Knowledge-Based Systems 000 (2016) 1–11

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

An efficient subscription index for publication matching in the cloud

Zongmin Cui[a,*], Zongda Wu[b,*], Caixue Zhou[a], Guangyong Gao[a], Jing Yu[a], Zhiqiang Zhao[a], Bin Wu[c]

[a] School of Information Science and Technology, Jiujiang University, Jiujiang, Jiangxi, China
[b] Oujiang College, Wenzhou University, Wenzhou, Zhejiang, China
[c] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei, China

ARTICLE INFO

Article history:
Received 14 January 2016
Revised 10 July 2016
Accepted 11 July 2016
Available online xxx

Keywords:
Publish/subscribe
Matching
Predicate
Index

ABSTRACT

Publish/subscribe has been successfully used in a variety of information dissemination applications. However, in a cloud computing environment, the enormous amount of information results in a very high requirement for the computing performance of a publish/subscribe method. In this paper, we propose an efficient index called Enindex for publish/subscribe matching. First, we group all the subscriptions submitted by subscribers, based on the key attributes (i.e., the most frequent attributes occurring in the subscriptions). Second, we group all the predicates contained in the subscriptions, according to three basic operators: $\geq$ (greater),=(equal), and $\leq$ (less), so as to remove the repeated predicates, and thus reduce the memory overhead. Finally, we propose an effective index structure to combine the grouped subscriptions together with the grouped predicates. Enindex not only has a small memory overhead, but also can support efficient publish/subscribe matching and online subscription updating. We conduct extensive experiments on synthetic datasets, and the experimental results demonstrate the superiority of the Enindex over state-of-the-art methods in terms of memory overhead and computing efficiency.

1. Introduction

Today, publish/subscribe (or P/S for short) systems are becoming increasingly popular and important, which have been successfully applied in a variety of information dissemination applications such as online advertising [17], stock market [21], e-commerce [30] and social network [31]. A P/S system includes two roles: data publisher and data subscriber. A publisher publishes data (such as files, pictures and videos) in the form of publications. A publication contains a group of predicates to describe the characteristics of its associated data. A subscriber online subscribes the data published by publishers. A subscription also contains a group of predicates to describe what data the subscriber is interested in. The interconnection between the two roles can be achieved by a client/server model [15] or a distributed network [8,24]. If a publication matches a subscription, i.e., their predicates match each other, the P/S system has to ensure a timely delivery of the data associated with the publication to the subscriber.

Recently, P/S systems have began to be widely used to cloud computing environments [1,12]. However, the enormous amount of information on the cloud raises many new challenges. In cloud computing, there exist a large number of subscriptions and publications, resulting in a high performance requirement for a P/S system in terms of memory overhead and computing efficiency. In addition, the online arrivals of subscriptions also require the online updating of subscriptions. P/S matching has been extensively studied in the last one decade; however, most of existing P/S matching methods are not designed for cloud computing. As a result, they cannot satisfy the above requirements, thus difficult to be applied to cloud computing environments immediately.

For example, an index method called K-index is proposed in [26], where a predicate contained in a subscription often needs to be rewritten into a group of equality predicates, consequently, greatly increasing the number of predicates and the memory overhead, and in turn, reducing the efficiency of P/S matching. The BE-tree [11] is a state-of-the-art method, where the number of tree nodes would increase with the increasing of predicate attributes. As the number of attributes increases, a BE-tree would generate a large number of tree nodes, incurring an inefficient performance on P/S matching. Recently, a two-level index method called Opindex is proposed in [30], which performs better than both K-index and BE-tree in terms of computing efficiency. However, in Opindex,

* Corresponding author.
E-mail addresses: cuizm01@gmail.com (Z. Cui), zongda1983@163.com (Z. Wu), charlesjjjx@126.com (C. Zhou), gaoguangyong@163.com (G. Gao), yujingellemma@gmail.com (J. Yu), zq_zhao@hust.edu.cn (Z. Zhao), wubincs@gmail.com (B. Wu).

http://dx.doi.org/10.1016/j.knosys.2016.07.017
0950-7051/

there still exist many repeated predicates, making the index structure relatively complex, i.e., the computing cost of its P/S matching needs to be further optimized. In short, Opindex still cannot well satisfy the performance requirements in cloud computing. In [10], a static index method on a road network is proposed, which can only support static P/S matching without taking into consideration the online updating problem of subscriptions.

In order to meet the performance requirements of cloud computing for memory overhead and computing efficiency, in this paper, we propose an efficient subscription index called Enindex for P/S matching. First, we group the subscriptions based on the key attributes (i.e., the most frequent attributes occurring in the subscriptions). Second, we group the predicates in the subscriptions, according to three basic relational operators ($\geq$, $=$, $\leq$), to remove repeated predicates and as a result improve the memory overhead. Finally, we build connections from the grouped subscriptions to the grouped predicates, consequently, generating the Enindex structure. Based on the index structure, P/S matching achieves better computing performance with much smaller memory overhead. Specifically, Enindex has the following four features: (1) it minimizes the computing cost of P/S matching; (2) it removes the repeated predicates to minimize the memory overhead; (3) it divides normal P/S matching into disjunctive matching and conjunctive matching, thereby improving the computing efficiency of predicate judgement operations; and (4) it provides an effective update algorithm to online update the subscriptions. In addition, we conduct experimental evaluations to compare Enindex against state-of-the-art methods (K-index, BE-tree and Opindex) in terms of memory overhead, index construction, subscription update and computing efficiency.

The rest of this paper is organized as follows. In Section 2, we present the problem statement of P/S matching in cloud computing. In Section 3, we describe the index structure of Enindex. In Section 4, we discuss the improvement of the Enindex. In Section 5, we compare the Enindex with existing methods by experiments. In Sections 6 and 7, we review the related works and conclude this paper.

2. Problem statement

In a P/S system, when subscribing data, a subscriber needs to use a boolean expression that consists of predicates to describe the subscription; when publishing data, a publisher also needs to use a boolean expression to describe the publication associated with the data. Finally, the P/S system performs P/S matching between all the subscriptions and each publication, and then timely delivers the data associated with each matched publication to the related subscribers. In this section, we present the system model that we use, and then formulate the problem that we study.

2.1. System model

To improve the computing performance of P/S matching, we propose an efficient subscription index method called Enindex. The system model of Enindex is presented in Fig. 1, which consists of six modules (1)–(6). As shown in Fig. 1, Enindex includes two roles: subscribers and publishers, whose processing flows are described as follows.

- **Subscriber**. First, in the module (1), a subscriber submits subscriptions to a P/S system by using predicates to describe publications interested him. Second, the module (2) selects a small number of key attributes from all the subscriptions, and then based on them, groups the subscriptions. Third, the module (3) groups the subscription predicates according to relational operators ($\geq$, $=$, $\leq$), where the repeated predicates are removed so as to reduce the memory overhead. Finally, the module (4) constructs the index structure by adding pointers from the grouped subscriptions to the grouped predicates.
- **Publisher**. In the module (5), a publisher publishes data in the form of publications. A publication contains a group of equality predicates used to describe the characteristics of its associated data.
- **P/S matching**. The module (6) performs matching operations between all the subscriptions and each new publication, and timely delivers the data associated with the publication to the subscribers of the matched subscriptions.

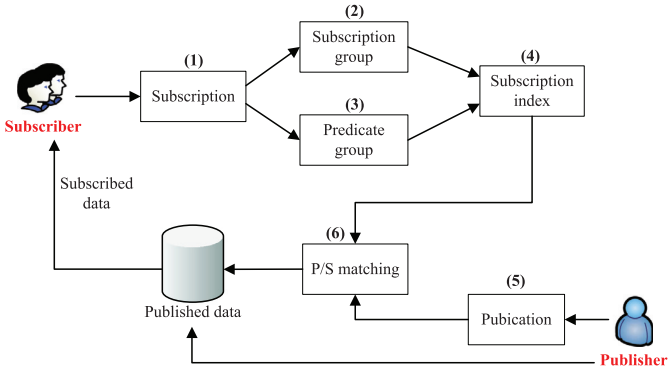

Fig. 1. The system model of Enindex.

Table 1
A running example of subscriptions.

s_1	$A = 2 \wedge B = 3$
s_2	$A \leq 8 \wedge C = 6 \wedge E \geq 2$
s_3	$B \leq 4 \wedge C = 6 \wedge E \in [2, 9]$
s_4	$B = 3$
s_5	$D \geq 8 \wedge E \leq 9$
s_6	$B = 3 \wedge C \leq 4 \wedge D \geq 6$

2.2. Publication matching

A predicate p is the most basic unit of a subscription, which includes an attribute $p.attr$, an operator $p.oper$ and an operand $p.opnd$. It can determine a boolean value (0 or 1) for any value x of the attribute $p.attr$, denoted by $p(x)$, which indicates whether or not the predicate constraint of p is satisfied by x. For example, given a predicate p: $A \geq 5$, we have $p.attr = A$, $p.oper = \geq$, $p.opnd = 5$, $p(4) = 0$ and $p(5) = 1$. Here, $p.oper$ can be all the relational operators ($>$, $\geq$, $=$, $\neq$, $<$, $\leq$). However, to simplify the presentation, we below ignore the relational operators ($>$, $\neq$, $<$) due to their similarity to ($\geq$, $=$, $\leq$). In addition, a subscription is a combination of predicates in either conjunctive normal form (CNF) or disjunctive normal form (DNF). To simplify the presentation, we assume that a subscription is represented in CNF (DNF will be discussed in Section 4). Now, a subscription $\boldsymbol{s}$ that consists of n predicates can be represented as $\boldsymbol{s} : (p_1 \wedge p_2 \wedge \ldots \wedge p_n)$.

Example 1. A set of subscriptions $\mathbb{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_6\}$ is shown in Table 1, which will be used as the running example in the rest of this paper.

A publication supplied by a publisher is also a combination of predicates. However, each predicate in a publication includes only an equality operator. Hence, a publication $\boldsymbol{b}$ consisting of m equality predicates can defined as $\boldsymbol{b} : (q_1 \wedge q_2 \wedge \ldots \wedge q_m)$. A publication also takes along with data, so all of its equality predicates are actually used to describe the characteristics of its data. For example, given a product data $\boldsymbol{r}$ about iPhone and its associated publication

Please cite this article as: Z. Cui et al., An efficient subscription index for publication matching in the cloud, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2016.07.017