A Fast Matching Algorithm for Content-Based Publish/Subscribe Systems

Tao Xue and Qi Jia

College of Computer Science, Xi'an Polytechnic University, Xi'an, Shaanxi, China
xthappy@gmail.com, jiaqi19861215@163.com

Abstract. How to efficiently match high volumes of events against large numbers of subscriptions is a key issue for large-scale content-based publish/subscribe systems. In this paper we present an efficient and applied matching algorithm that uses multi-dimensional indexing mechanism to speed up constraints query and exploits the covering relations between constraints to reduce unnecessary matching. Experiments show that our algorithm is significantly more efficient and scalable than other common used matching algorithms.

Keywords: Publish-subscribe, Content-based matching, Multi-dimensional indexing.

1 Introduction

Currently, the content-based publish/subscribe system is a hot research area. In this system the event is no longer dependent on an external standard (such as channel, theme, etc.) to classify, but the content in accordance with the event itself. Subscribers subscribe to the event according to the contents of the event, not to the subject of pre-defined themes by the system. When designing efficient content-based publish/subscribe systems, one of the key issues to be resolved is what strategy should be used to efficiently match large numbers of events and subscriptions.

IBM's Gryphon [1] system uses a parallel search tree to solve the matching between the event and subscription. Each event agent has a complete parallel search tree, and determines the forwarding direction of the incident by means of the tree. But the algorithm can only deal with equal determination. Siena [2][3] is a event notification service in the WAN, which uses the covering relation of the subscriptions and merging subscriptions to reduce the matching frequency, thus reducing the matching complexity. Yan and Garcia-Molina [4] describe the way of solving the matching of the keywords and the document object applying the counting and tree model algorithm, and achieved good results. Gough and Smith [5] proposed a match based on automata theory, it transfers all the constraints relying on a property to a finite state. It's been proved theoretically feasible, but not given a specific algorithm.

2 Subscription Language and Event Model

For content-based publish/subscribe systems, on the one hand sufficient expression languages must be provided so that users can describe the interesting events conveniently, on the other hand the cost that it takes to match the event and subscription should be balanced. As a compromise, we use a subset of SQL language of relational database as the subscription language.

- Event

Event consists of a set of attributes {A1, A2, ... An}, and each attribute is a triple (data type, attribute name, value). Events can be expressed as: Event = $\cup$ attribute.

- Constraint

Constraint is a stateless Boolean expression. It's expressed as quad (type, attribute, operator, value).

Definition 1. If attribute A ($type_a$, $name_a$, $value_a$) of the event matches against a constraint C ($type_c$, $attribute_c$, op_c, $value_c$), the condition must be: ($type_a = type_c$) $\wedge$ ($name_a = attribute_c$) $\wedge$ { op_c ($value_a$, $value_c$) = true}, denoted by $A \in_{match} C$.

Definition 2. Given two constraints c1 and c2, we say that c2 covers c1, denoted by $c1 \subseteq c2$, namely: ($\forall$ a: $a \in_{match} c1 \rightarrow a \in_{match} c2) \Rightarrow c1 \subseteq c2$.

- Subscription

Subscription is composed of a constraints set {C1, C2, ... Cn}. The relation between constraints is the conjunction relations, expressed as Subscription = $\cup$ constraint.

Definition 3. For each constraint c in the subscription S, there is at least one attribute a in the event E, making $a \in_{match} c$, then we affirm that the event E and subscription S match, namely: $\forall c \in S, \exists\ a \in E \wedge a \in_{match} c \Rightarrow E \in_{match} S$.

3 Matching Algorithm

- Data structure

The main data structure is shown in Figure 1. First, we break all the subscriptions into the constraint sets, remove duplicate constraints. Then these constraints are organized into a multidimensional indexing structure, in accordance with its data type, attribute name, comparison operators. Index is made level by level, and the final index entry comes into a list of constraints.

To maintain the subordination relation of the constraints and subscriptions, a constraint-subs link is taken, where each constraint points to a list that stores all the subscriptions identification that contains the constraint.

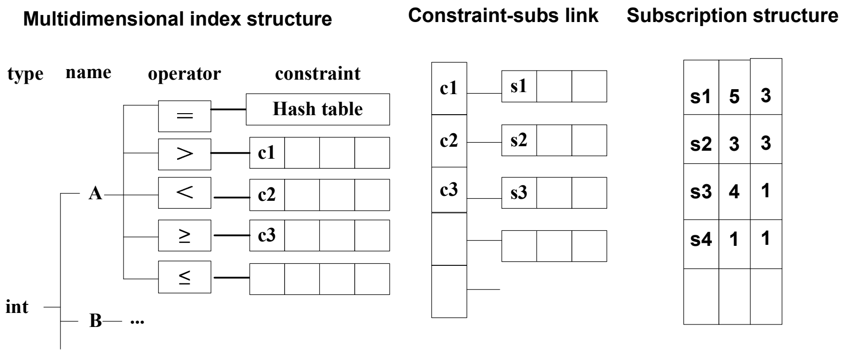


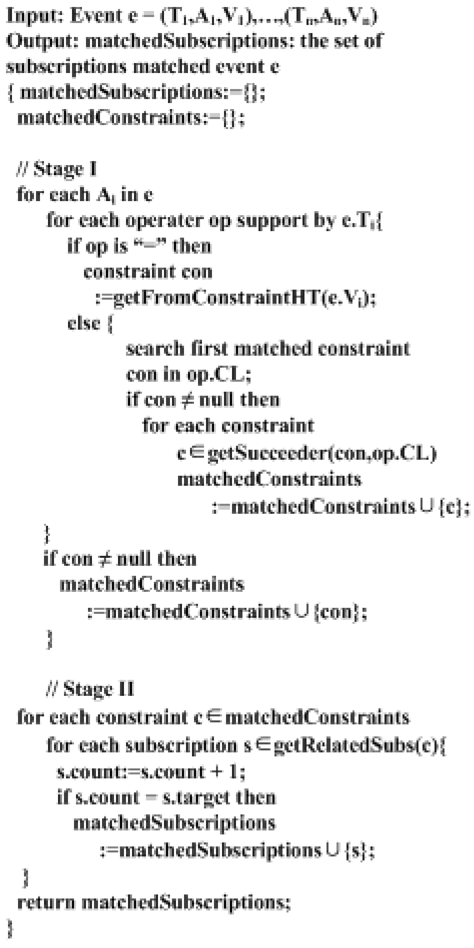
Fig. 1. Data Structure of index count matching algorithm

- Algorithm steps

The first stage is to find matching constraint for each attribute of the event according to their types and supported comparison operators; while the comparison operator is "=", do the hash table lookup directly, and for other comparison operators, get the first match of the constraints, thanks to the coverage relationship between constraints, its subsequent binding will also meet the match.

In the second phase, for each constraint has been successfully matched, the subscriptions belongs to it is searched. Then these subscriptions' corresponding count value will plus 1. If the subscription's count value is equal to the target value, the subscription matches.

Figure 2 shows the steps of the algorithm in detail.



```
Input: Event e = (T1,A1,V1),...,(Tm,An,Vn)
Output: matchedSubscriptions: the set of
subscriptions matched event e
{ matchedSubscriptions:={};
  matchedConstraints:={};

  // Stage I
  for each Ai in e
       for each operater op support by e.Ti{
          if op is "=" then
             constraint con
               :=getFromConstraintHT(e.Vi);
          else {
                  search first matched constraint
                  con in op.CL;
                  if con ≠ null then
                     for each constraint
                          c ∈ getSucceeder(con,op.CL)
                          matchedConstraints
                              :=matchedConstraints ∪ {c};
       }
       if con ≠ null then
          matchedConstraints
              :=matchedConstraints ∪ {con};
       }

       // Stage II
  for each constraint c ∈ matchedConstraints
       for each subscription s ∈ getRelatedSubs(c){
         s.count:=s.count + 1;
         if s.count = s.target then
            matchedSubscriptions
                :=matchedSubscriptions ∪ {s};
   }
  return matchedSubscriptions;
}
```

Fig. 2. Index count matching algorithm

4 Experiment and Evaluation

We implement the sequence comparison method, the traditional counting method and the proposed count matching algorithm with multi-dimensional indexing and constraints covering characteristics. The prototype system includes a data generator, which automatically generates appropriate events and subscriptions according to the configuration.

Experiment is conducted on the PC of a Pentium IV 3.0GHz processor, 2G memory, Windows2003 operating system. Each data point of the results is the average value obtained by running the same parameters 10 times.

Due to space limit, we only illustrate the number of subscribers' effects on the algorithm. The experiment result is shown in Figure 3.

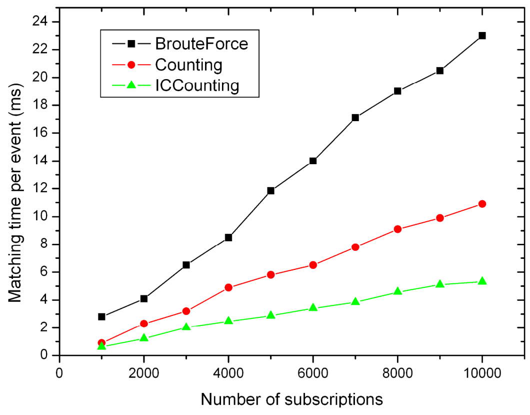


Fig. 3. The number of subscribers' effects on the algorithm

From the experimental results, it can be seen that linear comparison method has the least efficiency, and due to the characteristic of multiple index and constraints coverage, our algorithm is more efficient than the traditional counting method. In addition, as the number of subscribers increases, the efficiency of the algorithm declines much less than the linear comparison method and the traditional counting method, which indicates that its scalability is very good and it's ideal to be used in large-scale systems.

5 Summaries

In this paper, an efficient matching algorithm based on subscription language and event model is proposed, which uses multi-level indexing for high-speed matching of the event and related constraints and reduces the number of matching by the use of the cover relation of the constraints. We have implemented the algorithm and tested it with a variety of configurations, the experimental results show that the algorithm has better performance and strong scalability. Our future work is to integrate the algorithms into our prototype content-based publish subscribe system, and do further research on content-based routing algorithm.

Acknowledgment. The research work is supported by Foundation of Xi'an Polytechnic University PHD Scientific Research (Grant No.BS0725).

References

[1] Aguilera, M., Strom, R., Sturman, D., Astley, M., Chandra, T.: Matching events in a content-based subscription system. In: PODC: 18th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (1999)

[2] Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems 19(3), 332–383 (2001)

[3] Carzaniga, A., Wolf, A.L.: Forwarding in a Content-Based Network. In: Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, pp. 163–174 (August 2003)

[4] Yan, T.W., Garcia-Molina, H.: The SIFT information dissemination system. ACM Transactions on Database Systems 24(4), 529–565 (1999)

[5] Gough, K., Smith, G.: Efficient recognition of events in distributed systems. In: Proceedings of 18th Australasian Computer Science Conference (ACSC) (February 1995)