공간 해쉬 조인 알고리즘을 이용한 편중 데이터 처리 기법

심영복° 이종연
충북대학교 컴퓨터교육과
young°@query.chungbuk.ac.kr, jongyun@chungbuk.ac.kr

A Skewed Data Handling Method using Spatial Hash Join Algorithm

Young-Bok Shim° Jong-Yun Lee
Dept. of Computer Education, Chungbuk National University

요 약

이 논문은 인덱스가 존재하지 않는 두 입력 테이블의 공간 조인 연산 과정 중 여과 단계 처리에 중점을 둔다. 관련 연구는 Spatial Hash Join(SHJ)과 Scalable Sweeping-Based Spatial Join(SSSJ) 알고리즘이 대표적이다. 하지만 조인을 위한 입력 테이블의 객체들이 편중 분포할 경우 성능이 급격히 저하되는 문제를 가지고 있다. 따라서, 이 논문에서는 이러한 문제를 해결하기 위해 기존 SHJ 알고리즘과 SSSJ 알고리즘의 특성을 이용한 Spatial Hash Strip Join(SHSJ) 알고리즘을 제안한다. 기존 SHJ 알고리즘과의 차이점은 입력 데이터 집합을 버킷에 할당할 때 버킷 용량에 제한을 두지 않는다는 점과 버킷의 조인 단계에서 I/O 성능의 향상을 위해 우수한 SSSJ 알고리즘을 사용한다는 것이다. 끝으로 이 논문에서 제안한 SHSJ 알고리즘의 성능은 실제 Tiger/line 데이터를 이용하여 실험한 결과 기존의 SHJ와 SSSJ 알고리즘 보다 편중된 입력 테이블의 조인 연산에 대해 월등히 우수함이 검증되었다.

1. 서 론

공간 데이터베이스 관리 시스템은 GIS, CAD, 그리고 멀티미디어 정보 시스템과 같은 여러 분야에서 사용되는 대용량 데이터(예, 위성 이미지, 분자 구조)의 효율적인 관리에 중점을 두고 있다. 특히, 데이터 집합으로부터 공간 조건을 만족하는 모든 객체 쌍을 검색하는 공간 조인과 겹침(overlap) 연산이 중요한 요소 중 하나이다. 예로 "Find all forests in the United States that receive more than 20 inches of average rainfall per year."와 같은 공간 질의를 들 수 있다. 이와 같은 공간 질의의 처리를 위해 공간 조인 연산이 요구되며 이 연산은 매우 고비용의 연산이므로 공간 데이터베이스의 질의처리 연구 분야에서 핵심 요소로 취급되고 있다.

이 논문에서는 공간 조인 연산 중 공간 조인의 대상이 되는 두 입력 테이블에 대한 색인이 존재하지 않을 경우를 효율적으로 처리할 수 있는 알고리즘을 연구한다. 이러한 경우에 대한 기존 알고리즘으로는 Spatial Hash Join(SHJ)[1]과 Scalable Sweeping-based Spatial Join(SSSJ)[2] 알고리즘이 있다. 그러나 기존 알고리즘은 편중된(skewed) 공간 데이터에 대한 조인 연산 시 조인 성능이 급격히 저하되는 문제점을 가지고 있다. 특히, SHJ 알고리즘의 경우 두 입력 테이블의 데이터들이 서로 상이하게 편중되어 분포할 경우 해쉬 버킷의 빈번한 오버플로우 처리와 이로 인한 버킷에 중복 저장되는 데이터의 증가 때문에 조인 성능에 심각한 영향을 준다. 따라서, 이 논문에서는 SHJ 알고리즘의 편중 데이터에 대한 문제점을 해결할 수 있는 SHSJ(Spatial Hash Strip Join) 알고리즘을 제안한다. SHSJ 알고리즘의 기본 개념은 조인 연산 시 버킷 오버플로우를 허용하는 것을 전제로 하며, 해당 버킷의 조인 단계에서 I/O 성능이 우수한 SSSJ 알고리즘을 사용한다는 것이다.

이 논문의 구성은 다음과 같다. 제 2장에서는 공간 조인 연산을 위해 제안된 기존 알고리즘을 기술하고 제 3장에서는 편중 데이터의 처리를 위한 SHSJ 알고리즘을 제안한다. 제 4장에서는 이 논문에서 제안한 SHSJ 알고리즘과 기존 SHJ 알고리즘 및 SSSJ 알고리즘의 성능을 비교하여 분석한다. 마지막으로 제 5장에서는 본 논문의 결론 및 향후 연구과제에 대해 기술한다.

2. 관련 연구

두 입력 테이블 모두 인덱스가 존재하지 않을 경우에 대한 공간 조인 알고리즘으로는 가장 기본적인 알고리즘인 중첩 루프 조인(nested loop join)[3], Seeded-tree Join(STJ) 알고리즘[4], 해쉬 기반 조인 알고리즘[1, 5, 6], SSSJ 알고리즘 등이 있다. 이러한 알고리즘들 중 이 논문에서 제안한 SHSJ 알고리즘의 기원이 되는 SHJ와 SSSJ 알고리즘의 개념 및 특성을 간략히 기술한다.

2.1 Spatial Hash Join 알고리즘

SHJ 알고리즘의 기본 개념은 사각형으로 공간 분할된 공간 해쉬 버킷 집합을 정의하고 데이터 집합 A와 B를 생성된 버킷 사각형에 할당하고 같은 영역을 갖는 A와 B로부터의 각 버킷 쌍들은 조인 처리를 위해 디스크로부터 읽어 들여서 공간 조인 연산을 수행하는 기법이다. 해쉬 버킷의 영역은 데이터 집합 A에 의해 결정된다. 영역의 수 s는 한 버킷에 할당되는 사각형의 평균수가 메모리에 적재가능하고 영역을 나누는 동안 버퍼 쓰래싱(thrashing)을 피할 수 있게 정의된다. 데이터 집합에는 사각형 분할에 사용할 수 있는 정보가 없으므로 데이터 집합 A의 샘플에 의해 두 단계로 계산된다. 우선, s개의 사각형을 샘플로 선택하고, 선택된 사각형의 중심 값이 초기 버킷의 영역으로 정의된다. 그 후, 다른 입력 사각형들은 그들의 중심 값이 가장 근접한 영역으로 할당되며, 이때 버킷의 역경 확장이 필요할 때 영역 중심 값의 갱신이 요구된다.

해쉬 단계 동안 A로부터의 각 객체 r_A가 어떤 영역에도 포함되지 않을 경우 그 객체의 삽입을 위해 영역 확장이 최소로 요구되는 버킷에 할당된다. A로부터의 모든 객체가 할당되고 난 후 각 버킷의 영역은 변하게 되고 버킷들은 서로 겹침이 발생할 수도 있다.

그 후, 데이터 집합 B의 버킷 할당을 위해 A의 버킷 영역과 동일한 영역을 사용하여 버킷에 할당되지만 삽입과정에서 버킷의 영역은 고정적으로 유지되고 객체는 그것과 교차하는 모든 버킷에 삽입 된다 . 또한, 어떤 객체는 하나 이상의 버킷에 삽입되고(replicated), 어떤 객체는 어떠한 버킷에도 삽입되지 않는다(filtered). 데이터 집합 B에 대해 해쉬를 수행한 후, 생성된 두 해쉬 테이블에서 동일한 영역을 갖는 버킷 쌍들을 대해서

plane-sweep 알고리즘[7]을 이용하여 조인 연산을 수행한다.

2.2 Scalable Sweeping-Based Spatial Join 알고리즘

SSSJ 알고리즘은 앞에서 기술한 plane-sweep 알고리즘을 외부 조인이 가능하도록 확장한 알고리즘으로 이 논문에서 제안한 SHSJ 알고리즘의 버킷 조인 단계를 처리한다. SSSJ 알고리즘을 이용한 2-차원의 조인 처리에 대해서, 각 입력 테이블 $r \in P$ 혹은 $r \in Q$는 x-축에서 하한-경계값 r^x_{min}과 상한-경계값 r^x_{max}에 의해, 그리고 y-축에서는 하한-경계값 r^y_{min}과 상한-경계값 r^y_{max}에 의해 정의되며, 입력 테이블 P와 Q의 데이터 집합 사이의 교차 쌍을 검색하는 SSSJ 알고리즘은 그림 1과 같다.

Algorithm SSSJ :
Step 1 집합 P와 Q를 그것들의 하위 y-좌표에 대해 정렬한다.
Step 2 내부 메모리 plane-sweep 시작한다.
Step 3 **if** plane-sweep이 메인 메모리를 초과하면
plane-sweep이 메모리에 상주 가능할 수 있게 사각형 분할 조인을 사용하여 한 레벨의 분할을 수행하고 각 하위 문제에 대해 **SSSJ** 알고리즘을 재귀적으로 호출한다.

그림 1. 테이블 P와 Q의 조인을 위한 SSSJ 알고리즘

SSSJ 알고리즘의 초기 정렬 수행 후 조인 문제를 해결하기 위해 직접 plane-sweep을 시도한다. 초기 정렬 후 sweeping 구조가 메인 메모리 적재가 가능하다고 가정하고 단순한 내부 plane-sweep 알고리즘을 시작한다. plane-sweep의 수행 동안 sweeping 구조의 크기를 감시하고, 만약 사전에 정의된 한계값에 도달하면 사각형 분할 조인 알고리즘을 호출한다.

3. SHSJ 알고리즘의 설계

이 장에서는 1 장에서 언급했던 인덱스가 존재하지 않는 편중 데이터의 조인 시 기존 SHJ 알고리즘의 문제점을 개선 할 수 있는 SHSJ 알고리즘을 설계 한다.

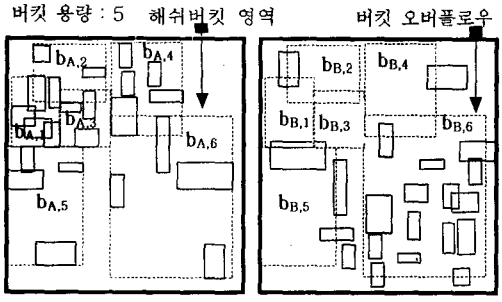

그림 2. 편중 데이터 집합에 대한 버킷 오버플로우
(a) 첫 번째 입력 데이터 집합의 버킷 할당 (b) en 번째 입력 데이터 집합의 버킷 할당

그림 2는 두 입력 데이터 집합을 SHJ 알고리즘을 사용하여 버킷에 할당한 예이다. 그림 2(b)에서 보는 것처럼 두 번째 데이터 집합의 할당 과정에서 버킷 $b_{B,6}$의 경우 버킷 용량을 초과하여 오버플로우 처리를 하여야 한다. 따라서 버킷 오버플로우 처리를 위한 오버헤드 및 버킷 분할에 따른 버킷 개수와 겹침의 증가로 인해 중복 저장되는 데이터의 수가 급격히 증가한다. 이러한 문제점을 개선하기 위해 SHSJ 알고리즘에서는 버킷 분할 단계에서 버킷의 오버플로우를 허용하고, 버킷의 조인 단계에서 I/O 성능이 우수한 SSSJ 알고리즘을 사용한다. 그림 3(a)는 그림 2(b)에서 오버플로우가 발생한 버킷 $b_{B,6}$를 SHJ 알고리즘을 사용하여 처리한 결과이며 그림 3(b)는 SHSJ 알고리즘을 사용하여 생성된 해쉬 테이블의 결과이다.

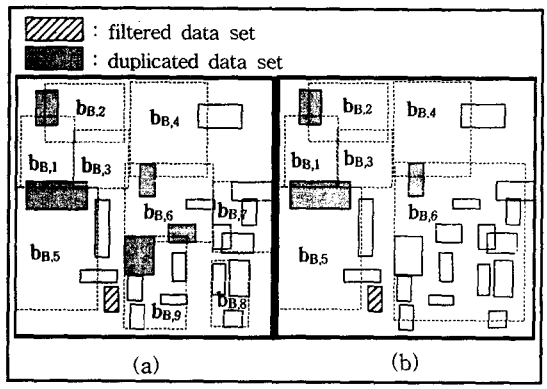

그림 3. 버킷 오버플로우의 처리 결과 비교; (a) SHJ의 오버플로우 처리 결과, (b) SHSJ의 버킷 오버플로우 처리 결과

그림 3에서 알 수 있듯이 SHSJ 알고리즘의 경우 SHJ 알고리즘보다 중복 저장되는 데이터의 개수가 훨씬 적다는 것을 확일 할 수 있다.

Algorithm SHSJ
Step1 첫 번째 입력 테이블에 대한 초기 버킷 영역을 정의한다; 정의된 버킷 영역에 첫 번째 입력 테이블의 데이터 집합을 할당한다;
Step2 첫 번째 입력 테이블에 대한 최종 버킷 영역을 두 번째 입력 테블의 버킷 영역으로 설정하여 두 번째 입력 테이블의 각 객체를 버킷 영역과 겹치는 모든 버킷 영역에 할당한다;
If 버킷 오버플로우 발생
버킷 분할을 수행하지 않고 오버플로우가 발생하지 않은 버킷으로 직접 삽입한다;
Step3 동일 영역을 갖는 해당 버킷 쌍 조인한다;
If 버킷 오버플로우가 발생 했던 버킷
알고리즘 **SSSJ**를 호출한다;
else
내부 plane-sweep 수행한다;
Step4 조인 결과중에서 중복된 객체 쌍을 제거한다;

그림 4. SHSJ 알고리즘

이 논문에서 제안하는 SHJS 알고리즘은 그림 4와 같다. Step1에서는 첫 번째 데이터 집합에 대한 초기 버킷 영역을 결정하고 첫 번째 데이터 집합을 정의된 버킷 영역으로 할당한다. Step2에서는 첫 번째 데이터 집합의 할당 후 생성된 최종 버킷 영역과 동일한 영역을 두 번째 해쉬 테이블의 버킷 영역으로 사용하여 두 번째 테이블의 데이터 집합을 할당한다. Step3 단계에서는 Step1과 Step2에서 생성된 두 해쉬 테이블에 대해서 동일한 영역을 공유하는 버킷 쌍을 SSSJ 알고리즘을 사용하여 조인 한다. 마지막으로 Step4에서는 조인 결과에서 중복된 결과를 제거하여 최종 결과를 반환하는 과정으로 처리된다.

4. 실험 및 평가

본 장에서는 앞 장에서 제안한 SHSJ 알고리즘의 성능 평가를 위한 실험 환경과 방법 및 실험 결과를 기술한다. 본 실험은 Intel Xeon 550MHz CPU, 1GHz RAM, 40GB DHH, Solaris 8 환경에서 수행하였으며 SHSJ 알고리즘의 구현을 위해 GNU C 컴파일러(gilbc 2.95.3)를 사용하였다.

실험 데이터로는 표준 평가 데이터 중 하나인 Tiger/line[8]을 사용하였으며, 그 특성은 표 1과 같다.

표 1. 실험 데이터의 특성

항 목		RhodeIsland	Connecticut	NewJersey	NewYork
도로	공간 객체 수	70058	197066	443472	972525
	객체 집합 크기	3.15B	8.71MB	20MB	43.9MB
수로	공간 객체 수	6475	27547	48202	157793
	객체 집합 크기	298KB	1.22MB	2.18MB	7.14MB

표 1과 같이 네 개 주에 대한 도로와 수로 데이터를 조인을 위한 입력 테이블로 사용하였고 평가 항목으로는 입력 테이블의 크기(네 개 주의 데이터 집합), 편중도, 버퍼 크기를 사용하였다.

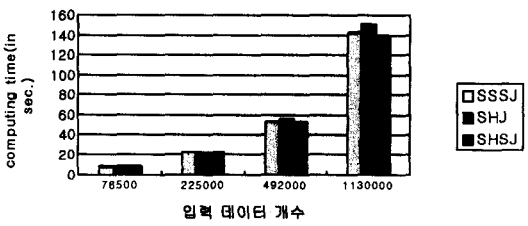

그림 5. 입력 데이터의 크기에 대한 조인 수행 시간

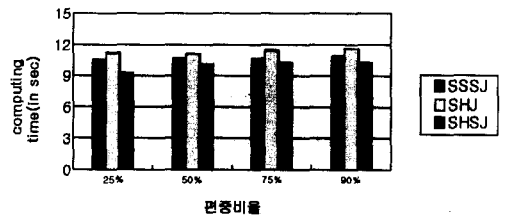

그림 6. 입력 데이터의 편중도에 대한 조인 수행 시간

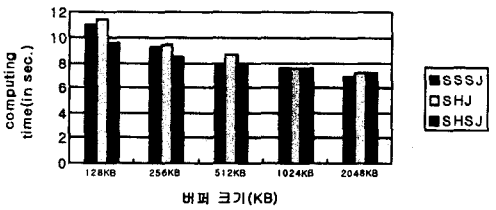

그림 7. 버퍼 크기에 대한 조인 수행 시간

그림 5는 버퍼의 크기를 1MB로 고정하고 크기가 서로 다른 네 개의 주를 입력 테이블로 했을 경우에 대한 조인 수행 시간을 나타낸다. 실험 결과 SHSJ 알고리즘이 기존 SHJ, SSSJ 알고리즘 보다 테이블의 크기가 클수록 우수한 성능을 나타냈다.

그림 6은 편중도에 대한 성능 특성을 분석하기 위하여 RI 주의 데이터를 재생성하여 편중도가 각각 25%, 50%, 75% 90%인 입력 테이블로 실험하였다. 실험 결과 SHJ 알고리즘은 편중도에 따라 수행 시간이 조금씩 증가하는 반면 SHSJ 알고리즘의 경우 조인 수행 시간이 거의 일정하였다.

그림 7은 입력 테이블을 RI주로 선택하고 버퍼크기를 각각 128KB, 256KB, 512KB, 1MB로 달리 하였을 때의 조인 성능을 나타낸다. 세 알고리즘 모두 버퍼 크기가 증가할 수록 전체적인 조인 수행시간이 감소하는 특성을 보였으며 특히, SHSJ 알고리즘의 경우 버퍼 크기가 작을 경우 SHJ와 SSSJ 알고리즘보다 월등히 우수한 성능을 나타냈다.

5. 결론 및 향후 연구과제

이 논문에서는 인덱스가 존재하지 않으며 편중 분포를 갖는 두 입력 테이블에 대해 효율적인 공간 조인 연산을 제공하기 위해 SHJ 알고리즘을 개선하여 SHSJ 알고리즘을 제안하였다. 제안된 SHSJ 알고리즘의 실험 평가를 위해 실제 Tiger/line 데이터를 사용하여 기존 SHJ 알고리즘과 SSSJ 알고리즘과 조인 성능을 비교 분석하였다. 실험 결과로서, SHSJ 알고리즘은 기존 SHJ 알고리즘과 SSSJ 알고리즘 모두에 대해 입력 데이터 집합의 크기가 클수록, 조인 연산 수행에 소요되는 버퍼의 크기가 작을수록, 그리고 입출력 버퍼의 크기가 작을수록 우수한 조인 성능을 나타냈다. 따라서 이 논문에서 제안한 SHSJ 알고리즘은 인덱스가 존재하지 않으며 편중된 분포를 갖는 두 입력 테이블에 대한 공간 조인 연산이 요구되는 공간 질의 처리에 효율적으로 이용될 수 있을 것이다.

향후 연구 과제로는 SHSJ 알고리즘과 기존 PBSM, seeded-tree 알고리즘 각각에 대한 성능 비교와 더불어 입력 테이블에 인덱스가 존재할 경우 기존 R-tree Join 알고리즘과의 성능 분석이 필요하다.

참고 문헌

[1] M. L. Lo and C. V. Ravishankar, "Spatial Hash-Joins," In Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 209-220, May 1996.

[2] L. Arge, O. Procopiuc, S. Ramaswami, T. Suel, and J. Vitter, "Scalable weeping Based Spatial Join," In Proceedings of International Conference on Very Large Data Bases, pp. 570-581, Aug. 1998.

[3] R. Elmasri and S. B. Navathe, Fundamental of Database Systems, 3rd edition, Addison-Wesley Publishers, 2000.

[4] M. L. Lo and C. V. Ravishankar, "The Design and Implementation of Seeded Trees: An Efficient Method for Spatial Joins," IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 1, pp. 136-151, 1998.

[5] J. M. Patel and D. J. DeWitt, "Partition Based Spatial-Merge Join," In Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 259-270, Jun. 1996.

[6] N. Koudas and K. Sevcik, "Size Separation Spatial Join," Proc. ACM SIGMOD International Conference Management of Data, pp. 324-335, May 1997.

[7] R. H. Buting and W. Schilling, "A Practical Divide-and-Conquer Algorithm for the Rectangle Intersection Problem," Information Sciences, Vol. 42, No. 2, pp. 95-112, July 1987.

[8] U. S, Bureau of the Census, "2002 Tiger/Line Files," 2002.

[9] 심영복, 이종연, 정순기, "효율적인 버킷 분할과 조인 방법을 이용한 공간 해쉬 스트립 조인 알고리즘 설계," 한국정보처리학회 제20회 추계학술발표논문집, 제10권 제2호, pp. 1367-1370, 2003년 11월.