윈도우 조인에서 실행 주기에 따른 성능 비교

김현규
삼육대학교 컴퓨터학부
e-mail : hgkim@syu.ac.kr

Performance Comparison of Window Joins with Different Evaluation Periods

Hyeon-Gyu Kim
Division of Computer,
Sahmyook University

요 약

윈도우 조인은 데이터 스트림 처리에서 중요한 역할을 차지한다. 많은 스트림 응용에서 시간-기반 윈도우를 이용하여 스트림을 분할하고 이를 바탕으로 조인을 수행한다. 본 논문에서는 윈도우가 업데이트되는 주기 내에서 조인을 자유로이 수행할 수 있으며, 이 경우 조인 결과의 정확성이 보장됨을 보여준다. 또한, 실험을 통해 조인 주기가 짧아질수록 윈도우 조인의 성능이 우수해짐을 보여준다.

1. 서론

데이터 스트림에 대한 제한 없이 조인을 수행하기는 불가능하다. 이는 데이터 스트림이 본질적으로 무한하며, 조인은 모든 입력 데이터가 준비되었을 때에 한해 수행될 수 있기 때문이다[1]. 이러한 문제를 풀기 위해, 많은 데이터 스트림 처리 엔진은 슬라이딩 윈도우(Sliding Windows)[2]를 활용하여, 처리 범위를 최근 t 시간 내에 전달된 튜플로 제한시킨 후 조인을 수행한다.

예를 들어, 어떤 네트워크 경로에서 최근 5 분 내 전달된 패킷을 식별해 내기 위한 질의를 생각해 보자. 식별값은 매 1 분 주기로 업데이트된다. 네트워크 경로는 세 개의 라우터 R1, R2, R3 로 구성되어 있다고 하자. 이 경우 패킷을 식별하기 위해, 세 개의 라우터로부터 전달되는 스트림 상에서 패킷 아이디 pid 를 기반으로 한 동등조인(Equijoin)이 이용될 수 있다. 이를 위한 질의는 Q1 과 같이 기술될 수 있다. 아래에서 RANGE 는 윈도우의 크기를 나타내기 위한 파라미터이며, SLIDE 는 윈도우의 업데이트 주기를 나타낸다.

```
Q1.   SELECT *
      FROM  R1 [RANGE 5 mins, SLIDE 1 min],
            R2 [RANGE 5 mins, SLIDE 1 min],
            R3 [RANGE 5 mins, SLIDE 1 min]
      WHERE R1.pid = R2.pid and R2.pid = R3.pid
```

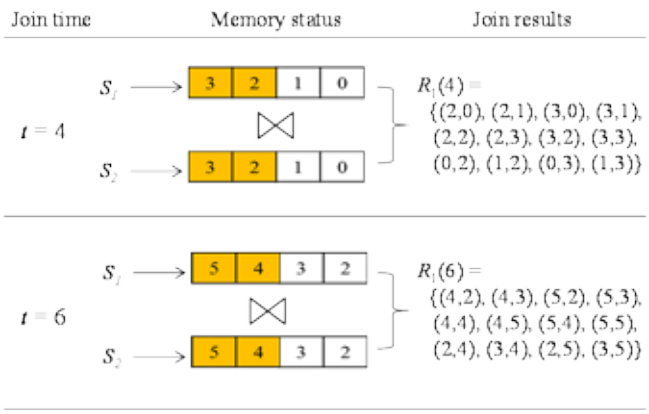
위 질의에서 윈도우 업데이트 주기가 1 분으로 주어졌으므로, 조인은 매 1 분 주기로 실행된다. 윈도우가 업데이트될 때마다 5 분 크기의 데이터 셋이 만들어진다. 예를 들어, Q1 이 어떤 시간 t 초에 수행되었다고 하자. 그러면 t 초 이전까지 전달된 튜플 중, 전달 시간이 $(t - 300)$초보다 적은 튜플은 모두 메모리에서 제거된다. 그리고 나머지 튜플에 대해 조인이 수행된다. 이는 매 1 분마다 반복된다.

위 예에서 조인의 실행 주기를 더욱 단축시킬 수 있다. 예를 들어, 매 30 초마다 수행할 수도 있고, 매 1 초마다 수행할 수도 있다. 즉, 조인의 실행 주기를 1 분 이하로 설정하는 것이 가능하며, 이는 조인의 성능에 영향을 미칠 수 있다. 본 논문에서는 서로 다른 조인 주기에 따른 성능을 실험을 통해 비교하였다. 그리고, 조인 주기가 달라지더라도 조인 결과의 정확성이 보장될 수 있음을 간단한 예를 통해 보여준다.

2. 조인 알고리즘

어떤 속성 A 를 기반으로, m 개의 데이터 스트림을 조인한다고 가정하자. S_i 를 i 번째 스트림이라 하고, 해당 스트림의 윈도우 크기를 W_i 라 하자. 즉 W_i 는 질의에 정의된 RANGE 값에 해당한다. 어떤 t 시간에 튜플 s 의 도착 시간이 $(t - W_i)$보다 크다면, s 는 윈도우 스트림 $S_i[W_i]$에 속한다고 할 수 있다. 이러한 경우, 윈도우 스트림에 기반한 m 차 동등조인(m-way Symmetric Hash Join)[3]은 $S_1[W_1] \bowtie_A S_2[W_2] \bowtie_A \ldots \bowtie_A S_m[W_m]$. 으로 표시할 수 있다. 그리고, s_i 가 $S_i[W_i]$에 속한 튜플이라고 하면, 조인 결과는 $s_1.A = s_2.A = \ldots = s_m.A$ 를 만족하는 튜플의 리스트 $(s_1, s_2, \ldots, s_m)$으로 표시될 수 있다.

모든 윈도우의 RANGE 와 SLIDE 파라미터 값이 각각 W 와 T 로 같다고 가정하자. 그러면 윈도우 조인을 위한 알고리즘은 아래와 같이 기술될 수 있다. 아래에서 T_e 는 조인 실행 주기를 의미한다.



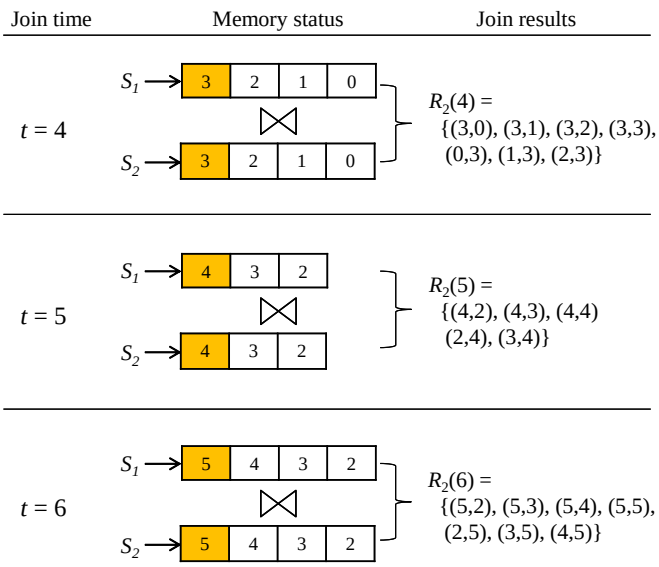
(그림 1) W = 4, T_e = 2 인 경우 조인 결과

Algorithm.1 *Window join*
Whenever T_e seconds expire …
1. For each tuple s arriving on S_k for the last T_e seconds
   1.1. Join s with all $S_i[W_i]$ ($i \neq k$)
   1.2. Add s to $S_k[W_k]$
2. If T seconds expire,
   2.1. Update all $S_i[W_i]$ by discarding expired tuples

만약 T_e가 T보다 작고 T mod T_e = 0 인 조건을 만족하면, 위 알고리즘은 항상 동일한 결과를 출력한다. 예를 들어, W = 4 이고 T = 2 인 이진 조인(Binary Join)을 생각해 보자. 편의상, 각 스트림에는 매 초 하나의 튜플이 들어온다고 가정하자. 튜플은 자신이 도착한 시간으로 표시되며, 조인 결과는 조인에 참여한 튜플의 도착시간의 쌍으로 표시된다.

먼저 T_e = T = 2 인 경우 조인 결과를 생각해 보자. 그림 1 은 이러한 경우 Algorithm 1 의 결과를 보여주고 있다. 그림의 윗 부분은 4 초가 되었을 때, 메모리 상태와 조인 결과를 보여준다. 4 초 이전에는 각 스트림 별로 도착시간이 0 에서 3 인 4 개의 튜플이 메모리에 유지된다. 그리고 정확히 4 초가 되었을 때, 조인이 수행되며 결과는 $R_1(4)$와 같다. 다음 조인은 그림의 아랫 부분에서 볼 수 있듯이 2 초가 지난 6 초에 수행된다. 6 초 이전까지 메모리에는 도착시간이 2 에서 5 인 튜플이 유지되며, 6 초가 되었을 때 조인 수행 결과는 $R_1(6)$와 같다.

다음으로 T_e = 1 인 경우 조인 결과를 생각해 보자. 이 경우 각 초마다 조인이 수행된다. 이 때 주의할 점은 윈도우 과정은 매 2 초마다 수행된다는 점이다. 그렇지 않을 경우 틀린 결과가 발생할 수 있다[4]. 그림 2 에서 볼 수 있듯이, 매 t 초마다 조인 결과 $R_2(t)$가 생성이 된다. 위 예에서 볼 수 있듯이, T_e가 달라지면 조인 주기가 달라진다. 그러나 그 결과는 조인 주기에 상관없이 동일하다. 이와 관련하여, 위 예제에서는 $R_2(t) = R_1(t-1) + R_1(t)$로 표현될 수 있다. 반대로, 조인 성능은 조인의 실행 주기에 따라 달라질 수 있다. 아래에서는 실험을 통해 실행 주기에 따른 성능 변화를 보이고자 하였다.



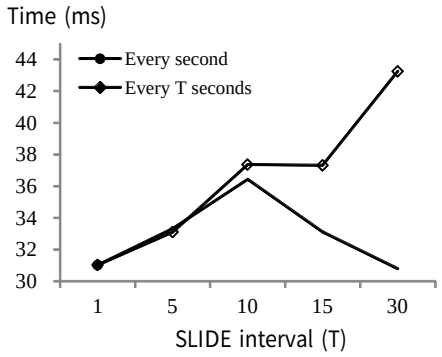
(그림 2) W = 4, T_e = 1 인 경우 조인 결과

3. 조인 주기에 따른 성능 차이

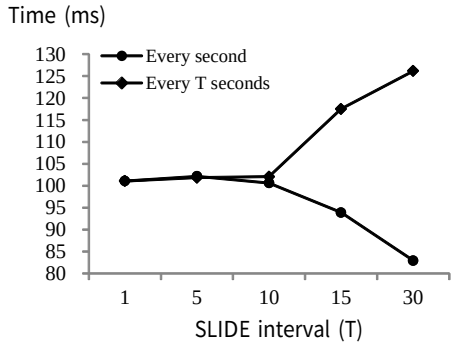
실험을 위해, Java 언어를 이용하여 Algorithm 1 을 구현하였다. 구현된 m 차 동등조인 알고리즘은 내부적으로 해시 테이블을 사용하며, 입력 파라미터로 윈도우 크기 N, 윈도우 업데이트 주기 T, 그리고 조인 실행 주기 T_e 를 받아들인다. 실험에서는 3 개의 원격 클라이언트로부터 생성되는 스트림을 서버에서 받아들여 조인을 수행하도록 구성하였다. 실험에 사용된 기종은 Window 7 운영체제를 이용하며, Inter Core i3-2120 3.3 GHz CPU 와 4G 메모리를 이용하였다.

실험에는 윈도우 업데이트 주기 T 가 주어졌을 때, 조인 실행 주기를 윈도우 업데이트 주기와 같은 T 초로 했을 경우와 1 초로 했을 경우 두 가지만 비교하였다. 윈도우 크기 W 는 30 초로 설정되었다. 그림 3 은 실험 결과를 보여준다. 가로축은 주어진 윈도우 업데이트 주기 T 를 나타내며, 세로축은 윈도우 조인에 걸린 시간을 Millisecond 단위로 표시하였다. 그리고, 실행 주기가 T 인 경우를 "Every T seconds", 그리고 실행 주기가 1 초인 경우를 "Every second"로 표시하였다.

실험 결과, 그림 3(a)는 각 스트림으로부터 초당 10000 개의 튜플이 전달되었을 경우에 조인 성능을 보여주며, 그림 3(b)는 초당 30000 개의 튜플이 전달될 때 성능을 보여준다. 그림에서 볼 수 있듯이, T 가 작을 경우 성능에 별 차이를 보이지 않다가, T 가 커질수록 점점 매 1 초마다 조인을 실행하는 경우의 성능이 좋아지는 것으로 결과가 나타났다. 이는 매 T 초마다 조인을 수행해야 하는 경우에 비해, 매 초 실행하는 경우에 메모리에 저장되는 튜플의 수가 평균적으로 더욱 작아진다 (2 장의 예에서 유추해 볼 수 있다). 이로부터 조인의 성능은 조인의 실행 주기가 짧아질수록 더욱 좋아지는 것으로 예측할 수 있다.



(a) Total join time (I = 10000)



(b) Total join time (I = 30000)

(그림 3) 조인 실행 주기에 따른 성능 비교

4. 결론

본 논문에서는 윈도우 조인에서 윈도우 업데이트 주기가 T 로 주어졌을 때, 조인의 실행 주기는 T 보다 더욱 적어질 수 있음을 보였다. 조인의 실행 주기를 T_e라 하면, T_e가 T 보다 작고 동시에 $T \bmod T_e = 0$ 을 만족하면 조인 결과의 정확성이 보장됨을 간단한 예를 통해 제시하였다. 이 경우, 조인의 실행 주기에 따라 윈도우 조인의 성능이 달라질 수 있다. 이를 확인하기 위해 실험을 수행하였으며, 그 결과, T_e 가 작아질수록 조인 성능이 향상됨을 알 수 있었다. 이는 T_e 가 적을수록 메모리에 저장되는 튜플의 수가 감소함에 기인한다.

참고문헌

[1] Brian Babcock et al., “Models and Issues in Data Stream Systems”, *In Proc. of the 21st ACM PODS*, 2002, pp.1-16.

[2] Arvind Arasu, Shivnath Babu and Jennifer Widom, The CQL Continuous Query Language: Semantic Foundations and Query Execution, *The VLDB Journal*, Vol.15(2), (2006) pp.121-142.

[3] Stratis D. Viglas, Jeffrey F. Naughton, and Josef Burger, “Maximizing the Output Rate of Multi-Way Join Queries over Streaming Information Sources”, *In Proc. of the 29th VLDB conference*, 2003, pp.285-296.

[4] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A. Tucker, “Semantics and Evaluation Techniques for Window Aggregates in Data Streams”, *In Proc. of the 2005 ACM SIGMOD international conference*, pp.311-322.