Monitoring Web-Based Subscription Services

Kun-Lung Wu and Philip S. Yu
IBM T.J. Watson Research Center
{*klwu, psyu*}*@us.ibm.com*

Abstract

Web-based subscription services are becoming important as more businesses migrate to the Web. Monitoring subscriptions is to determine, for a given event, the subset of all subscriptions whose predicates match the event. Usually, subscriptions contain both equality and non-equality clauses. The general approach to this problem is to identify a "small" candidate set of subscriptions for each occurring event and then determine the matching subscriptions from the candidate set. Previous work has focused on using the equality predicate clauses to identify the candidate set. We found that completely ignoring non-equality clauses can result in a much larger candidate set. However, the effect of using non-equality predicate clauses varies in the sense that not every one of them can reduce the size of the candidate set and there is a diminishing marginal return if too many non-equality predicate clauses are included. We provide a method to map each subscription into a data point in a multiple dimensional space based on its predicate, including the non-equality clauses, and devise an adaptive multiple key hashing (AMKH) method to judiciously select an effective subset of the non-equality clauses to be included in identifying the candidate set. AMKH further provides a controlling mechanism to limit the hash range of a non-equality clause, hence reducing the size of the candidate set. Simulation studies show that AMKH performs well under various conditions.

Keywords: Subscription Monitoring, Event Matching, Web Services, Adaptive Multiple Key Hashing.

1 Introduction

The Web provides a nice platform to allow subscribers from all over the world to subscribe to various services. Emerging subscription services include stock alerts, news event alerts, travel alerts on fare discounts or schedule changes, security violation alerts and bio-surveillance alerts. Depending on the urgency, alerts can be sent to the users via e-mails or cell-phones. Because of the potential large number of subscriptions, monitoring them can be rather challenging.

A subscription is defined by a predicate that is a conjunction of one or more predicate clauses. Each clause is a triple consisting of an attribute, an operator ($=$, $<$, $>$, $\leq$, $\geq$, $\neq$), and a value. For example, in stock alert services, (symbol, =, "IBM")$\wedge$(price, <, 100) is a subscription. An event is a conjunction of pairs, where each pair consists of an attribute and a value. For instance, (symbol, "CSCO")$\wedge$(price, 14) is an event. A subscription s matches an event e if every clause in s is matched by a pair in e.

We introduce an adaptive multiple key hashing (AMKH) approach to monitoring a large number of subscriptions containing non-equality predicate clauses. The idea is to represent a subscription as a data point in a multidimensional space and then use a multiple key hashing to maintain subscriptions and perform event matching. For a given attribute, a unique non-zero ID is assigned to each individual clause. Such clause IDs become the coordinate values of a subscription. Zero is assumed for the attribute where no predicate clause is specified in a subscription. Upon the arrival of an event, event values are used to identify all matched IDs for all individual attributes. These matched IDs are then used to construct *candidate coordinates*, which in turn are used as keys to find matched subscriptions from the hash table.

With non-equality predicate clauses, the number of candidate coordinates can potentially explode because the number of matched IDs for a non-equality attribute can be large. Based on subscription schemas and predicate clause selectivities, AMKH judiciously determines if any non-equality attribute should be included in its hash computation. If yes, how many and which ones should be chosen. Simulations are conducted to study the performance of AMKH. The results show that (1) a small number of non-equality clauses can be effectively included by AMKH in the hash computation and (2) the attributes whose overall non-equality predicate clauses are most selective should be chosen for inclusion by AMKH.

For a small number of subscriptions, event matching can be directly implemented by a sequential search or by the trigger functionality in most database systems. However, these approaches do not scale to hundreds of thousands or millions of subscriptions. Some of the existing event matching algorithms tackle the scalability issue, but most of them do not explicitly deal with non-equality clauses [1, 3, 2]. R-trees and many of its variants used in spatial data structures can be used to deal with non-equality predicates. However, as pointed out in [4], they are not suitable for event matching because R-trees would represent subscriptions as regions and would have extensive overlapping "slices" in event matching. In contrast, AMKH represent each subscription as a point instead of a region.

2 Adaptive multiple key hashing

For simplicity, we assume there are two types of attributes: *equality-only* and *general*. Equality-only attributes, such as "stock symbol", only permits equality predicate clauses. General attributes allows both equality and non-equality clauses, such as $>$, $<$, $\geq$, $\leq$ and others.

If subscriptions contain solely equality predicates, event matching is rather simple. The coordinate values of a subscription can be used as keys to hash into a multiple key hash table. Upon the arrival of an event, the event values can be used to find the matched IDs from each attribute. These matched IDs become a candidate coordinate, which is then used to search from the hash table to see if any subscription matches it.

With non-equality clauses, however, event matching can be rather complex. The number of candidate coordinates can grow fast because the number of matched IDs can be large. For example, an event value pair $(A_1, 100)$ matches all predicate clause $(A_1, >, v)$, where $v < 100$. If there are many such clauses specified on A_1, e.g., $(A_1, >, 5)$, $(A_1, >, 14)$, $\cdots$, $(A_1, >, 99)$, then the matched IDs for A_1 can be large. As a result, the candidate coordinates can become large.

Each candidate coordinate represents a potential hash location where matched subscriptions may reside. A large number of candidate coordinates mean a large number of subscriptions need to be checked. Hence, the inclusion of any general attribute in the hash computation must be done judiciously.

A multiple key hashing (MKH) scheme can be used to store subscriptions. There are many implementations for MKH (see pages 557-564 in [5]). We chose a flexible MKH for our consideration. A subscription with coordinate $(C_1, C_2, \cdots, C_N)$, where C_i is the predicate clause ID involving attribute A_i and N is the total number of attributes, is mapped to a hash value,

$$(w_1 C_1 + w_2 C_2 + \cdots + w_N C_N) \bmod M, \tag{1}$$

where M is any convenient number, and w_i is a "random" number. Collisions are handled by linked lists.

Because of the potential large number of candidate coordinates in the presence of general attributes, the hash computation must be done carefully. *Adaptive* multiple key hashing (AMKH) is a modification of the MKH in equation (1). It has two important changes. First, clause IDs from general attributes may not be included in the hash computation. It depends on the subscription predicate schemas and predicate value distributions. Second, the hash function of AMKH is $\sum_{i=1}^{N} h_i(C_i) \bmod M$, where $h_i(C_i)$ is the hash function for C_i and is defined as follows:

$$h_i(C_i) = \begin{cases} w_i(C_i \bmod B) & \text{if } C_i \text{ is a non-equality clause ID and } A_i \text{ is hashed} \\ w_i C_i & \text{if } C_i \text{ is an equality clause ID and } A_i \text{ is hashed} \\ 0 & \text{if } A_i \text{ is not hashed,} \end{cases} \tag{2}$$

where $B \geq 2$ is a constant. In AMKH, an equality clause ID is hashed differently from a non-equality clause ID. As such, AMKH is adaptive to the operator of a general attribute, in addition to subscription schemas.

The goal of AMKH is to control the number of subscriptions examined. AMKH uses two parameters (G, B) to achieve this goal. Basically, G is the number of general attributes included in the hash computation. Our studies showed that if there are more than 3 clauses from equality-only attributes in the subscription schemas, then G should be 0. Otherwise, G should be 1 in most cases. However, G can be 2 if there are only one or two clauses from a small set of equality-only attributes. B is used to reduce the number of hash entries for a general attribute. It hashes many candidate coordinates with different non-equality IDs into a small number of locations. Our studies showed that B should be 2 if the selectivity of the general attribute is low, otherwise B should be larger than any clause ID if the selectivity is high.

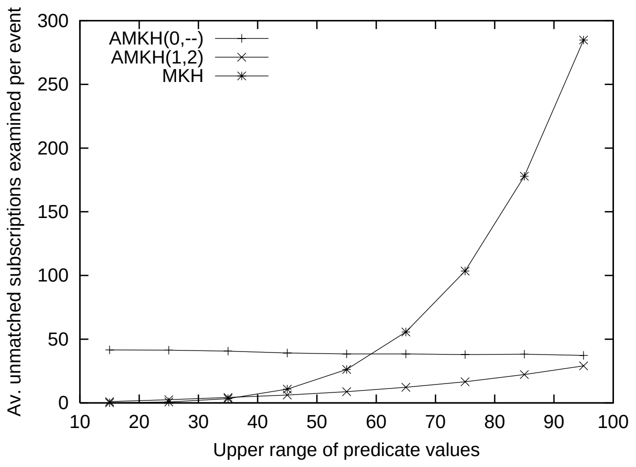

Figure 1: Comparisons of AMKH and MKH.

3 Performance Evaluation

Simulations were conducted to study the performance of AMKH. In the simulations, there were 10 equality-only attributes and 10 general attributes. For each clause from the general attribute, there is a 50% chance that the operator is $=$ and the rest either $>$ or $<$. Each subscription contains 3 clauses, one from a fixed equality-only attribute, one chosen randomly from the rest of 9 equality-only attributes, and one from a fixed general attribute. The predicate values were chosen uniformly from a range of $[6, x]$, where x changes from 15 to 95. Event values were uniformly chosen from [6, 95]. Fig. 1 shows the performance comparisons of AMKH and MKH in terms of total number of unmatched subscriptions examined per event. We demonstrate that a non-equality attribute can be effectively included in hash computation by AMKH(1,2) ($G = 1$ and $B = 2$) to improve performance over MKH and AMKH(0,-), which is equivalent to using equality-only clauses in hash computation. Moreover, both MKH and AMKH(1,2) perform rather well if the upper range of predicate values is small. Other performance studies show that AMKH performs well under various conditions.

4 Summary

We studied an AMKH approach to monitoring a large number of subscriptions on the Web. It represents subscriptions as multidimensional data points and looks up candidate coordinates from a hash table for event matching. AMKH effectively controls the number of subscriptions that need to be examined for matching an event. It judiciously includes a small set of clause IDs from general attributes in hash computation and it controls the hash ranges of non-equality coordinate values. Simulations were conducted to evaluate the performance of AMKH and the results show that AMKH performs well under various conditions.

References

[1] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching events in a content-based subscrption system. In *Proc. of Symp. on Principles of Distributed Computing*, 1999.

[2] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proc. of the ACM SIGMOD*, 2001.

[3] E. Hanson, C. Carnes, L. Huang, M. Konyala, L. Noronha, S. Parthasarathy, J. B. Park, and A. Vernon. Scalable trigger processing. In *Proc. of Int. Conf. on Data Engineering*, pages 266–275, 1999.

[4] E. Hanson and T. Johnson. Slection predicate indexing for active databases using interval skip lists. *Information Systems*, 21(3):269–298, 1996.

[5] D. E. Knuth. *The Art of Computer Programming, Vol 3: Sorting and Searching.* Addison-Wesley, 1973.