프 정점에 포함된 작업들에 공유 될 수 있는 가능성[그림 5에서 작업 (③, b)와 (⑥. b)의 경우, 공간 객체 b는 서로 다른 그래프의 정점(v_1, v_2)에게 공유됨]을 나타낸다. 따라서 그래프 분할 시 간선으로 연결된 경우 같은 프로세서에게 할당 될 확률이 커지게 되므로 공간 지역성을 보존하는 효과를 거둘 수 있다.

3.3 비용 모델을 이용한 작업량 표현

지금까지의 방법에서는 작업 수에 의존하여 작업량을 표현하였다. 따라서, 작업들간의 다양한 수행 시간의 차를 표현하지 못하므로 부하 균등화의 저해 요인이 되었다. 그러므로 작업량을 효과적으로 표현하기 위해서는 정제 단계 시 사용되는 연산에 대한 비용 모델이 필요하다.

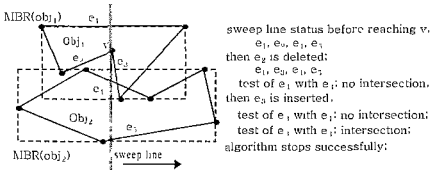


그림 7 교차 여부 검사 알고리즘 : plane-sweep 알고리즘[3]

본 논문에서는 교차 연산 수행을 위하여 가장 보편적인 plane-sweep 알고리즘[11]을 사용한다. 이는 그림 7에서와 같이 두 다각형간의 교차 여부를 계산하기 위해서 사용되는 알고리즘으로 크게 전처리 단계와 라인 교차 여부검사 단계로 나눌 수 있다.

전처리 단계에서는 디스크로부터 읽어온 공간 객체를 다각형의 정보로 만들고 모든 꼭지점들을 x축을 기준으로 정렬한다. 그리고 라인 교차 여부검사 단계에서는 수직선(sweep line)이 왼쪽에서부터 지나면서 만나는 꼭지점에 대해 수직선과 교차하는 간선(edge)들을 동적인 자료 구조에 저장한다. 여기서 만약 서로 다른 다각형의 간선들이 교차되는 것이 찾아진다면 수행을 마치게 된다.

예를 들면, 그림 7의 경우에는 수직선이 점 v에 닿기 전의 수직선과 만나는 모든 간선들은 수직선 상태(sweep line status)에 e_1, e_2, e_4, e_5와 같이 y축에 정렬되어 저장된다. 그리고 수직선이 v에 도달하게 되면 이 때 e_2는 더 이상 수직선과 교차하지 않으므로 제거되고 e_2가 제거될 때 e_2와 이웃하는 간선인 e_1과 e_4가 교차하는지 검사하게 된다. 여기서 e_1과 e_4가 서로 교차하지 않으므로 새롭게 수직선과 교차하게되는 e_3이 삽입된다. 이 때 수직선 상태(sweep line status)는 e_1, e_3, e_4, e_5이 된다. 따라서 e_3의 이웃인 e_1과 e_4에 대하여 검사(교차 검사는 수직선 상태에 저장된 모든 간선에 대하여 이루어지는 것이 아니라, 새로 변경되어 이웃하게 되는 간선에 대해서만 이루어진다)를 하게된다. 마침내 e_3과 e_4가 서로 교차하게되고 알고리즘은 성공적으로 마치게

된다. 그리고 보다 효율적인 처리를 위하여 MBR이 서로 교차하는 영역에 포함된 간선들에 한하여 알고리즘을 적용한다.

본 논문에서는 작업량 표현을 표현하기 위한 비용 모델을 제시한다. 이는 정제 단계 시 사용되는 알고리즘을 기반으로 하여 ***computation*** 비용($\mathbf{Cost_{pre}}$ + $\mathbf{Cost_{inter}}$)과 **I/O** 비용($\mathbf{Cost_{disk}}$)으로 표현된다. 작업량 표현은 작업(T_k)들이 공간 지역성을 보존을 위해 소집합으로 묶여서 v_i가 되기 때문에 정점의 무게 값 $W(v_i)$으로 표현된다

• $W(v_i)$ = ***computation*** 비용 + ***I/O*** 비용

$$= \sum_{k=0}^{n_i} \{Cost_{pre}(T_k) + Cost_{inter}(T_k)\} + Cost_{disk}(v_i)$$

작업 T_k의 전처리 비용 $Cost_{pre}(T_k)$은 T_k에 속한 공간 데이타들의 꼭지점의 합 P_k의 관계식으로 다음과 같이 표현된다.

• $Costpre(T_k) = O(Sort(P_k)) = O(P_k \cdot \log P_k)$

$= t_p \cdot P_k \cdot \log_2 P_k$

여기서, t_p = 전처리 단계 단위 시간

그리고 각 작업 T_k 라인 교차검사 단계 비용은 두 공간 객체의 MBR 교차 면적비 tr_k와 P_k간의 관계식으로 표현된다.

• $Costinter(T_k) = O(tr_k \cdot P_k) = t_{inter} \cdot tr_k \cdot P_k$

여기서, t_{inter} = 라인 교차 단계 단위 시간

마지막으로 ***I/O*** 비용은 v_i에 포함된 모든 공간 객체 수 m_i를 이용해서 표현한다.

• $Cost_{disk}(vi) = m_i \cdot t_{disk}$

여기서, t_{disk} = 공간 데이타의 디스크 처리 단위 시간 $W(v_i)$로 표현되는 작업량은 v_i에 속한 작업의 수 n_i 만큼의 기하학적 계산(geometric computation)을 수행하고 공간 객체 수 m_i만큼의 디스크 접근을 하게 된다. 따라서 평균 디스크 접근 시간과 m_i의 곱으로 ***I/O*** 처리에 소요되는 비용을 예상할 수 있다. 또한 작업 당 교차 연산 수행 시간은 두 공간 객체의 복잡도 P_k와 두 객체의 MBR간의 교차 정도(tr_k: 점의 분포가 균일하게 이뤄져 있다고 가정)에 따라 수행 시간에 있어 다양하기에 교차 연산에 사용되는 알고리즘을 기반으로 표현된다.

결론적으로 비용 모델은 전처리 단계의 단위 시간, 라인 교차 함수의 단위 시간, 꼭지점의 합, 교차 면적비 등을 고려한 관계식으로 다음과 같이 표현된다.

수식 1.

$$\bullet\ W(vi) = \sum_{k=0}^{n_i} \{\ P_k \cdot (t_p \cdot \log_2 P_k + t_{inter} \cdot tr_k)\ \} + m_i \cdot t_{disk}$$

수식 1을 통하여 작업들간의 다양한 수행 시간을 반영함으로써 할당시 효과적인 부하 균등화를 이룰 수 있다. 그러나, v_i마다 비용 모델을 적용해야 하는 오버헤드가 존재하게 되므로 이를 보상하기 위한 방법이 필요하다.

이를 위해서 두 공간 객체의 MBR간 위상관계[12]를 그림 7과 같은 모델을 기반으로 하여 교차 면적 비를 계산하기 위한 tr(topological relation) 변수에 적용한다.

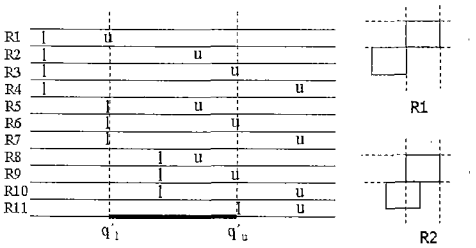


그림 8 11-level 교차 모델[13]

11-level 교차 모델은 MBR 쌍 중에서 하나의 객체를 기준(그림 8의 오른쪽 예에서 영역이 칠해진 부분)으로 기준이 되는 MBR의 x좌표(min,max)를 (q'_l, q'_u)로 두고 다른 MBR의 x좌표(min,max)를 (l, u) 라고 둘 때, 하나의 축에 대하여 후보 쌍이 가질 수 있는 모든 위상관계를 표현할 수 있다(예를 들어, R1은 기준이 되는 사각형(칠해진 영역)의 x축 좌표 (q'_l, q'_u)보다 빈 영역 사각형의 x축 좌표 (l, u)가 $u < q'_l$ 인 관계가 성립된다). 따라서 생성된 작업의 작업량 표현 시, 위상 정보를 통하여 정제 단계 없이도 공간 조건(교차 관계)을 결정할 수 있는 경우에 대하여 2차 여과 효과를 거두게 된다. 여기서 2차 여과 대상이 되는 작업들은 위상 관계 중에서 두 개의 MBR이 서로 교차된 형태(Crossed MBR이라고 정의함)로 표현될 경우, MBR속에 포함된 공간 데이타가 어떠한 모양으로 존재하든지 상관없이 무조건 교차됨을 알 수 있다.

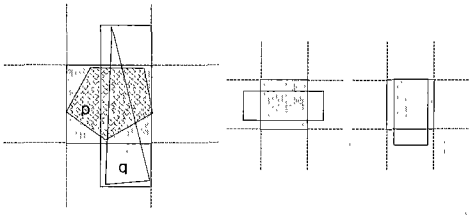


그림 9 Crossed MBR 위상 관계

예를 들어, 그림 9에서 MBR p, MBR q는 그 속에 어떤 모양의 다면체(polygon)들에 존재하여도 무조건 서로 교차하게 된다(경계가 만나는 것도 교차 관계에 포함). 이러한 MBR 위상을 가진 작업들은 실제 공간 데이타의 정제 단계 없이도 무조건 교차하게 되기 때문에 작업 생성 시 2차 여과 대상이 된다. 또한 MBR 위상 정보의 활용은 정제 단계 시 교차 연산과 포함 연산을 서로 다른 연산임에도 불구하고, 모든 작업들에 관하여 순차적으로 교차 연산 수행 후(교차 관계가 아닌 경우) 포함 연산을 수행하는 오버헤드가 존재하였다.

이를 보완하기 위하여 그림 10에서는 11-level 교차 모델[13]을 이용하여 총 121형태로 후보 쌍간의 MBR 위상관계를 나타낸다. 위상관계를 크게 3가지의 경우로 분류하면

1) 31가지 형태의 여과 대상(공간 데이타들이 무조건 교차하는 작업),
2) 18가지 형태의 포함 가능성이 존재하는 작업,
3) 교차 연산 수행만 필요한 작업

으로서 2차 여과 효과와 더불어 포함 연산을 독립적으로 수행하여 작업량 표현 시 발생하는 오버헤드를 보완할 수 있다.

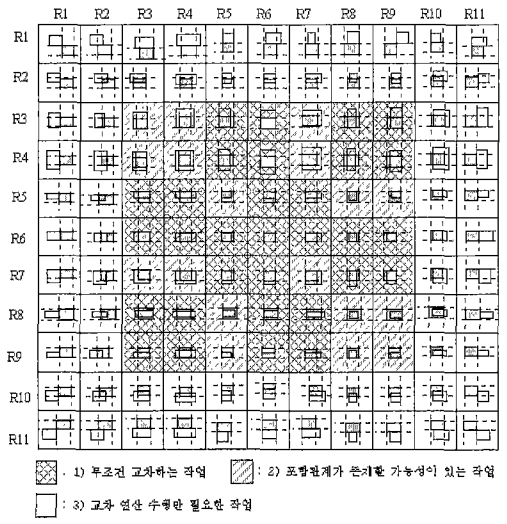


그림 10 MBR 위상 관계

3.4 그래프 분할을 이용한 작업 할당

비용 모델을 통해 표현된 작업 그래프는 병렬 수행할 프로세서 수만큼 균등 분할되어야 한다. 따라서 프로세서들 사이에 작업량의 균형을 맞추고, 통신량을 줄이기 위해 그래프 분할 방법[14]이 필요하다.

본 논문에서는 앞에서 정의한 것처럼 작업 그래프를 weighted, undirected graph 인 G = (V, E)로 표현한다(G는 self edge와 multiple edge가 없다고 생각한다). 여기서 v_i 은 작업들의 소집합으로 표현되고, $W(v_i)$는 v_i에 포함된 모든 작업들의 작업량의 합으로 나타낸다. 그리고 정점간의 간선은 통신량을 나타내는 것이 아니라, 작업들간에 공간 객체의 공유 가능성이 존재함을 나타내게 된다.

V : 정점(vertex)의 집합 (V ={v_1, v_2,...,v_Z },
Z : vertex수의 총합, $v_i \in V$, $1 \le i \le Z$)
E : 정점을 연결하는 간선의 집합
$W(v_i)$: v_i에 대한 음이 아닌 무게 값
W(e) : 간선에 대한 음이 아닌 무게 값

작업 그래프 G를 분할하는 것은 정점의 집합인 V를 p개의 겹치지 않는 부분으로 나누는 것으로, $V = V_1 \cup V_2 \cup \cdots \cup V_p$ 이고 $V_1 \cap V_2 \cap \cdots \cap V_p = \varnothing$으로 나타내며, 다음과 같은 조건을 만족시켜야 한다. 첫째, V_q에($1 \le q, j \le p$) 있는 정점들의 무게 값의 합이 일정하도록 해야 한다. 둘째, 다른 V_q와 V_j에 있는 정점간의 간선들이 잘려지는 W(e)값의 합을 최소화되도록 해야한다.

본 논문에서는 그래프 분할 알고리즘 중 가장 효율적이라고 알려진 multilevel k-way partition[14][15] 방법을 사용하여 작업 그래프를 분할한다.

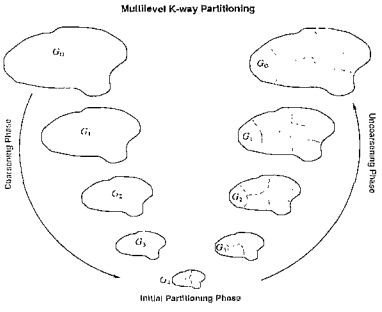


그림 11 multilevel k-way 그래프 분할 과정

그림 11은 coarsening, initial partitioning, uncoarsening (refinement)의 3가지 단계로 나뉘어 수행되는 multilevel k-way 그래프 분할 알고리즘을 보여준다.

그래프 분할의 처음 단계인 coarsening 단계 동안, 그래프 $G_i = (V_i, E_i)$, (그림 10의 경우, $0 \le i \le 4$)는 작업

그래프 $G_0 = (V_0, E_0)$으로부터 순서대로 작아지고 ($|V_{i+1}|<|V_i|$), G_i의 정점 집합은 G_{i+1}의 정점 집합으로 매칭 된다.

매칭 과정[14]은 복수의 노드들을 선택하여 하나의 노드로 합치는 것으로 이 과정에서는 여러 가지 알고리즘(Random Matching, Heavy Edge Matching 등)을 이용하게 된다. 이 결과 그래프는 복잡한 모양에서 단순한 모양으로 바뀌면서도 변형 이전의 정보를 원형 그대로 복구할 수 있도록 저장한다.

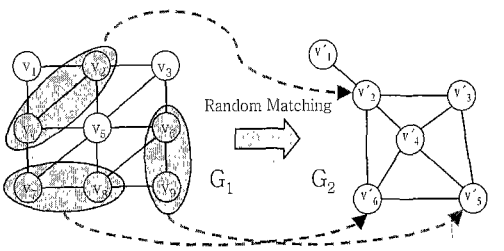


그림 12 Coarsening 단계 시 매칭 과정

그림 12은 매칭 알고리즘 중에서 Random Matching이 적용된 예로 G_1의 노드 v_2와 v_4가 G_2의 v'_2로 매칭됨을 보여준다.

두 번째 단계로 initial partitioning은 Coarsening 단계를 거쳐 최종적으로 생성된 Graph $G_i = (V_i, E_i)$ (그림 10의 경우, i=4)를 정점의 무게의 합이 일정하면서, 잘려지는 간선의 수가 최소인 k(그래프 분할 개수)개의 집합으로 분할한다. 이때 사용되는 분할 방법으로는 multilevel recursive bisection[14]을 사용한다.

마지막으로 uncoarseing 단계 동안에는 처음 단계(coarsening)의 역 단계를 수행하면서 분할 정보를 정제하는 단계를 거쳐 최종 목적인 정점 무게 합의 균형과 잘려지는 간선의 무게 값을 최소화하는 그래프 분할을 한다. 따라서 최종적으로 만들어진 그래프 분할 정보는 Master 프로세서로부터 Slave 프로세서들로의 작업 할당 시 이용된다.

4. 성능 평가

4.1 실험 환경

성능 평가에 사용된 병렬 처리 시스템은 독일 Parsytec사의 CC16시스템을 사용하였다. CC16 시스템은 16개의 프로세서들이 crossbar switch로 연결된 메시지 전달 기반의 병렬처리 시스템이며, PowerPC 604 100Mhz CPU를 사용하고 있다. 또한 switch간에는 1Gbps로 동작하는 Parsytec사의 HS-Link로 연결되어 있고, 1개의 공유 디스크와 3개의 지역 디스크로 구성되며 실험에서는 1개의 공유 디스크를 이용하였다.

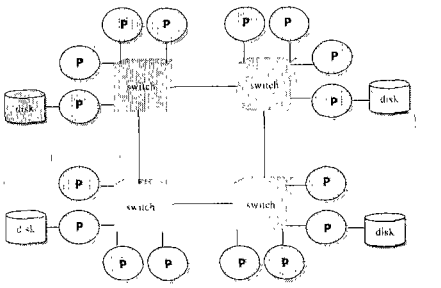


그림 13 CC16 시스템 구조

실험에 사용된 공간 데이타는 Sequioa 2000 storage benchmark[16]의 다각형 데이타를 사용하였다. 이는 미국의 지질학 측정부의 GIRAS(Geographic Information Retrieval and Analysis System) Land use / Land cover를 표현한 다각형 집합으로, 총 37개의 Land use/ Land cover 데이타와 1개의 island 영역으로 이루어져 있다. 본 실험에서는 그 중에서 도시 지역(거주 지역, 상업 지역, 산업 지역 등), 농업 지역(농경지, 과수원 등), 하천 지역을 나타내는 총 12개의 지도를 사용하였다.

실험에서 12개 지도들의 조합으로 공간 조인을 수행하였으며, 평균적으로 약 20000개의 공간 객체들로 구성되었다. 또, 공간 객체는 수 개에서 수천 개까지의 꼭지점으로 구성된 다각형 데이타를 사용하였고, R-tree를 이용하여 인덱스 구조를 만들었을 경우 평균 높이가 2가 되었다.(R-tree 노드의 최대 entry수가 44개이며, 최소 entry수가 22로 하였다. 단, 루트노드는 제외됨)

작업 할당 시 사용된 그래프 분할방법은 미국의 미네소타 대학에 개발한 소프트웨어 패키지인 Metis 버전 3.03[17]을 사용하여 작업 그래프를 분할하였다.

4.2 실험 결과

본 논문에서 수행한 실험은 그림 14과 같이 공간 영역을 나타내는 공간 데이타들에 대하여, "오염 지대에 속한 빌딩을 찾으시오." 와 같은 공간 조인 질의에 대하여 교차하는 객체의 쌍을 구하는 작업을 수행하였다.

여기서 본 논문에서 제시한 작업 생성 및 할당 방법을 적용하여 공간 조인의 병렬 수행 시에 얻은 결과를 정리하면, 다음과 같은 3가지 효과를 가진다.

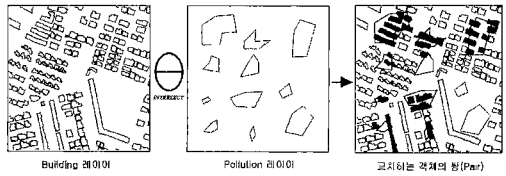


그림 14 공간 조인의 예[6]

첫째, 작업 그래프 생성 시 공간적 지역성을 고려하여 작업들을 하나의 노드로 표현하였기 때문에, 공간적으로 서로 인접한 공간 객체들이 같은 프로세서에게 할당되어 중복적인 디스크 접근의 수를 감소시키는 효과를 거둘 수 있다. 이는 프로세서마다 동일한 LRU 버퍼를 두었을 경우, 공간 지역성이 고려된 경우와 그렇지 않은 경우(생성된 작업들을 라운드 로빈 방식으로 프로세서들에게 작업을 할당하였음)의 정제 단계 수행 시간(T_a : 프로세서에게 할당된 작업들을 모두 수행하는데 소요되는 시간)을 비교하여 결과를 얻었다.

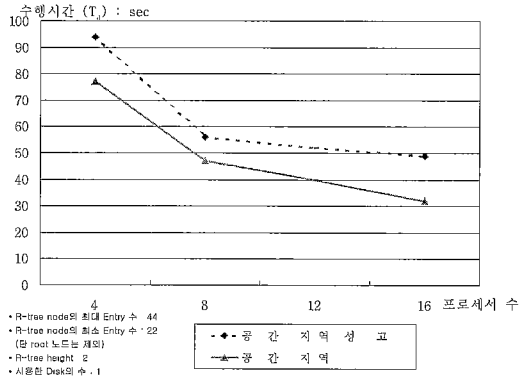


그림 15 공간 지역성 고려 결과

그림 15는 공간 지역성을 고려하여 반복적인 디스크 접근 수를 줄임으로써 얻은 결과로서 프로세서 수가 4개 일 때 약 19%, 8개 일 때 약 21%, 16개 일 때 약 31%의 수행 시간(T_a)의 단축을 가져왔다. 실험 결과에 따르면 T_a의 단축은 프로세서 수에 비례하는데, 그 이유는 공간 지역성이 고려되지 않는 경우에는 공간 객체를 공유하는 작업들이 서로 다른 프로세서들에게 할당됨에 따라 동일한 객체에 대한 디스크 접근이 반복적으로 일어나게 되어 최악의 경우, 동일한 공간 객체에 대하여 프로세서 수만큼의 디스크 접근이 일어나기 때문이다.

둘째, 비용 모델 적용 시 MBR 위상정보를 기반으로 무조건 교차하는 작업들을 여 과하는 효과와 함께 교차 관계, 포함 관계 연산을 분리하여, 순차적(교차 관계를 검사한 뒤 포함 관계를 검사)수행 시 발생하는 오버헤드를 줄였다.

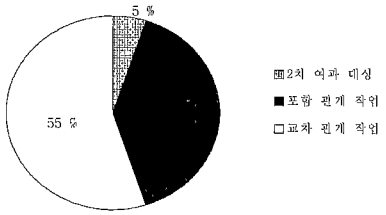


그림 16 MBR 위상 관계를 이용한 작업 분류

그림 16은 작업 생성 단계에서 후보 쌍(작업)들을 MBR 위상관계 분류했을 경우, 각각이 차지하는 비율을 나타낸 것으로 일반적으로 5%의 작업을 여과하는 효과(2차 여과)를 거둘 수 있다. 이는 정제 단계 시 수행 작업의 수를 줄임으로써 성능 향상을 이루었다

셋째, 비용 모델을 통하여 작업간의 다양한 수행 시간을 작업량 표현에 반영함으로써, 작업 할당(정적 할당에 한함)후 프로세서들간의 부하 균등화를 이루었다.

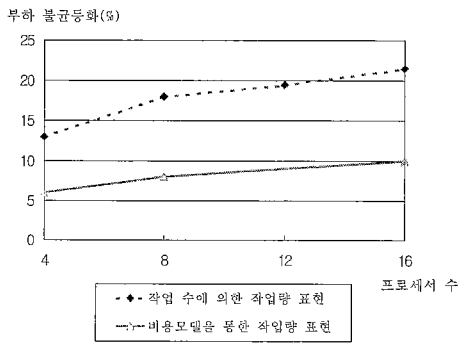


그림 17 부하 균등화 비교(정적 할당에 한함)

그림 17에서 y축 값은 프로세서들간 T_a(정제 단계 수행시간)의 표준 편차를 평균 T_a와의 백분율로서 부하 불균등화의 정도를 나타내며 프로세서 수에 비례하였다. 그 이유는 프로세서 수가 증가함에 따라 생기는 디스크 병목현상 및 통신량 증가 등의 오버헤드에 의한 T_a의 편차가 증가하였기 때문이다. 실험결과에 의하면 비용 모델을 적용했을 경우, 프로세서 수가 4개 일 때 약

13%에서 약 6%, 8개일 때 약 17%에서 약 8%, 16개일 때 약 21%에서 약 10%로 프로세서간의 T_a의 표준 편차를 줄임으로써 정적 부하 균등화를 이루었다.

결론적으로 본 논문에서 제시한 정적 부하 균등화를 위한 작업 생성 및 할당 방법을 적용함으로써 3가지 기대 효과(디스크 접근 회수 감소, 작업 여과, 부하 균등화)를 거두었다. 이를 위해서 본 논문에서 제시한 방법을 적용했을 경우와 미 적용한 2가지 경우(중간 인덱스 구조에서 작업 생성, 단말노드에서 작업 생성)의 전체 수행시간을 비교하였다.

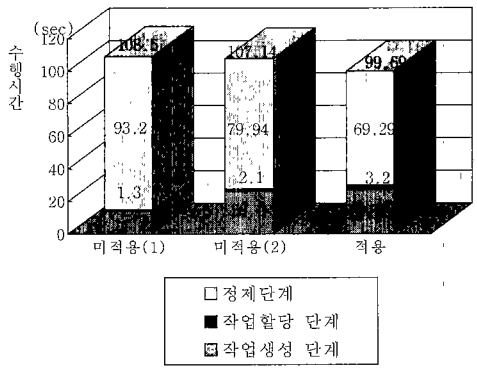
미 적용 방법의 첫 번째 방법은 일반적으로 사용되는 작업 생성 및 할당 방법으로 중간 노드 MBR이 교차하는 것을 하나의 작업으로 생성하여 작업 큐에 저장하고 이를 프로세서 수로 나누어 할당하였다. 여기서 적용된 공간 지역성 고려 및 부하 균등화 방법은 중간 노드를 공유하는 작업들이 서로 인접하여 작업 큐에 저장되므로 전체의 작업을 프로세서 수로 나누어 할당함으로써 노드를 공유하는 작업들이 같은 프로세서에 할당될 확률을 높였고 모든 프로세서들이 동일한 수의 작업을 수행하도록 하였다. 미적용한 경우의 두 번째 방법은 단말 노드에서 공간지역성을 고려하여 작업을 생성하여 동일한 작업 수를 프로세서에게 할당하였다.

그림 18은 본 논문에서 제시한 방법을 적용했을 때와 미적용한 두가지 경우의 전체 수행시간을 비교한 결과(프로세서 수를 4, 8, 16개에 적용함)이다. 프로세서 수가 4개인 경우(a)에는 작업 생성과 할당 단계에서 작업 그래프 생성 및 분할 단계가 추가됨으로써 미적용(1)의 경우보다 10.7초가 더 소요되었다.

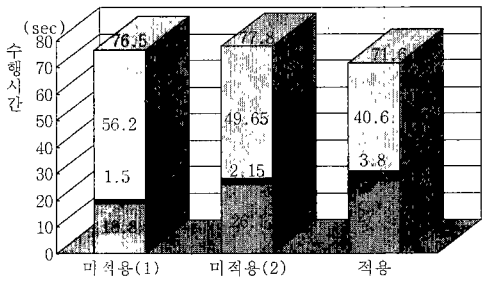
그러나 정제 단계에서 작업 그래프 표현으로 보다 효율적인 공간 지역성을 고려하여 디스크 접근 수를 줄임으로써 1.89초, 2차 여과효과로 정제 단계의 작업 수를 5.12%로 줄임으로써 3.9초, 정적 부하 균등화 정책을 통하여 프로세서간 수행 시간의 편차를 18%에서 6%로 줄임으로써 총 11.6초의 수행 시간 단축(미적용(2)경우에는 10.65초)을 가져왔다. 따라서, 본 논문에서 제시한 방법 적용을 했을 경우 전체 수행 시간에 있어 약 8%의 성능 향상을 거두었다.

프로세서 수가 8개인 경우(b)에는 전체 수행시간이 약 7초 단축되어 8%의 성능 향상을 거두었고, 프로세서 수가 16개인 경우(c)에는 전체 수행시간이 5.5초 단축되어 약 8.3%의 성능 향상을 이루었다.

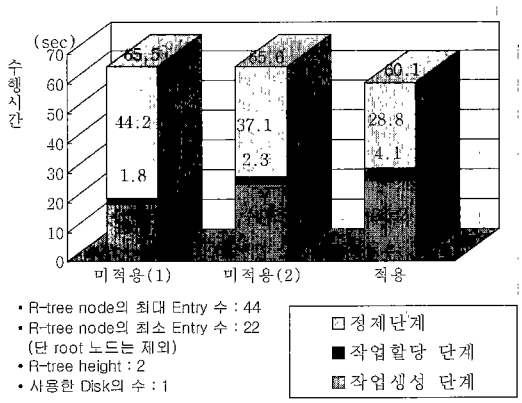
결론적으로 본 논문에서 제시한 방법을 적용하여 공간 조인을 병렬 수행했을 때 기존의 정적 할당 방법에 비해 약 8%정도의 성능 향상을 이루었다.



(a) 프로세서 수 : 4개



(b) 프로세서 수 : 8개



(c) 프로세서 수 : 16개

그림 18 전체 수행 시간 비교

4.3 단계별 수행 시간 분석

본 논문에서는 공간 조인을 작업 생성, 작업 할당과 정제 단계를 통해 수행하며 각 단계별 수행시간의 분석을 위하여 다양한 프로세서 수(1, 4, 8, 16개)에 적용하였다.

그림 19에서 정제 단계는 실제의 공간 데이타를 이용하여 복잡한 기하학적 계산(교차 연산)을 하므로 수행 단계 중 가장 많은 시간이 소요된다. 따라서 병렬 수행을 통하여 직렬 수행 시 전체의 91.5%를 차지하는 비율을 병렬 수행을 통하여 48.2%까지 낮출 수 있었다.

병렬 수행 시 존재하는 작업 할당 단계는 전체 수행 시간의 6.65%에 해당하며 그래프 분할하는 시간(2.5%)과 그래프 분할 정보를 Slave 프로세서들에게 전달하는 시간(4.15%)으로 구성된다.

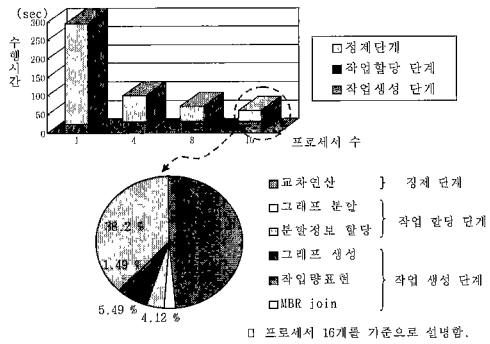


그림 19 단계별 수행 시간

작업 생성 단계 시 소요되는 시간은 전체 수행 시간의 45.2%에 해당하며 MBR join, 작업 그래프 생성과 비용 모델을 이용하여 작업량을 표현하는 시간으로 구성된다. 작업 그래프 생성은 전체 수행 시간의 5.49%에 해당하고, 비용 모델 적용을 적용하여 작업량을 표현하는 것은 1.49%에 해당되었다.

MBR join 단계는 1차 여과단계로서 공간 조인에 참여하는 공간 객체의 수와 밀접한 관계를 가지며 작업 생성 단계 중 가장 많은 시간이 소요된다. 또한, 병렬 수행되는 정제 단계와는 달리 단일 프로세서(Master)에

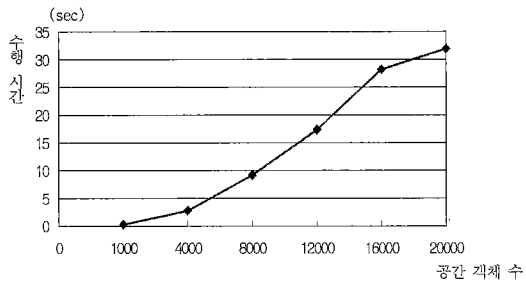


그림 20 공간 객체의 수와 MBR join 수행 시간과의 관계

의해 직렬 수행된다.

따라서, 그림 20에서와 같이 공간 조인에 참여하는 공간 객체의 수에 비례하여 MBR join 단계의 수행 시간이 증가하기 때문에 병렬화를 통한 성능 향상이 필요하게 된다.

5. 결론 및 향후 연구 과제

본 논문에서는 공간 조인의 병렬화 시 정적 부하 균등화를 위한 작업 생성 및 할당 방법을 제시하였다. 물론 병렬 수행 시 문제가 되는 공유 디스크의 병목 현상과 정확한 작업량의 표현이 불가능하기 때문에 생기는 부하 불균등의 문제가 존재하지만, 정적 할당으로서는 많은 성능 향상을 가져왔다. 그리고 정적 부하 균등화 이외에도 공간 지역성 및 2차 여과 효과 등을 통해 비용 모델 적용으로 인한 오버헤드를 제거하였다. 앞으로 작업 생성 및 할당 방법에 동적 할당 정책을 보완 시 더 좋은 결과를 가질 수 있는 밑바탕이 된다.

향후 연구 과제로는 병렬화 시점을 MBR join 단계부터 적용한 후에 정적 할당을 기반으로 하여, 실행 시 발생하는 부하 불균형을 최소화하기 위한 동적 할당에 대한 연구가 필요하다.

참 고 문 헌

[1] Guting, R. H., "An Introduction to Spatial Database Systems," VLDB Journal, No. 3, pp.357-399, 1994.

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

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

[4] Huang, Y. W. and Schneider, R., "Spatial Joins Using R-trees," Proc. ACM SIGMOD Int. Conf., pp.237-246, 1993.

[5] Brinkhoff, T., Kriegel, H. P. and Seeger, B., "Parallel Processing of Spatial Joins Using R-trees," Proc, 12th IEEE Data Engineering, pp.258-265, 1996.

[6] 김진덕, 홍봉희, 정상화, "Parallel Spatial Join using Grid Files," 한국정보과학회 춘계학술발표회 Vol. 25, No.1, pp.73-76, 1997.

[7] 서영덕, 홍봉희, "Task Creation and Assignment Algorithms for Parallel Spatial Join based on R-tree," 한국정보과학회 춘계학술발표회 Vol. 25,

No.1, pp.47-49, 1998.
[8] Graefe, G., "Query Evaluation Techniques for Large Databases," ACM Computing Surveys, Vol. 25, No. 2, pp.73-170, 1996.
[9] Beckmann, N. and Krigel, H. P., "R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD Int. Conf., pp.322-331, 1990.
[10] Guttmann, A., "R-tree:A Dynamic index structure for spatial Searching," Proc. ACM SIGMOD Conf. Boston, pp.47-57, 1984.
[11] Preparata, F. P. and Shamos, M. I., "Computational Geometry," Springer, 1985.
[12] Papadias, D. and Sellis, T., "Topological Relations in the World of Mininum Bounding Rectangles: A Study with R-trees," Proc. ACM SIGMOD'95, CA USA.
[13] Egenhofer, J., "Topological Relations between regions in IR2 and Z2**," Springer Verlog, pp. 316-336, 1993.
[14] Karypis, G. and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," Tech. Rep., CS-Dept., Univ. Minnesota, 1996.
[15] Karypis, G. and Kumar, V., "Multilevel k-way Partitioning Scheme for Irregular Graphs," Tech. Rep., CS-Dept., Univ. Minnesota, 1995.
[16] "Sequoia 2000 Benchmark Polygon Data FTP Server Site.," http://s2k-ftp.cs.berkeley.edu:8000/sequoia/benchmark/polygon
[17] Karypis, G. and Kumar, V., "Metis : A Software Package for Partitioning Unstructured Graph, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Martrices," CS-Dept., Univ. Minnesota, Army HPC Research Center, 1997.
[18] 김진덕, 홍봉희, "단일/다중할당 공간 색인에서 병렬 공간조인의 성능평가", 한국정보과학회 논문집(B), Vol. 26, No. 6, pp.763-779, 1999.
[19] 서영덕, 김진덕, 홍봉희, "병렬 공간조인을 위한 객체 캐쉬기반 태스크 생성 및 할당", 한국정보과학회 가을 학술발표 논문집, Vol. 26, No. 10, pp.1178-1192, 1999.

박 윤 필
1999년 2월 부산대학교 컴퓨터공학과(학사). 2001년 2월 부산대학교 컴퓨터공학과(석사). 2001년 1월 ~ 현재 한국증권전산(주). 관심분야는 컴포넌트 기반 소프트웨어 개발, 컴포넌트 저장소, 웹 기반 소프트웨어 개발, 객체 지향 소프트웨어, 작업 할당, GIS 등임.

염 근 혁
1985년 2월 서울대학교 계산통계학과(학사). 1992년 8월 Univ. of Florida 전산학과(석사). 1995년 8월 Univ. of Florida 전산학과(박사). 1985년 1월 ~ 1988년 2월 금성반도체 컴퓨터연구실 연구원. 1988년 3월 ~ 1990년 6월 금성사 정보기기연구소 주임연구원. 1995년 9월 ~ 1996년 8월 삼성SDS 정보기술연구소 책임연구원. 1996년 8월 ~ 현재 부산대학교 컴퓨터공학과 조교수. 부산대학교 컴퓨터 및 정보통신 연구소 연구원. 관심분야는 컴포넌트 기반 소프트웨어 개발, 도메인 공학, 소프트웨어 아키텍쳐, 객체지향 소프트웨어 개발방법론, 요구 검증, 분산 컴포넌트, 소프트웨어 재사용 등임.