병렬 공간 조인을 위한 준동적 태스크 할당

김진덕[+], 서영덕[++], 홍봉희[++]
[+]동의대학교 컴퓨터공학과, [++]부산대학교 컴퓨터공학과
jdk@dongeui.ac.kr,{ydseo,bhhong}@pusan.ac.kr

Semi-dynamic Task Allocation for Parallel Spatial Joins

Jin-Deog Kim[+], Young-Duk Seo[++], Bong-Hee Hong[++]
[+] Dept. of Computer Engineering, Dongeui University
[++] Dept. of Computer Engineering, Pusan National University

요 약

최근 병렬 시스템을 이용하여 공간 조인의 성능 방안에 연구가 진행되고 있다. 그렇지만 프로세서의 수가 증가할수록 병렬 처리에 의한 프로세서의 효율성은 급격히 떨어진다. 이것은 병렬 공간 조인을 수행할 경우 순차 공간 조인 보다 디스크 병목 현상과 메시지 전송 오버헤드가 심하게 발생하기 때문이다. 이 논문에서는 공유 디스크 구조에서 다중 프로세서의 디스크 동시 접근으로 인한 병목 현상을 완화하고, 메시지 전송을 최소화하기 위한 태스크 할당 기법을 제안하였다. 제안한 태스크 할당 기법을 두 가지 공간 조인 방법에 각각 적용하여 디스크 접근 횟수와 메시지 전송 횟수의 감소 효과를 실험으로 평가하였다. MIMD 구조 및 공유디스크 방식의 병렬 시스템에서의 다양한 실험에서 이 논문에서 제안한 준동적 태스크 할당 기법이 정적 할당과 동적 할당 기법에 비해 우수함을 보였다.

Abstract

It is usually appropriate to use parallel processing to improve the performance of spatial join processing. However, as the number of processors increases, the efficiency of each processor decreases rapidly because of the disk bottleneck and the overhead of message passing. This paper proposes the method of task allocation to soften the disk bottleneck caused by accessing the shared disk at the same time, and to minimize message passing among processors. In order to evaluate the performance of the proposed method in terms of the number of disk accesses and message passing, we conduct experiments on the two kinds of parallel spatial join algorithms. The experimental tests on the MIMD parallel machine with shared disks show that the proposed semi-dynamic task allocation method outperforms the static and dynamic task allocation methods.

1. 서 론

공간조인은 단일주사(single scan) 질의인 점 질의(point query)나 영역 질의(region query)와는 달리 다중주사(multiple scan) 질의이기 때문에 객체의 수가 증가함에 따라 연산 시간이 급격히 증가한다[1,4,7]. 그러므로 빠른 질의 처리를 요구하는 응용에서 공간 조인의 효율적 처리방안이 필요하다. 그래서 공간 색인을 이용한 공간 조인과 병렬 처리에 관한 연구[1,3,7,9]에 많은 관심이 집중되고 있다. 그러나 병렬 시스템을 이용한 공간 조인에서 CPU 연산시간의 단축효과는 뛰어나지만 동시 다발적인 디스크 접근으로 인한 디스크 병목 현상과 프로세서 수의 증가에 따른 과도한 메시지 전송 횟수는 전체적인 병렬 공간 조인의 성능 향상에 있어서 걸림돌로 작용한다[9].

공간 조인을 위한 공간 색인 기법은 크게 단일할당 공간 색인과 다중할당 공간 색인과 같은 두 가지로 분류된다[5,7]. 단일 할당은 주로 R-tree와 같은 객체 중심의 공간 색인이며, 다중할당은 Quad tree와 같은 영역 중심 정규분할 공간색인[3]에 주로 적용된다. 단일할당 공간 색인을 이용한 공간 조인은 단일 할당, 다중 조인(Single Assign, Multiple Join : SAMJ)[5,7]의 특징을 가지고 있다. 다중 할당 공간 색인을 이용한 공간 조인은 다중 할당, 단일 조인(Multiple Assign, Single Join : MASJ)[5,7]의 특징을 가지고 있다.

그래서 이 논문에서는 두 가지 대표적인 공간 조인 기법의 단점을 동시에 해결할 수 있는 태스크 할당 기법을 제시하고자 한다. 이 논문의 구성은 다음과 같다. 2장에서는 공간 조인의 정의와 특성 및 각 색인에 따른 공간 조인 기법을 제안하고 설명한다. 3장에서는 태스크의 할당 기법과 부하 평준화에 대해 설명한다. 4장에서는 다양한 실험평가의 결과를 살펴보고 5장에서 결론을 맺는다.

2. 병렬 공간 조인 방법

기존 관계형 데이터베이스 시스템에서 적용되는 병렬 조인 기법을 공간 데이터베이스 시스템에는 그대로 적용하기가 어렵다[7]. 그러므로 병렬 처리 공간 조인 연산을 수행하기 위해서는 전술한 SAMJ 또는 MASJ 방법이 사용되어야 한다. 이 논문에서 태스크 할당 기법을 적용할 병렬 공간 조인을 위한 공간 색인으로서 자료 관리의 편의성 및 타일 구조에 적합한 고정 그리드를 기반으로 하였다. 각 조인 방법을 간단히 PSJ_MA, PSJ_SA로 명명한다.

2.1 PSJ_MA

다중할당 고정 그리드를 공간 색인으로 이용하는 PSJ_MA는 다음과 같은 특성이 있다.

- **다중 할당** : 먼저, 데이터 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.
- **단일 조인** : PSJ_MA에서 하나의 태스크는 같은 위치의 셀 쌍의 공간 조인 작업을 의미한다. 전체 공간조인(R ▷◁ S)의 결과는 ∪(Ri ▷◁ Si)이다. 병렬 수행을 위해 각 태스크가 하나의 프로세서에 할당된다. 프로세서 개수가 P로 유한하다면 평균 n/P개의 태스크를 각 프로세서가 수행해야 하고, 적절한 태스크 할당 기법이 요구된다.

PSJ_MA는 단일 조인이므로 여과 단계에서 각 그리드 셀은 오직 한 번씩만 검색하면 된다. 따라서 공간 색인의 검색 범위가 좁혀지는 장점이 있다. 그러나 이 방식의 단점은 데이터를 중복해서 참조하기 때문에 조인의 결과가 중복된다[7]는 것이다. 이를 위한 태스크 할당기

법은 3장에서 자세히 다룬다.

PSJ_MA는 여과 단계를 수행한 후 정제단계의 비용을 줄이기 위해 생성된 후보 객체쌍의 중복을 정제 단계를 수행하기 전에 제거한다.

PSJ_MA는 병렬 여과 및 병렬 정제의 2단계로 수행된다. 각 단계는 또한 태스크 생성, 태스크 수행, 결과 취합의 3단계로 세분화 된다. 여과 단계에서는 색인 파일만을 읽고, 정제 단계에서는 원시 데이터 파일만을 읽는다. PSJ_MA의 여과 단계에서 디스크 접근 횟수를 줄이기 위해 버퍼를 사용한다. 버퍼의 한 슬롯(*slot*)의 크기는 1Kbytes이고 전체 버퍼 크기는 다양하게 적용한다.

2.2 PSJ_SA

단일 할당 고정 그리드를 공간 색인으로 이용하는 PSJ_SA는 다음과 같은 특성이 있다.

• **단일 할당** : 먼저, 다중 할당 고정 그리드처럼 데이터 집합 공간을 일정한 면적을 갖도록 X,Y 좌표축으로 등분하며 최초의 각 그리드 셀간에는 겹침이 존재하지 않는다. 그 뒤, 각 객체는 오직 하나의 그리드 셀(객체의 MBR의 중심점을 포함하는 셀)에만 포함된다. 모든 객체가 각 그리드 셀에 할당된 후 각 셀의 범위는 포함된 객체들을 모두 감싸는 최소 경계사각형으로 재조정된다. 이 논문에서는 재조정된 범위를 그리드 셀의 확장 MBR(*EMBR* : Extended MBR)이라 한다.

• **다중 조인** : 단일 할당 고정 그리드의 각 셀은 불규칙한 범위를 가지므로, 하나의 셀 R_i의 EMBR은 다수개의 셀 S_j의 EMBR과 겹칠 수 있다. 그러므로 PSJ_SA는 하나의 셀 R_i에 대해 다수개의 셀 S_j와 다중 조인을 수행해야 한다. 그러므로 PSJ_SA의 한 태스크(R_i)의 실행 결과는 $\sum_{j=1}^{\beta}(R_i \triangleright\triangleleft S_j)$이다. 이 때 β는 대응 그리드의 개수이고, S_j는 R_i의 대응 그리드들이다. 이 논문에서는 한 셀 R_i의 EMBR과 겹치는 EMBR을 가진 모든 S의 셀 들을 대응 그리드(*corresponding grid*)라 한다.

3. 태스크 할당

이 논문에서 태스크는 하나의 프로세서가 중단됨이 없이 한번에 수행할 수 있는 작업을 말한다. 그리고 각 프로세서는 여러 개의 태스크를 수행할 수 있다. 그러므로 프로세서간의 부하 평준화 작업이 이루어져야 한다.

3.1 정적 태스크 할당

부하 평준화를 위해 각 그리드 셀의 작업량을 미리 추정하여 태스크를 균등하게 분배한 뒤 실행 전에 한꺼번에 각 프로세서에 태스크들을 할당하는 정적 태스크 할당 기법을 고려해 볼 수 있다. 그림 1은 태스크가 11개이고 프로세서가 4개일 때의 정적 태스크 할당기법을 보여주고 있다. 구체적인 수행 단계는 다음과 같다.

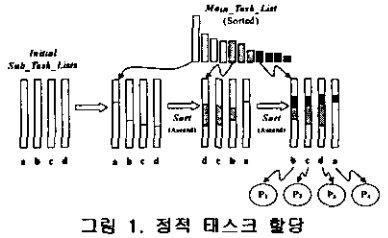


그림 1. 정적 태스크 할당

1) 프로세서는 미리 각 태스크들의 부하 크기를 추정하고, 크기별로 내림차순하여 리스트(*main task list*)로 구성한다. 각 태스크의 부하 크기는 색인 파일의 객체정보로서 추정이 가능하다.

2) Master 프로세서는 *main task list*로부터 태스크를 순서대로 각 *sub task list*에 삽입한다. 그림 1의 '*a*', '*b*', '*c*', 그리고 '*d*'가 각각 *sub task list*들이다. *sub task list*의 개수는 프로세서의 개수와 동일하다. 그림 1의 3번째 단계처럼 만약 한 *sub task list*의 삽입전의 부하 크기 합이 다른 *sub task list*의 삽입후의 부하 크기 합보다 크다면 삽입을 생략한다.

3) 각 *sub task lists* 들을 부하 크기 합에 따라 오름차순으로 정렬한다.

4) *main task list*의 모든 task들이 삽입되었다면, Master 프로세서는 각 *sub task list*를 Slave 프로세서에게 하나씩 할당한다. 그렇지 않다면, Master 프로세서는 단계 2부터 다시 진행한다.

비록 정적 태스크 할당 기법은 각 프로세서에 태스크의 부하 크기가 공평하게 분배되었다 할지라도, 각 프로세서에서의 수행 종료 시간은 많은 차이가 있을 것이다.

3.2 동적 태스크 할당

정적 태스크 할당 기법의 문제를 해결하기 위해 동적 태스크 할당 기법을 고려해 볼 수 있다. 그래서 이 논문에서는 태스크의 크기를 고려하여 동적으로 부하 평준화를 이루는 동적 태스크 할당 기법을 도입하였다. 구체적인 수행 단계는 다음과 같다.

1) 정적 태스크 할당 기법의 단계 1과 동일

2) 최초 Master는 각 Slave 프로세서에게 *task list*의 태스크를 순서대로 하나씩 할당한다. Slave 프로세서가 주어진 태스크를 종료함과 동시에 Master 프로세서에게 새로운 태스크를 요구한다. 요구를 받은 Master 프로세서는 *task list*의 다음 task를 할당한다.

3.3 준동적 태스크 할당

전술한 정적, 동적 태스크 할당 기법은 단지 태스크의 크기만을 고려하였다. 다시 말해 각 태스크의 지역성을 전혀 반영하지 못해 인접한 두개의 태스크가 서로 다른 프로세서에 할당되어 수행될 가능성이 매우 높다. 그러므로 지역적으로 인접한 두개 이상의 태스크는 가능한 하나의 프로세서에서 수행되는 것이 바람직하다. 그래서 이 논문에서는 태스크의 크기와 인접성을 고려한 태스크 할당 기법을 새로 제시한다.

전체 태스크를 크게 두 부분으로 나누어 한 부분을 먼저 정적으로 인접성을 고려하여 태스크를 할당한다. 그 뒤 나머지 부분은 동적인 부하 평준화 기법으로 크기를 고려하여 태스크를 할당한다. 이를 이 논문에서는 준동적 태스크 할당 기법이라 정의한다.

준동적 태스크 할당 기법은 인접성을 고려한 정적 태스크 할당으로 중복 디스크 검색 및 과도한 메시지 전송 등을 막을 수 있고, 동적 태스크 할당으로 부하 평준화가 용이해 진다는 점을 이용하는 것이다.

정적으로 할당된 태스크 들의 수행 완료 시간은 전술한 바와 같이 연산의 복잡도, 프로세서의 수행능력에 따라 일정하지 않다. 이를 보완하기 위해 그림 2의 오른쪽 하단에 있는 전체 태스크 중 1/4을 동적 부하 평준화 기법으로 태스크를 할당하게 된다.

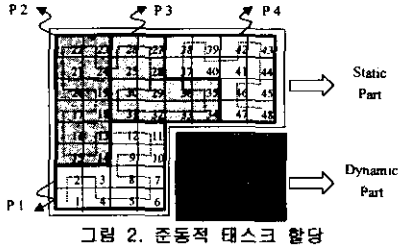


그림 2. 준동적 태스크 할당

4. 실험 평가

이 논문에서는 IBM SP2를 이용하여 제안한 병렬 공간 조인 방법에 따른 태스크 할당 기법의 성능을 평가한다. 프로세서의 수는 16개까지, 그리드 해상도는 16부터 256까지, 버퍼의 크기는 2Kbytes부터 256Kbytes까지 다양하게 실험하였다. 이 논문에서 사용하는 실험 데이터는 Sequoia 2000 Benchmark 데이터[8]로서 실제 데이터(real data)이고 불균일 분포 상태의 다각형 정보이다. 각 다각형을 구성하는 점들의 좌표 값까지 가지고 있다. 표 1에 실험 데이터의 특성을 정리하였다.

표 1. Sequoia 데이터의 특성

	#of obj.(R)	#of obj.(S)	Density(R)	Density(S)
Sequoia	41814	37793	0.39	0.42

4.1 페이지 단위 디스크 검색 횟수

일반적으로 하나의 태스크를 구성하는 R_i와 S_j 셀들은 한 개 이상의 페이지를 갖는다. 공간 조인이 다중 주사 질의 이므로 하나의 페이지를 여러 번 읽어야 한다. 그러므로 버퍼가 있다면 디스크 접근 횟수를

줄일 수 있다. 이와 같이 하나의 태스크를 수행할 때 발생하는 버퍼 효과를 이 논문에서는 **지역 버퍼 효과**(local buffer effect)라 정의한다. 태스크 할당 기법에 관계없이 지역 버퍼 효과를 얻을 수 있다.

PSJ_SA인 경우 대응 그리드가 여러 개 존재하고, 이웃한 두 태스크의 대응 그리드들간에는 공통되는 그리드 셀이 존재한다. 그러므로 이전에 수행했던 태스크들에 의해 버퍼 hit되는 경우가 발생하고, 이로 인해 디스크 접근 비용의 감소로 이어진다. 이와 같이 태스크들간의 지역성에 의해 발생하는 버퍼 효과를 **전역 버퍼 효과**(global buffer effect)라 정의한다.

그림 3은 PSJ_MA의 디스크 접근 횟수를 버퍼별로 그래프화 한 것이다. 그림 3은 다음과 같이 요약할 수 있다.

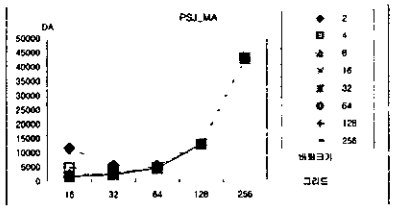


그림 3. PSJ_MA의 디스크 접근 횟수

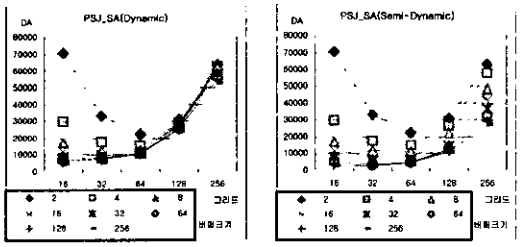
우선, 태스크 할당 기법에 관계 없이 디스크 검색 횟수는 일치한다. 즉, 태스크의 지역성을 반영하는 준동적 태스크 할당 기법이라 할 지라도 전역 버퍼 효과가 전혀 없음을 보여준다.

둘째, 그리드 해상도가 낮을 경우 지역 버퍼 효과에 의해 버퍼의 크기가 증가함에 따라 디스크 접근 횟수가 상당히 감소함을 보여주고 있다. 반면, 그리드 해상도가 256일 경우 전역 버퍼 효과만을 기대할 수 있으므로 상대적으로 버퍼의 효과가 크지 않았다. 이것은 그리드 해상도가 낮을 때 하나의 그리드 셀은 여러 개의 디스크 페이지를 차지하기 때문이다.

그림 4는 PSJ_SA에 각각 동적 태스크 할당 기법과 준동적 태스크 할당 기법을 적용했을 경우의 디스크 접근 횟수를 버퍼별로 그래프화 한 것이다. 그림 4는 다음과 같이 요약할 수 있다.

우선, 그림 4의 a), b) 그래프 모두 버퍼의 크기가 클수록 디스크 접근 횟수가 상당히 감소함을 알 수 있다.

둘째, 동적 태스크 할당 기법에 비해 이 논문에서 제안한 준동적 태스크 할당 기법을 적용하면 전역버퍼효과에 의해 상당한 디스크 접근 횟수의 감축 효과가 있다.



(a) 동적 태스크 할당 (b) 준동적 태스크 할당

그림 4. PSJ_SA의 디스크 접근 횟수

그림 3과 그림 4를 비교하면, 전반적으로 PSJ_MA의 디스크 접근 횟수가 PSJ_SA의 그것보다 적지만 그리드 해상도가 매우 높고, 버퍼의 크기가 매우 클 때 준동적 태스크 할당 기법을 적용하면 PSJ_SA의 디스크 접근 횟수가 오히려 적다.

4.2 후보 객체쌍의 중복제거를 위한 메시지 전송 횟수

후보 객체쌍의 중복 제거 과정은 PSJ_MA에서만 존재한다. 그림 5는 중복 제거를 위한 메시지 전송횟수를 프로세서 개수와 그리드 해상도를 기준으로 도식화한 것이다. 그림 5는 다음과 같은 두가지 사항으로 요약된다.

첫째, 이 논문에서 제안한 준동적 태스크 할당 기법은 그리드 해상도에 관계없이 메시지 전송횟수가 거의 일정하며, 그리드 해상도가 높을 경우 동적 태스크 할당 기법에 비해 메시지 전송 횟수가 현격히 줄어듦을 알 수 있다. 이것은 준동적 태스크 할당 기법을 적용하면 중복되는 후보 객체쌍이 프로세서 내에서 거의 제거되기 때문이다.

둘째, 프로세서의 개수가 증가할수록 메시지 전송량은 증가한다. 이것은 프로세서 개수가 증가할수록 중복되는 후보 객체쌍이 분산될 확률이 높아지고, 또한 Sort Merge 방법의 단계가 증가하기 때문이다.

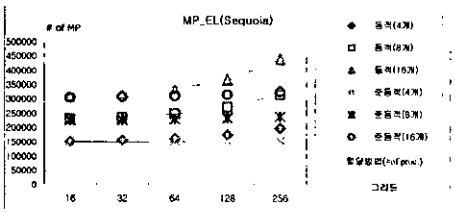


그림 5. 후보 객체쌍의 중복 제거를 위한 메시지 전송 횟수

5. 결론 및 향후 연구 과제

병렬 시스템을 이용한 공간 조인에서 CPU 연산시간의 단축효과는 뛰어나지만 디스크 병목 현상과 프로세서 수의 증가에 따른 과도한 메시지 전송은 성능 향상의 저해 요소이다. 이 논문에서는 대표적인 두 가지 공간 조인 기법(MASJ, SAMJ)을 병렬 시스템을 이용하여 처리하고자 할 때 발생하는 문제점을 해결하는 태스크 할당 기법을 제시하였다. 이 논문에서 새롭게 제시한 준동적 태스크 할당 기법은 공간 데이터의 지역성과 동적 부하 평준화를 동시에 고려하여 태스크 중 일부는 서로 인접한 태스크를 그룹으로 묶어 정적(static)으로 할당하고, 나머지 태스크는 태스크의 크기 순으로 동적(dynamic)으로 할당한다. 그리고 제안한 태스크 할당 기법을 적용하기 위한 두 가지 병렬 공간 조인 기법(PSJ_MA, PSJ_SA)은 고정 그리드를 기반으로 하였다.

실제 데이터를 이용한 실험 결과 이 논문에서 제시한 준동적 할당 기법을 적용할 경우 기존의 정적 또는 동적 태스크 할당 기법에 비해 좋은 성능을 보였다. 구체적으로 준동적 태스크 할당 기법을 적용한 PSJ_MA는 디스크 접근 횟수의 감소로 성능 향상이 있었고, PSJ_SA는 결과의 중복 제거를 위한 메시지 전송 횟수의 감소로 성능 향상이 있었다.

참고문헌

[1] 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

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

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

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

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

[6] M. Murphy, D. Rotem, "*Processor Scheduling for Multiprocessor Joins*", Proc. of Int. Conf. on Data Engineering, pp. 140-148, 1989

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

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

[9] 서영덕, 김진덕, 홍봉희, "*병렬 공간 조인을 위한 객체 캐쉬 기반 태스크 생성 및 할당*", 한국정보과학회 논문지, 26권 10호, pp 1178-1192, 1999