Interval Skip Lists를 이용한 정확도기반 우선순위 검색 기법의 설계

이은식 · 조대수
동서대학교

A Design of Priority Retrieval Technique based on Accuracy using The Interval Skip Lists

Eun-Sik Lee · Dae-Soo Cho
Dongseo University
E-mail : crubeido@nate.com, dscho@dongseo.ac.kr

요 약

전통적인 Pub/Sub(Publish/Subscribe) 시스템은 중개자(Broker)를 통해 출판자(Pub)의 출판조건(Event)에 매칭되는 구독자(Sub)의 모든 구독조건(Subscription)들을 검색한다. 즉, 출판조건과 구독조건간의 매칭의 정도는 고려되지 않고, 매칭의 유무만을 판단한다. 매칭된 구독조건 간에도 우선순위가 존재할 수 있다. 따라서 기존 Pub/Sub 시스템의 기능을 확장한 우선순위 Pub/Sub 시스템이 필요하다.

이 논문에서는 구독조건간의 우선순위를 결정하기 위해 검색결과의 정확도를 정의 하고 우선순위에 따라 구독조건을 검색할 수 있는 정확도 기반 우선순위 검색기법을 제안한다. 본 검색기법은 IS-Lists(Interval Skip Lists)를 기반으로 동작하며, 정확도를 통해 정렬된 검색 결과를 얻을 수 있다.

ABSTRACT

Traditional Pub/Sub(Publish/Subscribe) Systems search all subscriptions that match an incoming event by broker(i.e. it is not considering the accuracy of matching between an incoming event and subscriptions and only consider that an event either matches a subscription or not). However, subscriptions that match an event may have priority, therefore, we need priority Pub/Sub system.

In this paper, we define what the accuracy means in order to prioritize among subscriptions and propose the Priority Retrieval Technique based on accuracy that able to search subscriptions. The Priority Retrieval Technique is based on IS-Lists. We can search the results ordered by accuracy.

키워드

Publish/Subscribe(Pub/Sub) system, Interval Skip Lists, Accracy, Priority Retrieval

Ⅰ. 서 론

전통적인 Pub/Sub(Publish/Subscribe) 시스템[1]의 구조는 출판자(Pub), 출판조건(Event), 중개자(Broker), 구독자(Sub), 구독조건(Subscription)으로 이루어진다. 그리고 구독자가 등록해 놓은 수많은 구독조건들을 출판자의 출판조건에 효율적으로 매칭시키기 위해 설계된다. 구독조건이 간격으로 표현된 경우 효율적인 매칭을 위해 중개자에서는 간격(Interval) 색인을 사용한다. 다음은 전통적인 Pub/Sub 시스템을 적용한 취업사이트의 절차를 보여준다.

① 구인자는 구인조건을 중개자에 등록한다.
② 구직자는 자신의 구직조건을 중개자에 제공한다.
③ 중개자는 구직조건에 만족하는 모든 구인조건을 검색한다.

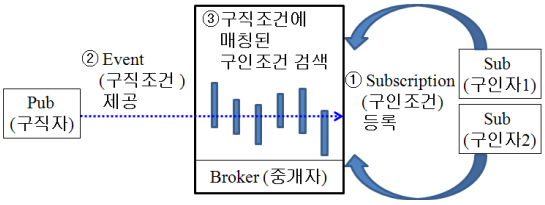

그림 1. 전통적인 Pub/Sub시스템을 적용한 취업 사이트의 예

현재의 Pub/Sub 시스템에서는 구직조건에 매칭된 모든 구인조건들을 검색한다. 구인조건 간의 매칭의 정도는 고려하지 않고 매칭의 유무만을 판단한다. 하지만 구인조건들 간에도 우선순위가 존재할 수 있다. 따라서 우선순위에 따라 구인조건들을 검색할 수 있는 방법이 요구되며 우선순위 Pub/Sub 시스템[2]이 필요하다. 이 논문에서는 우선순위 결정을 위해 검색결과의 정확도를 정의 하고 우선순위 검색기법을 제안한다. 해당기법은 Segment Tree[3], Interval Tree[3][4], R-Tree[5] 등과 같은 간격색인중의 하나인 IS-Lists(Interval Skip Lists)[6][7]를 기반으로 동작한다.

본 논문의 구성은 2장에서는 해당기법을 설계하기 위한 관련 연구를 설명하고 3장에서는 해당기법의 설계를 위한 정확도 정의 및 알고리즘을 제안 하고 마지막으로 4장에서는 결론 및 향후과제를 설명한다.

Ⅱ. 관련 연구

2.1 Stabbing 질의 문제

Stabbing 질의 문제[6][7]에서 하나의 질의(Query)는 특정한 값으로 표현되고, 데이터는 일정한 길이를 갖는 간격(Interval)으로 표현된다. Stabbing 질의 문제는 수많은 간격 중 질의 값에 겹쳐지는 간격들을 빠른 시간 내 검색하는 것을 의미한다. 그림 2는 Stabbing 질의의 예를 보이고 있다. A부터 G까지 모두 7개의 간격데이터가 있다. 이때 '44를 포함하는 모든 간격데이터를 찾으시오' 라는 의미의 Stabbing 질의에 대해서 질의 결과로써 {A, B, D, G}를 검색하게 된다.

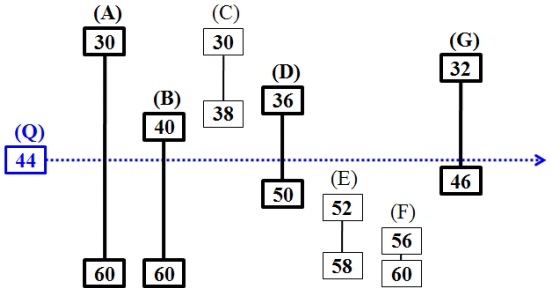

그림 2. Stabbing 질의의 예

2.2 Skip Lists

Skip List[8]는 연결리스트의 개선된 형태로써 한 노드가 1개 이상의 포인터를 가진다. 포인터의 개수는 한 노드의 레벨을 의미하고 랜덤 값에 의해 확률적으로 정해진다. 새로운 노드가 k 레벨을 가질 확률은 다음과 같다.

$$P(x=k)=\begin{cases}0 & \text{for } k<1\\ (1-p)\cdot p^{k-1} & \text{for } k\geq 1\end{cases}$$

p 값이 1/2일 때는 레벨1인 노드들이 전체에서 대략 1/2 이 되고 레벨2인 노드는 1/4, 레벨3인 노드는 1/8과 같은 확률로 계산한다.

그림 3에서 키(Key) 값 44의 검색 시 12, 23, 26 값을 가진 노드들을 생략(Skip)하였다. 연결리스트의 검색횟수는 6번이지만 Skip List의 검색횟수는 3번에 불과하다.

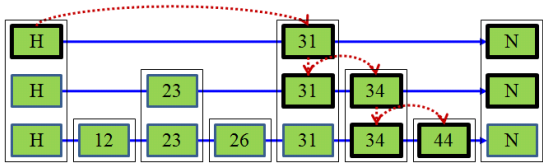

그림 3. Skip Lists의 예

2.3 IS-Lists(Interval Skip Lists)

IS-Lists는 Skip List에 간격의 개념을 확장시킨 간격색인이다. IS-Lists에서 간격데이터는 두 개의 노드로 구성되고 각 노드안의 숫자는 간격데이터의 한 끝점을 의미한다. 짝이 되는 노드의 숫자가

서로 다를 경우 일정한 길이를 갖는 간격데이터로 나타낼 수 있고 노드의 숫자가 같을 경우 길이가 없는 간격데이터로 나타낸다. 그림 4는 IS-Lists에서 간격을 표현하는 예를 보이고 있다. 간격 I = (20, 80)을 표현하기 위해서는 I가 모두 포함(Containment)하는 포인터와 노드를 검색해서 검색된 포인터들과 노드들 위에 마커(Marker)를 표시한다. 이 때 검색된 노드의 최상위(Maximality)레벨에 위치한 포인터 위에 마커를 표시 한다 . 상위 레벨의 포인터에 포함되는 포인터들에는 상위레벨의 포인터가 가진 마커를 표시할 필요가 없다.

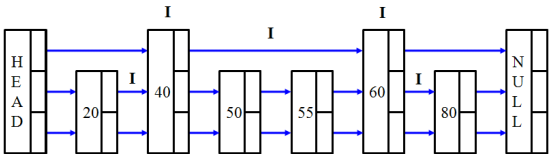

그림 4. IS-Lists의 간격표현의 예

Ⅲ. 정확도 기반 우선순위 검색기법의 설계

1장에서 설명한 우선순위 기반의 Pub/Sub 시스템을 개발하기 위해서는 우선순위를 정하기 위한 검색결과의 정확도를 정의해야 하며, 정확도 기반의 우선순위 검색기법을 개발해야 한다.

3.1 정확도 정의

이 논문에서는 구독자의 구독조건이 간격으로 표현되는 경우를 가정하여 정확도를 정의하고자 한다. 간격데이터의 정확도는 간격의 길이를 기반으로 정의될 수 있다. 즉, 특정 값을 포함하는 간격이 둘 이상 존재할 경우에 더 좁은 간격의 데이터가 정확도가 높다고 판단할 수 있다. 취업사이트를 예를 들면 나이를 구인조건으로 표현할 경우에 20~25세의 구인조건과 20~60세의 구인조건은 둘 다 25세의 구직자에게 매칭될 수 있다. 그런데 이 경우에 더 좁은 범위의 구인조건이 구직자에게 더 정확한 검색 결과가 될 수 있다.

3.2 결과 집합 정렬

검색 기법의 개발을 위한 가장 간단한 방법은 최종적으로 얻어진 마커들의 정렬(sorting)을 통해 얻을 수 있다. 그림 5에서 출판조건(Query) 값을 7로 가정할 때 결과 집합 {0, 4, 3}을 얻을 수 있다. 이 결과 집합을 정확도의 순서대로 나열한 정렬된 집합은 (3, 0, 4)로 표현된다. 정렬된 결과 집합에서 출판자가 요구한 수만큼 추출한다. 이 방법은 구현이 간단하나 간격데이터가 증가할수록 검색비용에 문제가 발생한다.

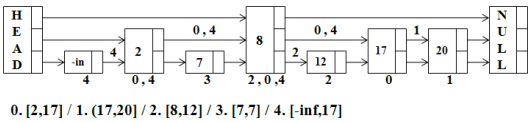

그림 5. IS-Lists의 기본적인 형태

3.3 삽입시간 정렬과 스택을 이용한 역 탐색

결과 집합 정렬 방법은 정렬비용이 검색비용에 더해지는 문제와 출판자가 요구한 수보다 많은 구독조건이 결과 집합에 추가되어 생기는 비용 문제가 있다. 정렬 비용을 삽입 시간에 추가시켜 검색 시간에서 정렬 비용을 빼고 출판자가 요구한 구독조건만을 결과 집합에 추가하는 방법을 제안한다.

첫 번째로, 삽입시간 정렬은 새로운 마커의 추가 시 실행한다. 정렬은 정확도에 의해서 이루어진다. 다음은 삽입시간 정렬의 Pseudo Code다.

```
    addMarkers(newMarker)
1   i = 0;
2   j = the number of markers on the edge;
3   if(there is no marker on edge) then
4       add new Marker to marker[0] on the edge;
5   else if(there are markers more than 0) then
6   begin
7       while(i < j) do
8           if(accuracy of marker[i] on the edge <
9                accuracy of newMarker) then
10               move each marker[i+1] to [j-1]
11               to the next marker[i+2] to [j] and
12               add newMarker to marker[i+1] on edge
13               break;
14          else if(accuracy of marker[i] on the edge
15                   >= accuracy of newMarker) then
16               move each marker[i] to [j-1]
17               to the next marker[i+1] to [j] and
18               add newMarker to marker[i] on edge;
19               break;
20      i = i + 1;
21  end
```

출판자가 요구한 구독조건만을 결과 집합에 추가하기 위해서는 매칭된 마커 중 정확도가 높은 순서대로 검색해야 한다. 그림 6에서 하위 레벨에 있는 마커들이 상위 레벨의 마커들 보다 정확도가 더 높다는 것 을 알 수 있다. 즉, 정확도가 가장 높은 최하위 레벨부터 검색을 시작해야 한다. 하지만 IS-Lists의 검색의 처음 단계는 최상위 레벨에서부터 시작 한다. 이러한 점을 극복하기 위해 자료구조 스택(Stack)을 이용하였다. Skip Lists의 검색 절차에 따라서 주어진 출판조건 값까지 찾아가는 것을 기본으로 한다.

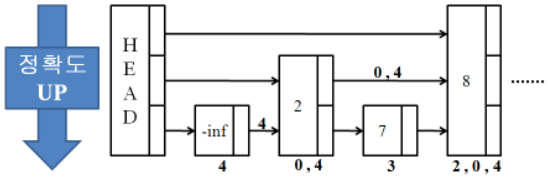

그림 6. 포함조건에 의한 마커간의
정확도 상하관계

여기서 주의 할 점은 기존의 IS-Lists 검색 절차와 다르게 주어진 출판조건을 찾을 때 까지는 구독조건(마커)들을 추출 하지 않는다. 단, 레벨이 다운될 때 구독조건의 추출 대신 해당레벨의 포인터를 가진 노드를 스택에 담는다. 스택을 이용하여 역 순으로 이동하며 출판자가 요구한 구독조건만을 추출한다. 삽입시간에 이미 구독조건들은 정렬이 되었다. 다음은 스택을 이용한 역 탐색의 Pseudo Code를 나타낸다.

```
findMarkers(K, List, userCount)
1  x = header;
2  Stack[maxLevel] = null;
3  i = maxLevel;
4  currentLevel = 0;
5  while i >= 0 and(x is the header or x->key != K)
6  do
7  begin
8      while(x->forward[i] != null and
9          x->forward[i]->key < K) do
10         x = x->forward[i];
11     if(x is not the header and x->key != K) then
12         Stack[currentLevel] = x->forward[i];
13     else if(x is not the header) then
14         Stack[currentLevel] = x->eqMarkers[i];
15     i = i - 1;
16     currentLevel = currentLevel + 1;
17 end
18 i = currentLevel-1;
19 j = 0;
20 while(i >= 0 and userCount != 0) do
21 begin
22     while(j <= the number of markers on the edge
23         in Stack[i] and userCount != 0) do
24         List.add(a marker on the edge in Stack[i]);
25         j = j + 1;
26         userCount = userCount - 1;
27     i = i - 1;
28 end
```

V. 결론 및 향후 과제

이 논문에서는 전통적인 Pub/Sub시스템을 적용한 어플리케이션에서의 문제점에 대하여 논의하였다. 그리고 문제점을 개선하기 위하여 구독조건에 우선순위를 정하고 우선순위의 기반이 되는 정확도에 대하여 정의 하였다. 마지막으로 우선순위에 따라 검색 가능한 검색기법들을 제안하였고 우선순위에 의해 정렬된 구독조건들을 받아 볼 수 있다.

향후 과제로는 현재보다 더욱 정확한 우선순위를 정하기 위해서는 단순하게 간격의 길이가 아닌 다른 방법들이 요구된다. 따라서 정확도에 대한 재정의가 필요하다. 그리고 이 논문의 기법에서 사용되는 색인은 한 가지다. 즉, 구독조건으로 나타낼 수 있는 것이 간격의 길이 하나라는 말이다. 구독조건이 한 가지 인 것 보다 다수 일 때 더욱 효율적인 성능을 기대 할 수 있기 때문에 다양한 색인을 사용하는 시스템 연구가 필요하다.

참고문헌

[1] Kenneth P. Birmanand Thomas A. Joseph, "EXPLOTING VIRTUAL SYNCHRONY IN DISTRIBUTED SYSTEMS" ACM SIGOPS Operating Systems Review, 1987

[2] Ashwin Machanavajjhala, Erik Vee, Minos Garofalakis, Jayavel Shanmugasundaram "Scalable Ranked Publish/Subscribe"

[3] M.Edelsbrunner "Dynamic Data Structures for Orthogonal Intersection Queries"

[4] http://www.dgp.toronto.edu/people/JamesStewart/378notes/22intervals/

[5] Antonin Guttman "R-TREES A Dynamic Index Structure for Spatial Searching", Proceedings of the 1984 ACM SIGMOD international

[6] Eric N.Hanson, Theodore Johnson, " Selection Predicate Indexing for Active Databases using Interval Skip Lists", Information Systems 1996

[7] Eric N.Hanson, Theodore Johnson, "The Interval Skip List: A Data Structure for Finding All Intervals That Overlap a Point", Algorithm and Data Structures, 1991

[8] W Pugh, "Skip Lists: A Probabilistic alternative to balanced trees", Communications of the ACM, 1990