제 26 회 한국정보처리학회 추계학술발표대회 논문집 제 13 권 제 2 호 (2006. 11)

데이터 스트림 환경에서의 조인 연속 질의의 그리디 질의 최적화 성능 분석

박홍규, 이원석
연세대학교 컴퓨터과학과
e-mail : {gladiator11, leewo}@database.yonsei.ac.kr

Greedy Query Optimization Performance Analysis for Join Continuous Query over Data Streams

Hong Kyu Park, Won Suk Lee
Dept. of Computer Science, Yonsei University

요 약

최근에 제한된 데이터 셋보다 센서 데이터 처리, 웹 서버 로그나 전화 기록과 같은 다양한 트랜잭션 로그 분석 등과 관련된 데이터 스트림 처리에 더 많은 관심이 집중되고 있으며, 특히 데이터 스트림의 질의 처리에 대한 관심이 증가하고 있다. 본 논문에서는 질의 중에서 2 개 이상의 스트림을 조인하는 조인 연속 질의를 처리하는 방법과 성능에 대해서 연구한다. 각 조인의 비용을 스트림의 입력 속도와 조인 선택도를 이용한 조인 비용 모델로 정의하고 그리디 알고리즘을 이용하여 최적화하는 기법을 제안하고 실험을 통해 다양한 스트림 환경에서 최적화 알고리즘이 어떤 성능을 보이는 지를 알아본다.

1. 서론

최근에는 제한된 데이터 셋을 처리하는 것이 아닌 데이터 스트림 처리에 더 많은 관심이 집중되고 있다. 실제로, 센서 데이터 처리, 인터넷 트래픽 분석, 주식 거래, 웹 서버 로그나 전화 레코드와 같은 다양한 트랜잭션 로그 분석등과 관련된 데이터 스트림 처리 분야에 많은 연구가 진행되었다.[1,8,9]

데이터 스트림 환경에서 조인의 경우, 단순히 셀렉션 연산자를 처리하는 것에 비해서 조인 대상 스트림에 저장되어 있는 스트림에 접근하여 결과를 출력해야 하기 때문에 시스템에 커다란 부하를 주며, 다중 조인일 경우에는 그 부하는 훨씬 커질 것이다. 또한 스트림의 특성에 따라 처리 시간이 많이 차이가 나기 때문에 질의 최적화가 매우 중요하다. 기존에 존재하던 DBMS의 경우 시스템 내에 저장되어 있는 데이터들에 대한 메타데이터들을 바탕으로 질의 최적화를 수행하는데 반해, 데이터 스트림 환경에서는 실시간으로 들어오는 스트림 데이터들에 대해서 질의를 처리하여야 한다. 들어오는 스트림 데이터의 특성에 따라 주어진 질의를 처리하는 데 최적화된 질의계획은 달라질 수 있으므로, 정적인 질의 계획을 가지고 질의를 수행한다면, 시스템의 성능이 저하될 수 있다. 그러므로 스트림 데이터의 특성이 바뀜에 따라서 동적으로 적응적 질의 최적화를 수행하는 것이 필요하다.

본 논문에서는 n개의 스트림을 조인하는 단일 연속 질의를 위한 질의 최적화 알고리즘을 제안한다. 제안하는 알고리즘은 조인 비용 모델과 통계 테이블을 이용하여 최적화된 질의 계획을 찾는다. 여기에서 최적화된 질의 계획은 수행하는 조인들의 순서와 같다. 조인의 비용 모델은 해당 조인을 수행하는 데 필요한 튜플 접근 횟수로 정의하고, 통계 테이블은 조인 가능한 모든 조인의 비용을 가지고 있다고 가정한다. 비용 모델과 통계 테이블을 이용하여 가장 작은 비용을 가지는 질의 계획을 산출하게 된다. 그러나 2개 이상의 스트림을 조인하는 질의를 수행하는 가능한 질의 계획의 개수는 기하급수적으로 늘어나기 때문에 실시간으로 데이터를 처리하고 있는 상황에서 이 모든 질의 계획에 대해서 비용을 검사하고 비용이 가장 작은 질의 계획을 선택하고 수행하는 것은 오히려 시

스템의 성능을 저하시키는 요인이 될 수 있다. 그러므로 본 논문에서는 조인 비용 모델을 기반으로 한 greedy algorithm을 이용하여 최적화된 질의 계획을 찾는 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 먼저 2절에서는 관련 연구에 대해서 살펴보고 3절에서는 조인 비용 모델을 정의하고, 이 모델을 기반으로 질의 최적화 알고리즘에 대해서 설명한다. 4절에서는 제안한 알고리즘의 성능을 평가하고 마지막으로 5절에서는 결론 및 앞으로의 연구 방향에 대해 논의한다.

2. 관련 연구

데이터 스트림 환경에서 효율적인 스트림 처리를 위해서 조인의 비용을 예측하기 위한 조인 비용 모델의 연구가 진행되고 있다. [2]에서는 단일 이진 조인에 대해서 스트림 데이터를 처리하는 방식(Hash join, Nested join)에 따라서 각각의 조인 비용 모델을 정의하고 비용 모델과 실제 스트림 처리 실행 시간이 비례함을 실험으로서 증명하였다. [3]에서는 처리 비용과 메모리 사용의 관점에서 효율적인 자원 할당을 예측하기 위해서 비용 모델을 정의하였다.

데이터 스트림 환경에서 2개 이상의 스트림 정보를 처리하는 방법으로는 크게 2가지 방법이 있다. 그 중 하나는 각 조인의 출력 결과를 저장을 하고, 이 결과를 다음 조인에 사용하는 binary linear processing tree(BLPT)[4]이며, 또 다른 하나는 조인의 중간 결과를 저장하지 않고 다중 중첩 반복(multi-way nested loop)을 사용하는 pipelined processing tree(PPT)이다. BLPT는 [5]에서 사용되며, PPT는 [6], [7], [8]에서 사용된다.

[5]에서는 다중 조인 질의의 출력 속도를 최대화하기 위한 조인 방법을 제시하며, [6]에서는 스트림의 입력 속도가 낮은 스트림일수록 중첩 반복의 바깥쪽(outer loop)으로 하는 방식으로 스트림을 처리하는 방법과 몇 개의 경험적(heuristic)인 방법을 제시하였다. 또한 [7]은 스트림 데이터가 들어왔을 때 각각의 스트림들과 조인 결과가 나오는 지를 확인하는 삭제 검색 단계(drop probing)와 조인을 수행하여 결과를 출력하는 출력 생성(output generation) 단계, 두 단계에 거쳐서 조인을 수행하며, 삭제 검색 단계에서 조인 결과의 유무를 모니터링하여 이를 조인 순서에 반영하며 [8]에서는 Eddy와 티켓 라우팅이라는 방법을 이용하여 다수의 스트림들을 처리한다.

3. 알고리즘

본 절에서는 이진 조인의 비용 모델을 정의하고, 이를 바탕으로 조인 질의를 최적화하는 알고리즘을 제안한다.

3.1 조인 비용 모델

여기에서는 단일 이진 조인의 비용을 측정하고 이에 대해서 비용 모델을 정의한다. 기본적으로 2 개의 스트림을 조인하는 것은 2 가지 특징을 가지고 있으며, 그 중 하나는 슬라이딩 윈도우를 이용한 윈도우 조인(windowed join)이라는 것이다. 슬라이딩 윈도우는 앞에서도 언급했듯이 유한한 메모리 내에서 무한으로 들어오는 스트림을 처리하기 위해서 정의된 것이다. 예를 들어 스트림 R 과 S 를 조인하는데 스트림 R 에 들어오는 데이터들 중 최근 t_1 시간에 들어온 튜플들만을 저장하고 스트림 S 에서 들어오는 스트림도 마찬가지로 최근 t_2 시간에 들어온 튜플들 만을 저장하며, 저장된 튜플들만을 대상으로 조인을 수행하게 된다. 이 조인은 새로운 입력 튜플이 들어올 때마다 새로운 결과를 생성하면서 연속적으로 질의를 수행하게 된다. 그림 1 은 윈도우 조인을 그림으로 표현한 것이다. A, B 라는 2 개의 스트림이 존재하며, 입력 스트림 A 에 대해서 입력 속도 λ_a 와 윈도우 크기를 T_a 라고 가정하며(똑같이 입력 스트림 B 에 대해서 λ_b, T_b 라고 각각 정의한다.) 이동 윈도우의 크기는 $\lambda_a T_a$ 이다(똑같이 $\lambda_b T_b$).

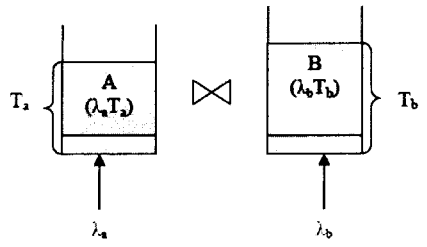


그림 1. Windowed Join

스트림 조인의 특징 중 다른 하나는 SJ(Symmetric Join)이다. 스트림 A 와 B 의 데이터는 고정되어 있는 것이 아니라 스트림이 계속 흘러 들어오기 되는데, 새로운 스트림 데이터가 흘러 들어올 때마다 그 데이터를 이용하여 조인을 해주어야 한다. 즉 A 에서 새로운 데이터가 들어오면 B 의 유효한 스트림 데이터와 조인을 하여야 하며 B 에서 새로운 데이터가 들어오면 A 의 유효한 스트림 데이터와 조인을 하여야 한다. 이와 같이 각 스트림에서 데이터가 들어올 때마다 조인을 하는 것을 Symmetric Join 이라고 하며, 본 논문에서는 해시조인 방법을 사용하는 SHJ(Symmetric Hash Join)을 기본으로 한다.

조인을 수행할 때 필요한 연산을 그림 1 에 비추어서 보면, A 스트림에서 튜플이 들어오면 3 가지의 연산이 필요하다. 조인 결과를 만들기 위해 윈도우 B 를 보는 연산, A 스트림에서 들어온 튜플을 저장하는 연산 그리고 A 스트림에서 윈도우 범위를 벗어난 튜플을 삭제하는 연산 이렇게 3 가지이다. 이 것을 기반으로 조인 비용식을 만들어 보면 다음과 같다. [2]

$$C_{A \bowtie B} = \lambda_a(\text{prove(b)+insert(a)+invalidate(a)}) + \lambda_b(\text{prove(a)+insert(b)+invalidate(b)})$$

본 논문에서는 인용된 논문에서와는 달리 삭제 연산(invalidate)을 튜플이 들어올 때마다 하는 것이 아니라

정의 3.1. (Greedy Invariant) $Cost(J_1)$, $Cost(J_2)$,....$Cost(J_{n-1})$는 Greedy Invariant를 만족한다.

$$Cost(J_i) \le Cost(J_j), \qquad 1 \le i \le j \le n-1$$

조인이 이루어질 때 수행하기 때문에 별도의 비용이 필요 없게 되며, 이에 따라 조인의 비용은 다음과 같이 정의할 수 있다.

$C_{A \bowtie B} = \lambda_a(\text{prove(b)+insert(a)}) + \lambda_b(\text{prove(a)+insert(b)})$

각 연산들의 비용을 정의하기 위해 용어는 그림 2 와 같으며, 이 용어들은 윈도우 B 도 동일하게 정의될 수 있다.

λ_a	스트림 A의 입력 속도
T_a	윈도우 A의 윈도우 크기
|A|	윈도우 A의 해시 버켓의 개수
A	윈도우 A($\lambda_a T_a$)에 있는 튜플의 개수
Cost(A,B)	스트림 A와 B의 조인 비용

그림 2. 용어 정리

위 용어들을 가지고 각 연산에 대한 비용식을 정리하면,

$$prove\ (a) = \frac{\lambda_b T_b}{|B|}$$

$$insert\ (a) = 1$$

로 정의할 수 있다.

그러므로 이진 조인의 비용은 다음과 같다.

$$C_{A \bowtie B} = \lambda_a\left(\frac{\lambda_b T_b}{|B|} + 1\right) + \lambda_b\left(\frac{\lambda_a T_a}{|A|} + 1\right) = \lambda_a\lambda_b\left(\frac{T_a}{|A|} + \frac{T_b}{|B|}\right) + (\lambda_a + \lambda_b)$$

위와 같이 정의된 이진 조인의 조인 비용 모델은 실행 시간과 비례하다.[2]

3.2 질의 최적화 알고리즘

이 절에서는 본 논문에서 제안하는 통계와 조인 비용 모델을 기반으로 greedy algorithm 을 이용하여 n 개의 스트림을 조인하는 질의 최적화에 대해서 설명한다. 질의 최적화 알고리즘을 설명하기에 앞서 몇 가지 가정을 하였다.

① 조인 연산만을 하는 질의를 대상으로 한다.
② 각 스트림의 입력 속도, 윈도우의 크기, 버켓 크기와 각 조인의 조인 선택도는 다 알고 있다.
③ 통계 테이블은 조인 가능한 모든 조인들의 비용을 알고 있다.(2번의 정보를 이용하여 산출할 수 있다.)

2 번의 가정에서 각 스트림의 입력 속도, 윈도우의 크기, 버켓의 크기는 기본 스트림과의 조인 비용을 구하기 위해서 사용되며, 각 조인의 조인 선택도는 조인 결과의 출력 속도를 구할 때 사용되며, 이 출력 속도는 다음 조인의 입력 속도를 산출하기 위해서 사용된다.

통계 테이블은 해당 조인의 입력 스트림에 대한 정보, 조인의 레벨(level), 조인 비용 등을 저장하고 있으며, 이 정보들은 알고리즘을 수행하는 데 사용된다.

n 개의 스트림을 조인하는 질의를 최적화하기 위해서 다음과 같은 Greedy Invariant(GI)를 만족하여야 한다.

정의 3.1. (Greedy Invariant) $Cost(J_1)$, $Cost(J_2)$,.....$Cost(J_{n-1})$ 는 Greedy Invariant 를 만족한다.

여기에서 $Cost(J_i)$는 최적화 알고리즘으로 인해 i 번째 선택된 조인의 비용을 의미한다. 우리의 질의 최적화 알고리즘의 목표는 알고리즘이 끝날 때까지 위의 Greedy Invariant 를 만족하는 것이며, 그 알고리즘은 그림 3 과 같다.

```
Optimize_greedy(qpt, ctl)
    qpt: 지금까지 선택된 조인 연산들을 저장
    ctl: 통계 정보
    temp_min: 최소 비용을 가진 조인을 임시로 저장하는 버퍼(초기값 NULL)
For(i=0;i<ctl_count;i++) do //통계 테이블 스캔
  If detect_collision(qpt, ctl[i]) == 0 do //충돌이 없다면
        //qpt에 저장된 조인 정보들과 현재 스캔하고 있는 조인이 충돌이 있는지 없는지 검사
        If temp_min is NULL
            temp_min에 통계 테이블에 있는 조인 저장
        else
            temp_min에 저장되어 있는 조인보다 현재 통계테이블의 조인의 비용이 더 작다면 temp_min에 저장
  temp_min을 qpt에 저장
새로운 조인이 선택하고 이를 저장한 qpt를 리턴
```

그림 3. 그리디 질의 최적화 알고리즘

그림 3 에 설명된 알고리즘을 모든 스트림을 조인할 때까지 반복 수행하면 질의 최적화를 마치게 된다. 여기에서 충돌이라고 하는 것은 2 개의 조인이 하나의 질의 계획에 공존할 수 있는지 없는지를 체크하는 것이다. 예를 들어, A, B, C 를 조인하는 질의에서 A 와 B 를 조인하는 연산과 B 와 C 를 조인하는 연산은 같은 질의 계획 안에 있을 수 없다. 이런 경우에 두 조인은 충돌이 난다고 정의하며, 이러한 조인은 알고리즘을 수행할 때에 대상 조인이 되지 않는다. 즉, A 와 B 를 조인하는 연산이 이미 선택되어져서 qp 에 저장되어 있다면, 그 다음 조인을 선택할 때 B 와 C 를 조인하는 연산은 아무리 조인 비용이 낮아도 (Cost(B,C) > Cost(A,B)는 만족되어야 한다.) 선택 대상에서 제외된다.

4. 실험 및 성능 평가

이 절에서는 이러한 질의 최적화 방법이 다양한 스트림 환경에서 어떤 성능을 가지고 있는 지에 대해서 실험을 한다. 성능의 비교는 그리디 질의 최적화 알고리즘을 통해서 선택된 질의 계획의 비용과 최적화된 절의 계획의 비용의 비교로 이루어진다.

실험 방법은 각 조인들의 조인 비용을 저장하고 있는 통계 테이블을 생성하고, 모든 가능한 하위 조인 집합들에 대해서 조인 비용을 계산하여 통계 테이블에 저장한다. 조인 비용은 3절에서 설명한 조인 비용

모델을 기반으로 계산하며 각 해시 버켓의 크기와 윈도우의 크기는 고정을 하고, 조인 선택도와 스트림의 입력 속도는 값의 범위를 결정하고 이 범위 내에서 랜덤하게 생성한다. 또한 조인 질의에 포함되는 스트림의 개수의 변화에 따른 실험도 하였다.

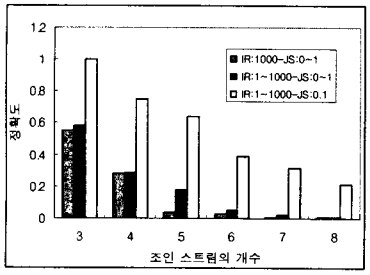
실험 1은 다양한 스트림 환경에서 조인 스트림의 개수를 증가시키면서 조인 스트림의 개수에 따른 그리디 알고리즘의 성능을 평가한다. 여기에서 정확도란 그리디 최적화 방법으로 구한 질의 계획과 실제 최적화 질의 계획이 같을 확률이다.

IR:1000-JS:0~1	입력 속도:1000, 조인 선택도 0~1의 값 중 랜덤하게 생성
IR:1~1000-JS:0.1	입력 속도: 1~1000의 값 중 랜덤하게 생성, 조인선택도 0.1 고정
IR:1~1000-JS:0~1	입력 속도: 1~1000, 조인 선택도 0~1사이의 값을 랜덤하게 생성

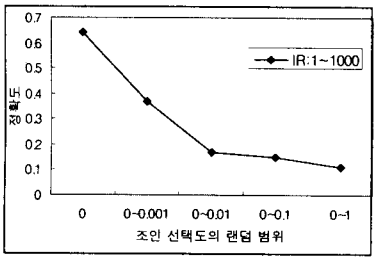
표 1. 실험 1 에서 사용하는 데이터셋

실험 1 에서 알 수 있는 사실은 조인하는 스트림의 개수가 많아질수록 정확도가 떨어진다는 점과 조인 선택도와 입력 속도의 랜덤 범위, 특히 조인 선택도의 랜덤 범위가 성능에 영향을 끼친다는 점이다.

실험 2 는 입력 속도의 랜덤 범위와 조인 스트림의 개수(6 개)로 고정하고, 조인 선택도의 랜덤 범위를 변화시키면서 실험을 하였다. 그래프와 같이 조인 선택도의 랜덤 범위가 작을수록 그리디 질의 최적화 방법은 좀 더 좋은 성능을 가진다.



실험 1.조인 스트림의 개수에 따른 성능 비교



실험 2.조인 선택도의 랜덤 범위에 따른 성능 비교

5. 결론 및 연구방향

본 논문에서는 스트림을 처리하는 조인의 조인 비용을 모델로 정의하고, 이 모델과 그리디 알고리즘을 이용하여 질의 최적화하는 방법을 제안하였고, 실험에서는 조인 스트림의 개수에 따른 성능 비교와 조인 선택도와 입력 속도의 랜덤 범위에 성능 비교를 하였으며, 그 결과 조인 스트림의 개수가 많을수록 성능이 저하됨을 알 수 있었으며, 조인 선택도의 랜덤 범위에 따라서 성능의 차이가 있다는 것을 알게 되었다. 본 논문에서는 2개 이상의 스트림을 조인하는 단일 질의를 처리하는 기법을 연구하였으므로, 향후 연구 방향으로 단일 질의가 아닌 다중 질의를 처리하는 것으로 확장할 수 있을 것이다. 또한 조인 비용 모델을 조인 선택도도 고려하는 모델로 확장시켜, 좀 더 정확한 비용 예측이 가능하도록 할 수 있을 것이다.

참고 문헌

[1] The STREAM groups STREAM: The Stanford Stream Data Manager (short overview paper) *IEEE Data Engineering Bulletin, March 2003*

[2]J. Kang, J. F. Naughton, and S. Viglas. Evaluating window joins over unbounded streams. *In Proc. of the 2003 Intl. Conf. on Data Engineering, Mar. 2003*

[3] Michael Cammert, J¨urgen Kr¨amer, Bernhard Seeger, Sonny Vaupel A Cost model for adaptive Resource Management in data stream systems

[4] A. Krishnamurthy, H. boral, and C. Zaniolo. Optimization of nonrecursive queries. In *proc. Of the 1986 Intl. Conf. In Very Large Data Bases*, pages 128-137, Aug. 1986.

[5] Stratis D. Viglas Jeffrey F. Naughton Josef Burger. Maximizing the output rate of multi-way join queries over streaming information sources

[6] Stratis Viglas, Jeffrey F. Naughton, and Josef Burger. Maximizing the output rate of multi-way join queries over streaming information sources. *In VLDB, 2003.*

[7] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, J. Widom, Adaptive ordering of pipelined stream filters, *in: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, 2004, pp. 407{418.*

[8] S. Madden, M. Shah, J. Hellerstein, and V. Raman. Continuously adaptive continuous queries over streams. *In Proc. Of the 2002 ACM SIGMOD Intl. Conf. on Management of Data*, pages 49-60, June 2002.

[9] J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. *In ACM SIGMOD, 2000.*

[10] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems *Invited paper in Proc. of PODS 2002, June 2002*