SIF기반 우선순위 검색기법의 설계 및 구현

이은식* · 조대수**

Design and Implementation of Priority Retrieval Technique based on SIF

Eun-Sik Lee* · Dae-Soo Cho**

요 약

전통적인 Pub/Sub(Publish/Subscribe) 시스템에서는 출판자(Pub)가 출판정보(Event)를 중개자(Broker)에게 전달을 하고 등록되어져 있는 구독조건(Subscription)과 매칭유무를 파악한 뒤 매칭된 구독조건들을 등록한 구독자(Sub)에게 출판정보를 전달한다. 이 시스템에서 정보의 전달은 출판자에서 구독자로의 단 방향으로 이루어졌다. 최근 새로운 응용 프로그램에서는 구독조건을 출판자에게도 전달하는 양 방향 정보전달의 필요성이 제기되었다. 따라서 출판자와 구독자들 간의 양방향 정보전달이 가능한 확장된 Pub/Sub 시스템을 제안하고자 한다.

확장된 Pub/Sub 시스템에서는 출판정보에 매칭되는 구독조건이 다수가 있을 수 있으므로 우선순위에 따라 상위 n개만을 출판자에게 전달할 수 있는 기능이 요구된다. 이 논문에서는 구독조건 간 우선순위를 결정하고 정하기 위한 SIF(Specific Interval First)를 정의하고 IS-List(Interval Skip List)를 이용하여 SIF기반의 우선순위 검색기법을 두 가지 방법으로 제안하였다. 성능 평가 결과 집합 정렬 방법은 색인 생성 측면에서 그리고 삽입 시간 정렬 및 스택을 이용한 역 탐색 방법은 검색 시간 측면에서 좋은 성능을 보였다.

ABSTRACT

In traditional Publish/Subscribe system, the first procedure to deliver event from publisher to subscriber is that publisher publishes publisher's event to broker. Next step is that broker checks simple binary notion of matching : an event either matches a subscription or it does not. Lastly, broker delivers the event matched with subscriptions to the corresponding subscribers. In this system, information delivery has been accomplished in one way only. However, current some applications require two way delivery between subscriber and publisher. Therefore, we initiate an extended Publish/Subscribe system that supports two way delivery.

Extended Publish/Subscribe system requires additional functions of delivering subscription to publisher and especially deciding top-n subscriptions using priority because broker might has a number of subscriptions. In this paper, we propose two priority retrieval techniques based on SIF using IS-List with deciding priority among subscriptions and defining SIF(Specific Interval First). The performance measurements show that RSO(resulting set sorting) technique results in better performance in index creation time and ITS&IS(insertion time sorting and inverse search using stack) technique results in better performance in search time.

키워드

출판/구독 시스템, 간격 스킵 리스트, SIF(구체적 간격 우선), 우선순위

Key word

Publish/Subscribe System, Interval Skip List, SIF(Specific Interval First), Priority

* 동서대학교 컴퓨터정보공학부 학부생

접수일자 : 2010. 10. 29

** 동서대학교 컴퓨터정보공학부 부교수 (교신저자, dscho@dongseo.ac.kr)

Ⅰ. 서 론

전통적인 Pub/Sub(Publish/Subscribe) 시스템[1]은 출판자(Pub), 출판정보(Event), 중개자(Broker), 구독자(Sub), 구독조건(Subscription)으로 이루어진다. 이 시스템은 출판자가 출판정보를 중개자에게 전달하면 등록되어진 구독조건들과의 매칭 유무를 확인하고 매칭된 구독조건들을 등록한 구독자에게 전달하는 단 방향 정보전달 시스템이다. 최근 새로운 응용 프로그램에서는 출판정보를 구독하는 구독자의 구독조건들을 출판자에게 전달하는 양 방향의 정보전달의 필요성이 제기되었다[2]. 예를 들어 취업 사이트의 경우 구인을 원하는 회사(구독자)가 요구하는 구인조건(구독조건)을 사이트에 등록해 놓으면 구직을 원하는 사람(출판자)이 사이트에 접속하면 구직자의 정보(출판정보)와 매칭되는 구인조건들을 검색해서 해당 회사로 정보를 전달하게 된다. 즉, 출판자인 구직자로부터 구독자인 구인회사로 정보가 전달된다. 이 경우에 출판자인 구직자 또한 본인의 구직정보와 매칭되는 구인조건들을 전달받기를 원할 수 있으므로, 양 방향 정보전달이 요구된다. 이 논문에서는 출판자와 구독자들 간의 양 방향 정보전달이 가능한 확장된 Pub/Sub 시스템을 제안하고자 한다.

확장된 Pub/Sub 시스템에서 매칭된 구독조건이 다수일 수 있으며 구독조건 간 매칭의 정도는 고려하지 않고 매칭의 유무만을 판단하여 전달을 하는 특징 때문에 출판자가 원하는 n개의 구독조건에 모두 만족하기 어렵다. 여기서 구독조건 간에도 우선순위가 존재할 수 있기 때문에 우선순위에 따라 구독조건들을 검색할 수 있는 방법이 요구된다. 해당 시스템을 취업 사이트나 쇼핑 몰 사이트와 같은 곳에 적용시켰을 때 구독조건 대부분의 경우 간격으로 표현될 수 있다. 따라서 해당기법은 Segment Tree[3], Interval Tree[3][4], R-Tree[5] 등과 같은 간격색인중 하나인 IS-List (Interval Skip List)[6][7]를 기반으로 중개자에서 동작한다.

이 논문에서는 우선순위를 결정하기 위한 SIF(Specific Interval First)를 정의하고 SIF에 기반한 우선순위 검색기법을 제안한다. 가장 일반적인 결과 집합 정렬 방법과 해당 방법의 문제점을 보완한 삽입 시간 정렬 및 스택을 이용한 역 탐색 방법을 성능측정하고 비교분석 한다.

Ⅱ. 관련 연구

2.1 Stabbing 질의 문제

Pub/Sub 시스템에서 중개자에 등록되어진 구독조건들을 출판자가 전달한 출판정보에 매칭된 구독조건들을 빠르게 찾는 것을 Stabbing 질의 문제라고 할 수 있다. Stabbing 질의 문제에서 하나의 질의(Query)는 특정한 값으로 표현되고, 데이터는 일정한 길이를 갖는 간격(Interval)으로 표현된다. 그림 1은 Stabbing 질의 문제의 예를 보이고 있다. A부터 G까지 모두 7개의 간격데이터가 있다. 이때 '44를 포함하는 모든 간격데이터를 찾으시오' 라는 의미의 Stabbing 질의에 대해서 결과로 {A, B, D, G}를 검색하게 된다.

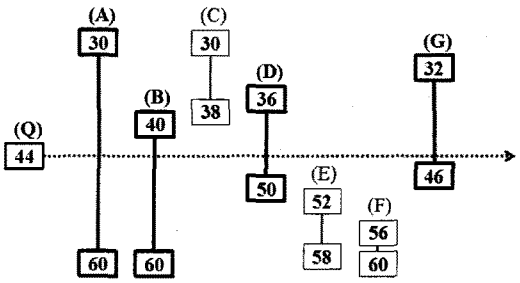

그림 1. Stabbing 질의 문제
Fig. 1 Stabbing Query Problem

2.2 Skip Lists

Skip List[8]는 순차적 탐색을 하는 연결리스트의 단점을 개선시킨 자료구조다. 연결리스트와 달리 하나의 노드가 1개 이상의 포인터를 가진다. 포인터의 개수는 그 노드의 레벨을 의미하며 노드 생성 시 랜덤 값에 의해 확률적으로 정해진다. 새로운 노드가 k레벨을 가질 확률은 다음과 같다.

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

p값이 1/2 일 때는 레벨1인 노드들이 전체의 약 1/2이 되고 레벨2인 노드는 1/4, 레벨3인 노드는 1/8 등 이러한 확률로 계산된다. 그림 2에서 키(Key) 값 44를 검색 할 때 12, 23, 26 키 값을 가진 노드들을 생략(Skip)하

고 있다. 연결리스트로 44를 가진 노드를 검색하기 위해서는 6번의 탐색이 필요하지만 Skip List는 3번에 불과하다.

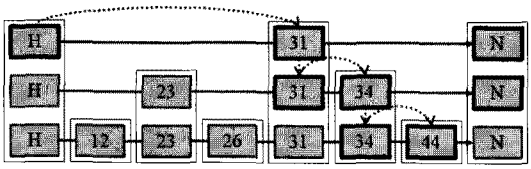

그림 2. Skip List 의 예
Fig. 2 Example of Skip List

2.3 IS-List(Interval Skip List)

IS-List는 Skip List에 간격의 개념을 확장시킨 간격색인이다. IS-List에서 간격데이터는 두 개의 노드로 구성되고 한 노드안의 키 값은 간격데이터의 한 끝점을 의미한다. 하나의 노드는 자신의 레벨만큼의 Forward Edge를 가지며 그 위에 간격을 나타내는 Marker를 표시하게 된다. 예를 들어그림 3에서 0이라는 Marker는 [2, 17]을 나타낸다. 이때 Marker 0은 끝점 중 하나인 2를 가진 노드부터 다른 끝점 중 하나인 17을 가진 노드까지 Marker를 Forward Edge에 표시하게 된다. 이때 가장 최상위 레벨의 Forward Edge에 표시하며 하위 레벨에는 Marker를 표시할 필요가 없다.

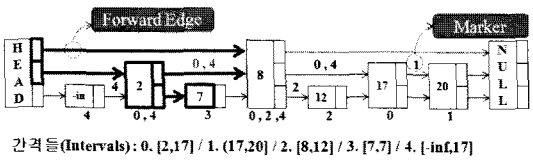

그림 3. IS-List의 예
Fig. 3 Example of IS-List

Ⅲ. SIF 기반 우선순위 검색기법의 설계

1장에서 설명한 확장된 Pub/Sub 시스템을 개발하기 위해서는 우선순위를 정해야 한다. 우선순위를 결정하기 위한 구독조건의 SIF(Specific Interval First)를 정의해야 하며, SIF기반의 새로운 우선순위 검색기법을 개발해야 한다.

3.1 SIF 정의

이 논문에서는 구독조건이 간격으로 표현되는 경우를 가정하여 SIF를 정의하고자 한다. 구독조건의 SIF는 간격의 길이로 정의 할 수 있다. 즉, 특정 값을 포함하는 간격이 둘 이상 존재할 경우 더 좁은 간격의 구독조건이 SIF가 높다고 판단 할 수 있다. 취업사이트에 적용시켜 예를 들면 나이를 구인조건으로 표현할 경우 20~25세의 구인조건과 20~60세의 구인조건은 25세의 구직자의 구직정보에 매칭 될 수 있다. 이 경우 더 좁은 범위인 20~25세의 구인조건이 더 구체적인 검색 결과가 될 수 있다.

3.2 결과 집합 정렬

우선순위를 결정하기 위한 가장 간단한 방법은 최종적으로 얻어진 구독정보들을 SIF에 의해 정렬하는 것이다. 그림 3에서 출판정보(Query) 7에 대한 구독정보의 집합은 {0, 4, 3}이다. SIF의 순서대로 나열한 정렬된 집합은 (3, 0, 4)이다. 여기서 출판자가 요구한 구독정보만을 추출한다. 이 방법은 그림 4와 같은 문제가 발생한다.

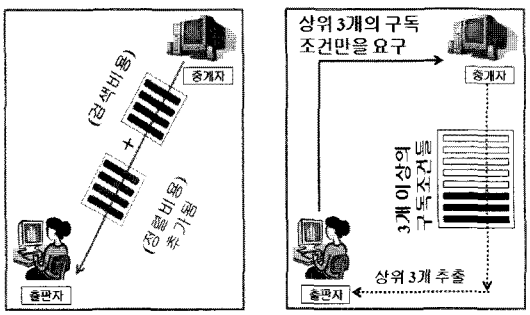

그림 4. 결과 집합 정렬의 문제점
Fig. 4 Problem of Resulting Set Sorting

3.3 삽입 시간 정렬 및 스택을 이용한 역 탐색

결과 집합 정렬 방법에서 정렬비용이 검색비용에 추가되는 문제와 출판자가 요구한 수보다 많은 구독조건들을 모두 검색해야 하는 불필요한 검색비용이 발생한다. 정렬 비용을 삽입시간에 추가시키고 출판자가 요구한 수만큼 검색할 수 있는 삽입 시간 정렬 및 스택을 이용한 역 탐색 방법 을 제안한다. 첫 번째로, 삽입 시간 정렬은 새로운 구독조건의 추가 시 실행한다. 그림 5는 삽입 시간 정렬 부분의 Pseudo Code를 나타낸다.

```
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 <                accuracy of
9  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              >= accuracy
15 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
```

그림 5. 삽입 시간 정렬 알고리즘
Fig. 5 Insertion Time Sorting Algorithm

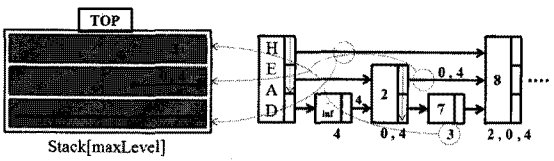

그림 6. 스택을 이용한 역 탐색의 예
Fig. 6 Example of Inverse Search using Stack

다음으로 출판자가 요구한 개수만큼만 검색하기 위해서는 매칭된 구독조건 중 SIF의 간격이 좁은 순서대로 검색해야 한다. 하지만 IS-List의 특징은 상위 레벨의 구독정보가 하위레벨의 구독정보의 간격보다 넓다. 그리고 검색은 최상위 레벨에서 시작한다. 이러한 점을 극복하기 위해 자료구조 스택(Stack)을 이용하였다. 그림 6과 같이 먼저 일반적인 검색절차에 따른다. 여기서 주의 할 점은 기존의 검색방법과 다르게 주어진 키 값을 찾을 때까지 구독조건들을 추출하지 않는다. 단, 해당레벨의 Forward Edge를 스택에 담는다.

```
findMarkers(K, List, userCount)
1
2  x = header;
3  Stack[maxLevel] = null;
4  i = maxLevel;
5  currentLevel = 0;
6  while i >= 0 and(x is the header or x->key != K) 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
```

그림 7. 스택을 이용한 역 탐색 알고리즘
Fig. 7 Inverse Search Algorithm using Stack

최종적으로 스택에서 사용자의 요구 개수만큼 추출할 수 있다. 그림 7은 스택을 이용한 역 탐색부분의 Pseudo Code를 나타낸다.

Ⅳ. 성능 평가

이 장에서는 결과 집합 정렬 방법과 삽입 시간 정렬 및 스택을 이용한 역 탐색 방법의 성능을 평가한다. 실험은 해당 기법을 취업사이트에 적용 시켰을 때를 가정하였다. 실제로 취업사이트에서 간격데이터 범위의 대부분은 취업에 관심이 있는 20세에서 30세로 볼 수 있다. 따라서 20~30세가 포함되는 간격데이터의 분포가 80%인 조건과 간격데이터의 분포가 임의로 만들어진 조건에서 성능평가를 하였다.

간격의 범위는 사람의 나이로 표현 될 수 있는 1부터 99까지로 제한하였다. 질의 셋(QS) = {q=10, 20, 30, 40, 50, 60, 70, 80, 90}으로 9개의 stabbing 질의로 구성하였다. 데이터 셋은 간격이 랜덤하게 분포되어 있는 데이터 셋(DS1)과 일정한 구간(이 논문에서는 20~30)을 포함하는 간격의 분포가 전체 80%가 되는 데이터 셋(DS2)에 구성하였다. 각 데이터 셋에 포함되는 데이터(간격)의 개수는 1000개부터 5000개까지 500개씩 증가시켜가며 성능을 평가하였다.

표 1과 표 2는 DS1과 DS2의 평균 색인 생성 시간과 평균 검색 시간을 나타낸다. A는 결과 집합 정렬을 나타내고 B는 삽입 시간 정렬 및 스택을 이용한 역 탐색을 나타낸다. 각 시간은 간격개수별로 QS를 구성하는 질의들을 수행하여 평균을 구한 값이다.

그림 8은 A에 대한 B의 색인 생성 시간의 비율을 보이고 있다. B에 비해 A의 생성시간이 DS1에서는 18 ~ 174배 까지 증가하며 DS2에서는 31 ~ 280배 까지 증가한다. 또한, 임의의 분포를 가지는 DS1보다 일정 구간에 치우친 분포인 DS2에서 A의 색인 생성 시간이 B에 비해 상대적으로 빨라지는 것을 알 수 있다. B는 정렬 비용이 색인 생성 시간에 포함되기 때문에 일정 구간에 치우친 분포일 경우 색인 생성 시 다른 비용이 들지 않는 A와 차이가 나게 된다.

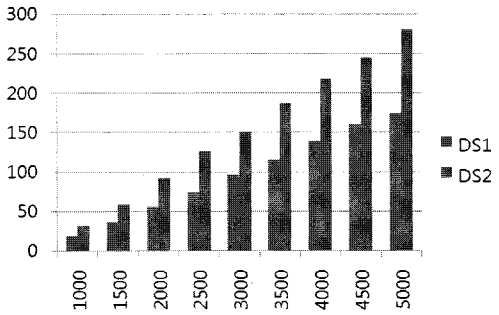

그림 8. A에 대한 B의 색인 생성 시간 비율
Fig. 8 Ratio of Index Creation Time of B over A

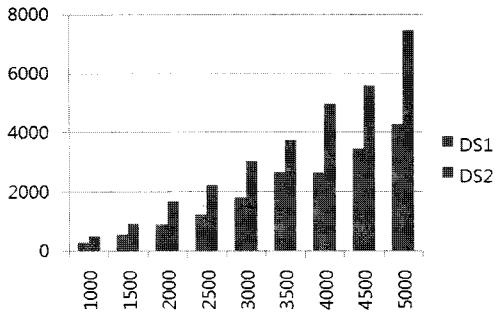

그림 9. B에 대한 A의 검색 시간 비율
Fig. 9 Ratio of Search Time of A over B

표 1. DS1의 평균 색인 생성 시간과 평균 검색 시간
Table. 1 Average Index Creation Time and Average Search Time of DS1

간격개수		1000	1500	2000	2500	3000	3500	4000	4500	5000
생성 (초)	A	0.14	0.26	0.42	0.59	0.81	1.07	1.37	1.77	2.16
	B	2.58	9.30	23.2	43.6	77.2	123.69	190.45	282.91	376.20
검색 (초)	A	0.003086	0.006050	0.010755	0.015808	0.023439	0.039803	0.039661	0.051672	0.063990
	B	0.000011	0.000011	0.000012	0.000013	0.000013	0.000015	0.000015	0.000015	0.000015

표 2. DS2의 평균 색인 생성 시간과 평균 검색 시간
Table. 2 Average Index Creation Time and Average Search Time of DS2

간격개수		1000	1500	2000	2500	3000	3500	4000	4500	5000
생성 (초)	A	0.22	0.42	0.69	1.03	1.42	1.91	2.43	3.12	3.71
	B	6.9	24.61	63.16	129.97	213.77	356.64	529.38	762.22	1040.48
검색 (초)	A	0.005425	0.010954	0.019948	0.031104	0.045167	0.059534	0.079188	0.096914	0.126588
	B	0.000011	0.000012	0.000012	0.000014	0.000015	0.000015	0.000016	0.000017	0.000017

그림 9는 B에 대한 A의 검색 시간을 나타내고 있다. B에 비해 A의 검색 시간이 DS1에서는 280 ~ 4266배 까지 증가하고 DS2에서는 493 ~ 7446배 까지 증가한다. 또한, DS2에서는 일정 구간에 치우친 분포임에도 불구하고 검색시간이 임의의 분포를 가지는 DS1일 때 보다 B의 검색시간이 A보다 상대적으로 더 빨라지는 것을 알 수 있다. A에서는 일정 구간에 치우친 분포일 경우 질의와 겹치는 간격의 개수가 증가함에 따라 최종적으로 얻어오는 Marker의 개수 또한 증가하기 때문에 최종적으로 정렬하는 비용에 오버헤드가 생긴다.

그에 비해 B는 간격의 개수나 Marker의 개수에 영향을 받지 않기 때문에 성능에 큰 문제가 없다. 이번 실험을 통해 결과 집합 정렬의 경우 색인 생성 측면에서 우수함을 보이기 때문에 색인 구조가 자주 바뀌는 동적 환경에 적합하며 삽입 시간 정렬 및 스택을 이용한 역 탐색의 경우 탐색 시간에서 우수함을 보이므로 색인 구조가 자주 바뀌지 않는 정적 환경에 적합할 것이다.

V. 결 론

이 논문에서는 전통적인 Pub/Sub 시스템의 확장된 시스템을 제안하고 그 문제점에 대해 논의 하였다. 문제점을 해결하기 위한 우선순위 검색기법을 제안하고 우선순위를 정하기 위한 SIF를 정의 하였다. 이 검색 기법을 통해 출판자 또한 원하는 개수의 구독조건을 받아 볼 수 있게 되었다.

성능 측정에서는 결과 집합 정렬 방법과 삽입 시간 정렬 및 스택을 이용한 역 탐색 방법 두 가지를 비교 분석하였다. 색인 생성 시간에서는 결과 집합 정렬이 뛰어난 성능을 보였고 검색 시간에서는 삽입 시간 정렬 및 스택을 이용한 역 탐색이 뛰어난 성능을 보였다.

향후 과제로는 현재 단순하게 간격의 길이로만 제한하고 있는 SIF의 정의를 다른 방면에서 찾아볼 필요가 있다. 그리고 IS-List 색인 한 가지만이 아닌 다양한 색인을 연계하여 실제 시스템에 적용해 볼 수 있는 연구가 필요하다.

참고문헌

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

[2] Ashwin Machanvaijhala, Erik Vee, Minos Garofalakis, Javavel Shamugasundaram "Scalable Ranked Publish/Subscribe"

[3] M.Edlsbrunner "Dynamic Data Structures for Orthogonal Intersection Queris"

[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 Structures 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

저자소개

이은식(Eun-Sik Lee)

동서대학교 컴퓨터정보공학부

※관심분야 : Pub/Sub 시스템, 간격 질의처리 및 색인, Stabbing 질의처리, Stream Data 처리

조대수(Dae-Soo Cho)

한국해양정보통신학회논문지
제13권 제8호 참조