SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2016, Vol. 59 022315:1–022315:16
doi: 10.1007/s11432-015-5500-x

Toward high efficiency for content-based multi-attribute event matching via hybrid methods

Wenhao FAN[1,2*], Yuanan LIU[1,2] & Bihua TANG[1,2]

[1]*School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing* 100876*, China;*
[2]*Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing* 100876*, China*

Received July 7, 2015; accepted September 30, 2015; published online January 5, 2016

Abstract Event matching is a core in decoupled end-to-end communications, which are extensively applied to various areas. Event matching seeks the subscriptions that match a given event from a subscription set, however, this work becomes increasingly complicated in content-based multi-attribute scenarios, where events and subscriptions are formed in content, and described by multiple attributes. In addition, large-scale systems are easier to suffer from severe degradation in event matching performance. To this end, this paper presents a high-efficiency content-based multi-attribute event matching algorithm, called HEM (hybrid event matching), which is hybridized by 2 different methods. In HEM, the matching on each single attribute (called single-attribute matching) is processed by a triangle-based matching method or a direct matching method dynamically. All single-attribute matchings are sorted via a fast near-optimal algorithm, and each of them is carried out sequentially. In this manner, the searching space of event matching shrinks gradually, so that the searching performance is boosted along with the process of event matching. Experiments are conducted to evaluate HEM comprehensively, where it is observed that HEM outperforms 3 state-of-the-art counterparts (TAMA, H-TREE and REIN) in main criteria, such as event matching time, insertion time and deletion time. Moreover, the gap of performance between HEM and the counterparts enlarges with the increase of system scale.

Keywords event matching, publish/subscribe, data dissemination, end-to-end communications, interval search, combinatorial optimization

Citation Fan W H, Liu Y A, Tang B H. Toward high efficiency for content-based multi-attribute event matching via hybrid methods. Sci China Inf Sci, 2016, 59(2): 022315, doi: 10.1007/s11432-015-5500-x

1 Introduction

Decoupled end-to-end communication is a typical pattern in distributed communications, which aims at realizing full decoupling of time, space and synchronization between information publishers and subscribers [1]. Decoupled end-to-end communication has been extensively adopted in many areas, such as content delivery service [2], mobile push notification [3], IoT searching [4], content-based routing [5], high frequency trading [6], network security monitoring [7], and genomic feature matching [8].

In decoupled end-to-end communications, a publisher is defined as an information sender who publishes events, while, a subscriber is defined as an information receiver based on its submitted subscriptions which

* Corresponding author (email: whfan@bupt.edu.cn)

declare the subscriber's preferences to certain classes of events. Once the system receives an event from a publisher, it carries out a searching process for all subscribers whose preferences match the event, then distributes the event to all matched subscribers. The above searching process is called event matching, which is a cross point between information input and output, thus, it is a core in decoupled end-to-end communications, and the event matching performance directly impacts the system performance.

In content-based multi-attribute scenarios, the main characteristics lie in that events and subscriptions are formed in content, and described by multiple attributes. Therefore, the content-based multi-attribute event matching has to evaluate an event on each attribute, compare it with subscriptions in content form, then find out all subscriptions that match the event on all attributes.

The rapid growth of users' demands has greatly expanded the type and amount of applications of decoupled end-to-end communications. This trend drives an exponential increase of data scales and network topologies, thus, event matching needs to cope with high-frequency event inputs and subscription inputs, and to maintain high performance in high-load systems which contain massive contents and attributes. In these situations, the searching space of event matching increasingly expands, which results in high searching costs, meanwhile, the dynamic of system enlarges, which strengthens the intension of subscription insertions and deletions. Therefore, it is vital and also very challenging to improve the performance of event matching, which is the bottleneck of the system.

Multiple event matching algorithms have been proposed in recent years. Different models, methods and patterns are designed to perfect the performance of event matching in content-based multi-attribute scenarios. In these studies, some algorithms [9–13] improve the performance through wiping off unnecessary searching. The final matching result is obtained by carrying out single-attribute matching on each attribute and then integrating the partial result on each attribute. Some algorithms [14–17] improve the performance via wiping off redundant subscriptions. The searching space shrinks by simplifying the subscription set, which is done through exploiting the relationships among subscriptions before the process of event matching.

This paper focuses on reducing unnecessary searching, however, the existing work still lacks flexibility in the process of event matching, which leads to inefficiency of system, especially in high-load situations. Actually, for different events, the single-attribute matching varies on different attributes, so methods can be adjusted to adapt to current context, and moreover, optimization can also be done between two adjacent single-attribute matchings. To this end, in this paper, we propose a high-efficiency content-based multi-attribute event matching algorithm, called HEM (hybrid event matching), which is hybridized by two different methods. HEM supports range-based content descriptions, using a triangle-based matching method or a direct matching method dynamically to carry out single-attribute matching. The former is designed based on analytic geometry theories, where subscriptions are mapped to the points in a triangle area, and the matching is conducted through filtering out unmatched subscriptions with a rectangle in the area, whereas, the latter picks up the matched subscriptions directly from a partial result set. The two methods have different matching costs on the same attribute. In the process of event matching of HEM, all single-attribute matchings are deployed to a queue. HEM sorts the single-attribute matchings based on the matching rate for current event, and decides which method from the two is employed for each single-attribute matching. The above two steps are carried out by executing a fast near-optimal algorithm, which efficiently solves the optimization problem derived by the event matching cost model of HEM with only a small computation cost. Then, these single-attribute matchings are conducted sequentially according to the sequences and methods on attributes, where the partial result output from a previous single-attribute matching is used to be the input of the next single-attribute matching. Finally, the output from the last single-attribute matching at the end of the queue is the matching result. In such way, HEM filters out unmatched subscriptions or selects matched ones step by step, so that the searching space of each single-attribute matching shrinks gradually. Thus, this mechanism can promote the event matching performance efficiently, especially in high loading situations with massive contents and attributes. Besides, HEM uses arrays to store and arrange subscriptions, so it has very fast subscription insertion and deletion, since these operations can be completed in a constant time due to the direct random access of arrays. Hence, HEM is also very adaptable to high dynamic scenarios.

Experiments are conducted to evaluate the performance of HEM extensively, where the number of subscriptions, the number of attributes, and the width of range-based constraints are chosen as parameters to form different communication scenarios. HEM is compared with several recent typical counterparts (TAMA, H-TREE and REIN). We also analyze the fast near-optimal algorithm used in HEM, and compare it with the optimal algorithm to evaluate the impact to event matching performance. The experiment results demonstrate that HEM outperforms its counterparts to a large extent in main criteria, such as matching time, insertion time and deletion time. Moreover, the superiority is more significant when system scale increases.

Our main contributions in this paper are as follows:

(1) We design two different methods to cope with single-attribute matching, where, the triangle-based matching method transforms the searching to analytic geometry problems, and filters out unmatched subscriptions via a rectangle in a triangle area. The direct matching method picks up matched subscriptions from a subscriptions set directly.

(2) We design a queue-like workflow to model the process of event matching, where a fast near-optimal algorithm is adopted to decide the sequence and method on each attribute. The fast near-optimal algorithm can efficiently solves the optimization problem derived by HEM with only a small cost.

(3) The performance of HEM is evaluated extensively in multiple scenarios, and is compared with several counterparts. Moreover, the fast near-optimal algorithm is analyzed in detail, and is compared with the optimal algorithm.

The rest of this paper is organized as follows: Section 2 discusses the related work. Section 3 defines relevant terminologies employed in HEM. Section 4 presents the design, details and analysis of HEM. Section 5 shows the experiment results and evaluations of the performance of HEM. Section 6 concludes our work.

2 Related work

Currently, event matching is a hot issue in decoupled end-to-end communications. The related work can be generally divided into two categories: wiping off unnecessary searching and wiping off redundant subscriptions. HEM belongs to the category of wiping off unnecessary searching.

The performance of event matching can be enhanced by wiping off unnecessary searching. TAMA [9] builds up a layered index structure for each attribute, and bisects each layer into cells, where corresponding subscriptions are placed based on the ranges of its constraints. TAMA collects the partial result on each attribute by traversing corresponding cells from the top layer to the bottom layer, then obtains the final result from all partial results via a counting algorithm. The main drawbacks of TAMA are that it can only support approximate matching, and it needs extra storage since a subscriptions may be stored into multiple cells, and also it uses the counting algorithm which can not effectively wipe off unnecessary searching. H-TREE [10] is based on a tree-like index structure, which connects several specified attributes layer by layer. In each layer, the range of each attribute is partitioned into several partially overlapped cells. A subscription is mapped into a single or multiple cells based on the center locations and width of its constraints. In this way, H-TREE groups similar subscriptions, so that event matching can be only carried out in related groups. However, the performance of H-TREE is restricted by width of constraint. Subscriptions with wide constraints are split and stored into too many cells, which lead to increase of storage costs and deterioration of the performance of event matching comparatively. REIN [11] uses two bucket lists to store the low values and high values of constraints in subscriptions, respectively. Unmatched subscriptions are excluded by traversing the two bucket lists. A bit set is employed to cache partial results. Although, the dynamics among each single-attribute matching are still not fully utilized since partial results are simply accumulated in the process of event matching. Further, the insertion and deletion of a subscription need to operate both bucket lists, which generates extra costs.

The performance of event matching can be also improved through decreasing the scale of subscription set before event matching. The main principle is to merge similar subscriptions according to their relation-

ships. Literatures [14, 15] organize subscriptions via tree-like structures, and Hilbert space-filling curve is adopted to reduce the multi-dimensional searching space to one-dimensional space. A normalization method is used in Beretta [16], which enables efficient parametric and structural updates of subscriptions. Literature [17] proposes a similarity-based filter clustering to reduce overall event traffic and performs self-tuning summary precision selection to optimize throughput. Wiping off redundant subscriptions is a previous step before the process of event matching, thus, HEM is compatible with these algorithms, and the performance can be further promoted via them.

3 Relevant terminologies

Attributes are descriptions of an object from different aspects. In content-based multi-attribute scenarios, events and subscriptions can be described by multiple attributes when they show multiple characteristics. The quantification of an attribute is defined as the attribute value of the attribute. We define a set, which consists of all attributes in the system, as $\boldsymbol{A} = \{a_1, a_2, ..., a_M\}$, where $a_m \in \boldsymbol{A}$. The attribute value of a_m is denoted by v_m. For simplicity, we define the low value of the attribute value of a_m as 0, and the high value as R_m, thus the range of a_m can be expressed as $[0, R_m]$.

An event, which is published by a publisher at a certain time, location and area, is a conceptual encapsulation of information. According to application context, event can be called message, notification or publication as well. In content-based multi-attribute scenarios, an event is described by multiple attributes and their attribute values, so here, we define an event with M attributes as $\boldsymbol{E} = \{v_1, v_2, ..., v_M\}$.

A subscriber can subscribe or unsubscribe subscriptions. A subscription is the preference of a subscriber to a certain class of events. In content-based multi-attribute scenarios, a subscription is composed of multiple quantified constraints which correspond to the attributes in $\boldsymbol{A}$, and is expressed as $\boldsymbol{S}_n = \{c_1^{(n)}, c_2^{(n)}, ..., c_M^{(n)}\}$. We represent all N subscriptions in the system as a set $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, ..., \boldsymbol{S}_N\}$. HEM supports range-based constraints, that is, a constraint consisting of an interval in the range of its corresponding attribute. The type of an interval is partitioned into 4 categories: left-inclusive and right-inclusive, left-exclusive and right-inclusive, left-inclusive and right-exclusive, left-exclusive and right-exclusive.

Given $\boldsymbol{A} = \{a_1, a_2, ..., a_M\}$, $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, ..., \boldsymbol{S}_N\}$ and $\boldsymbol{E} = \{v_1, v_2, ..., v_M\}$, $\boldsymbol{S}_n$ matches $\boldsymbol{E}$ when all constraints in $\boldsymbol{S}_n \in \boldsymbol{S}$ satisfy their corresponding attribute values in $\boldsymbol{E}$, that is, for each v_m in $\boldsymbol{E}$, $v_m \in c_m^{(n)}$. The set that includes all matched subscriptions is defined as result set, and we express it as $\widehat{\boldsymbol{S}} = \{\boldsymbol{S}_n | \boldsymbol{S}_n \in \boldsymbol{S} \wedge (\boldsymbol{S}_n \text{ matches } \boldsymbol{E})\}$. For an attribute $a_m \in \boldsymbol{A}$, $\boldsymbol{S}_n$ partially matches $\boldsymbol{E}$ on a_m if $c_m^{(n)}$ satisfies v_m. The set that includes all partially matched subscriptions is defined as partial result set, and we express it as $\boldsymbol{S}^{(m)} = \{\boldsymbol{S}_n | \boldsymbol{S}_n \in \boldsymbol{S} \wedge (\boldsymbol{S}_n \text{ partially matches } \boldsymbol{E} \text{ on } a_m)\}$, and is generally referenced as $\widetilde{\boldsymbol{S}}$.

4 Design, details and analysis of HEM

In this section, firstly, we introduce the basic idea of HEM and its features; secondly, the triangle-based matching method and the direct matching method utilized for single-attribute matching are explained respectively; thirdly, the workflow of event matching of HEM is described, including the event matching cost model of HEM and the fast near-optimal algorithm used to determine sequences and methods.

4.1 Overview of HEM

HEM makes improvements on both the single-attribute matching and the integration of partial results. HEM constructs a queue, which is actually an ordered sequence of the single-attribute matchings on all attributes. The partial result output by a previous single-attribute matching is taken as the input of the next single-attribute matching. For an event, HEM first evaluates the matching rate of the event on each attribute, then decides the sequence of each attribute in the queue. Then according to a fast near-optimal algorithm, HEM chooses the better method from the triangle-based matching method and the

direct matching method for each single-attribute matching. Finally, the matching result can be obtained at the end of the queue.

The choice of sequence and method for each attribute is the dynamic factor in HEM, which is determined by the event and the corresponding matching costs of the two methods. We establish the event matching cost model of HEM, and then design a fast near-optimal algorithm based on several observations from the model. The algorithm can provide near-optimal solutions, but only costs $O(M\log M + M)$ time.

Different from existing algorithms, such as TAMA [9], H-TREE [10] and REIN [11], etc., which only employ a single method to carry out event matching, conversely, HEM uses two methods adaptively to improve event matching performance via fully utilizing the different characteristics of the two methods. Moreover, the existing algorithms neglect the fact that a subscription unmatches an event as long as any one of its constraints does not satisfy the event on the corresponding attribute. Thus, HEM conducts event matching sequentially, and the partial result of a next single-attribute matching is searched from its previous single-attribute matching. Unmatched subscriptions are filtered out at each single-attribute matching gradually, and in this way, the searching space shrinks in the process of event matching, so the performance is promoted.

4.2 Methods used for single-attribute matching

Given $\boldsymbol{A} = \{a_1, a_2, \ldots, a_M\}$, $\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N\}$, for $\boldsymbol{E} = \{v_1, v_2, \ldots, v_M\}$ and $\boldsymbol{S}_n = \{c_1^{(n)}, c_2^{(n)}, \ldots, c_M^{(n)}\} \in \boldsymbol{S}$, single-attribute matching decides whether $\boldsymbol{S}_n$ partially matches $\boldsymbol{E}$ on $a_m \in \boldsymbol{A}$ by checking if $c_m^{(n)}$ satisfies v_m. Here, HEM uses a triangle-based matching method or a direct matching method to handle the single-attribute matching on a_m.

HEM uses intervals to express range-based constraints, as were aforementioned, which can be categorized by 4 types according to the boundary types of constraints. For $c_m^{(n)}$, we express its low value as $c_m^{(n)}$.low_value, the boundary type of its low value as $c_m^{(n)}$.low_value_type, its high value as $c_m^{(n)}$.high_value, and the boundary type of its high value as $c_m^{(n)}$.high_value_type. The values of boundary type are picked up from EXCLUSIVE and INCLUSIVE.

Since range-based constraints are expressed as intervals, the single-attribute matching is transformed into interval searching.

The single-attribute matchings are connected end to end in a queue, that is to say, current single-attribute matching is carried out based on the partial result set $\widetilde{\boldsymbol{S}}$ from its previous single-attribute matching.

4.2.1 *The triangle-based matching method*

The idea of triangle-based matching method is inspired from analytic geometry theories. The method maps a subscription into a point in a two-dimensional area, according to the constraint of the subscription on current attribute. For $a_m \in \boldsymbol{A}$ and $\boldsymbol{S}_n \in \boldsymbol{S}$, the coordinate of $c_m^{(n)}$ mapped by the triangle-based matching method is formulated by $c_m^{(n)}.x = (c_m^{(n)}.\text{high\_value} - c_m^{(n)}.\text{low\_value})/2 + c_m^{(n)}.\text{low\_value}$ and $c_m^{(n)}.y = (c_m^{(n)}.\text{high\_value} - c_m^{(n)}.\text{low\_value})/2$, respectively, where $c_m^{(n)}.x$ is the mapped value of $c_m^{(n)}$ on x-axis, which is actually the middle point in the range of $c_m^{(n)}$. $c_m^{(n)}.y$ is the mapped value of $c_m^{(n)}$ on y-axis, which is actually the length from the middle point to the border of the range of $c_m^{(n)}$. According to all possible distributions of coordinates from $c_m^{(n)}$ with range $[0, R_m]$ via the above mapping mechanism, all coordinates are located in an isosceles triangle area, as is shown in Figure 1. The short sides L_1 and L_2 of the triangle are formulated as

$$L_1 : y = x, \qquad L_2 : y = -x + R_m. \tag{1}$$

L_1 and L_2 restrict $c_m^{(n)}$, which means that $c_m^{(n)}$ cannot be mapped out of R_m.

The basic principle of single-attribute matching on a_m via the triangle-based matching method is to find out all subscriptions whose constraints on a_m do not satisfy v_m, then exclude them from $\widetilde{\boldsymbol{S}}$, thus,

the rest subscriptions in $\widetilde{\boldsymbol{S}}$ all partially match $\boldsymbol{E}$ on a_m. We have the following theorems to describe all situations that a subscription unmatches an event on an attribute.

Theorem 1. Given a_m, v_m of $\boldsymbol{E}$ and $c_m^{(n)}$ of $\boldsymbol{S}_n$, we say $c_m^{(n)}$ does not satisfy v_m when (1) $c_m^{(n)}$.high_value_type = EXCLUSIVE, $c_m^{(n)}$.high_value $\leqslant v_m$, or (2) $c_m^{(n)}$.high_value_type = INCLUSIVE, $c_m^{(n)}$.high_value $< v_m$.

Theorem 2. Given a_m, v_m of $\boldsymbol{E}$ and $c_m^{(n)}$ of $\boldsymbol{S}_n$, we say $c_m^{(n)}$ does not satisfy v_m when (1) $c_m^{(n)}$.low_value_type = EXCLUSIVE, $c_m^{(n)}$.low_value $\geqslant v_m$, or (2) $c_m^{(n)}$.low_value_type = INCLUSIVE, $c_m^{(n)}$.low_value $> v_m$.

Proof. As is formally mentioned, the single-attribute matching is transformed into interval searching. So a constraint does not satisfy an event when the corresponding attribute value of the event locates out of the interval of the constraint. Thus, the 4 conditions listed in Theorem 1 and Theorem 2 cover all possibilities that the interval of $c_m^{(n)}$ does not contain v_m.

Theorem 3. If any one of the 4 conditions in Theorem 1 and Theorem 2 holds, then $\boldsymbol{S}_n$ unmatches $\boldsymbol{E}$.

Proof. $\boldsymbol{S}_n$ matches $\boldsymbol{E}$ only if every constraint in $\boldsymbol{S}_n$ satisfies its corresponding attribute value in $\boldsymbol{E}$, otherwise $\boldsymbol{S}_n$ unmatches $\boldsymbol{E}$. If any one of the conditions in Theorem 3 holds, it means $c_m^{(n)}$ does not satisfy v_m on a_m, so $\boldsymbol{S}_n$ unmatches $\boldsymbol{E}$.

The triangle-based matching method adopts a special analytic geometry approach to handle the single-attribute matching. For v_m of $\boldsymbol{E}$ on a_m, as shown in Figure 2, we put v_m on x-axis based on its value, then starting from which, we draw two lines l_1 and l_2 to partition the triangle area into 2 triangle subareas (Subarea 1 and Subarea 2) and 1 rectangle subarea (Subarea 3). l_1 is parallel to L_2, so the coordinate of the point t_1 is $(0, v_m/2)$. l_2 is parallel to L_1, so the coordinate of the point t_2 is $(0, (R_m - v_m)/2)$. l_1 and l_2's formulations are

$$l_1 : y = -x + v_m, \qquad l_2 : y = x - v_m. \tag{2}$$

It can be observed that the subscriptions whose coordinates locate into Subarea 1 all belong to the conditions described in Theorem 1. Similarly, the subscriptions whose coordinates locate into Subarea 2 all belong to the conditions described in Theorem 2. Hence, the above subscriptions all unmatch $\boldsymbol{E}$ according to Theorem 3. Therefore, all subscriptions in Subarea 3 partially match $\boldsymbol{E}$ on a_m since they are the rest ones after subscriptions in Subarea 1 and Subarea 2 being excluded.

The process of single-attribute matching via triangle-based matching method is to traverse all subscriptions in Subarea 1 and Subarea 2, and exclude them from $\widetilde{\boldsymbol{S}}$ based on Theorem 1 and Theorem 2. The subscriptions whose coordinates locate at l_1 and l_2 should be handled particularly. A subscription $\boldsymbol{S}_n$ which lies at l_1 means that its $c_m^{(n)}$.high_value $= v_m$, thus the high value type of $c_m^{(n)}$ needs to be further checked to judge if it satisfies the condition (1) in Theorem 1. A subscription $\boldsymbol{S}_n$ which lies at l_2 means that its $c_m^{(n)}$.low_value $= v_m$, thus the low value type of $c_m^{(n)}$ needs to be further checked to judge if it satisfies the condition (1) in Theorem 2.

The index structure for a_m is organized by a two-dimensional array, which is a triangle-like non isometric array. The array divides the triangle area into multiple rectangle grids. Figure 3 illustrates an instance of an array with 72 grids. From top to bottom of the array, the number of grids on a previous level is less than that on the next level by 2. A subscription is inserted to or deleted from its corresponding grid based on its coordinate. As shown in Figure 4, we arrange a sequence index for each grid in the array. A sequence number consists of a row index and a column index. We use a hash algorithm to decide which grid that a subscription should be allocated to. We define the max number of grids on a column or a row for a_m as τ_m and $2\tau_m$, respectively, then the row index i_r and column index i_c of $\boldsymbol{S}_n$ on a_m are formulated by

$$i_r = \begin{cases} c_m^{(n)}.y/(R_m/\tau), & c_m^{(n)}.y \neq R_m, \\ \tau - 1, & c_m^{(n)}.y = R_m, \end{cases} \qquad i_c = \begin{cases} c_m^{(n)}.x/(R_m/(2\tau)), & c_m^{(n)}.x \neq R_m, \\ 2\tau - 1, & c_m^{(n)}.x = R_m. \end{cases} \tag{3}$$

The index i_v of v_m can be also computed from Formula 3, and $c_m^{(n)}.x$ or $c_m^{(n)}.y$ is replaced by v_m in this case.

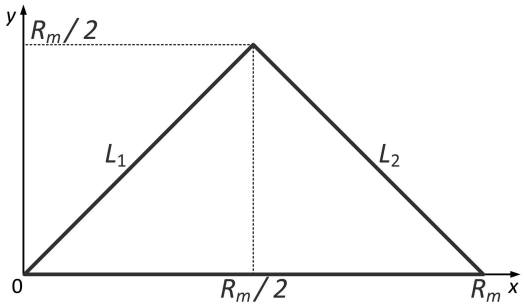

Figure 1 The isosceles right angled triangle area of a_m, and its short sides L_1 and L_2.

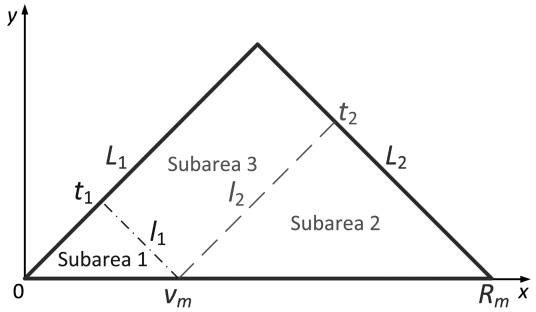

Figure 2 The 3 subareas partitioned from the triangle area of a_m.

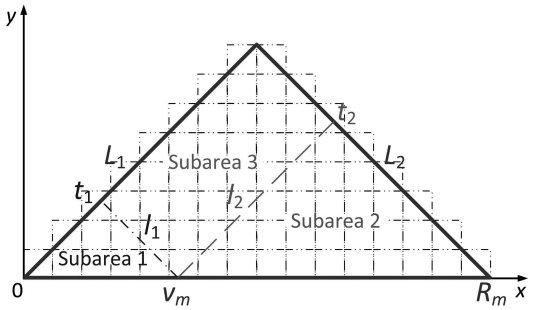

Figure 3 An 2D array with 36 grids for the triangle area of a_m.

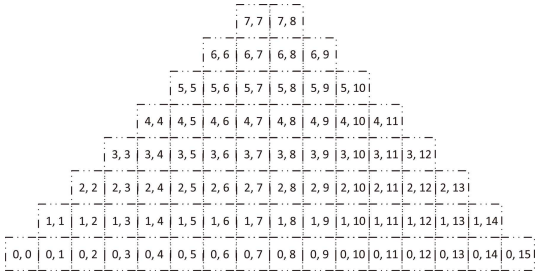

Figure 4 The sequence number of an 2D array with 72 grids.

Now we can explain the process of single-attribute matching via the triangle-based matching method, which can be briefly described as 1) compute the index i_v of v_m, then locate the grids that intersect with l_1 and l_2, so Subarea 1 and Subarea 2 can be partitioned from the triangle area; 2) traverse the subscriptions in Subarea 1. When the grid intersects with both of l_1 and l_2, in the process of traversing and exclusion, the low value, low value type, high value and high value type of constraints all need to be further checked to decide if it should be excluded from $\widetilde{\boldsymbol{S}}$. When a grid only intersects with l_1, the high value and its type of constraints need to be checked to decide if it should be excluded. Exclude other subscriptions from $\widetilde{\boldsymbol{S}}$; 3) traverse the subscriptions in Subarea 2. When the grid intersects with both of l_1 and l_2, do nothing since the same work has been done in 2). When a grid only intersects with l_2, the low value and its type of constraints need to be checked to decide if it should be excluded from $\widetilde{\boldsymbol{S}}$. Exclude other subscriptions from $\widetilde{\boldsymbol{S}}$.

The details of the process of the triangle-based matching method are shown in Algorithm 1.

Because a two-dimensional array is taken as the index structure of the triangle-based matching method, using the hash algorithm described in Formula 3, the subscription insertion and deletion can be completed efficiently. The operations run fast with $O(1)$ time complexity, and cost small storage with $O(1)$ space complexity. This is another superiority of HEM, whereas, in its counterparts (such as TAMA [9], H-TREE [10] and REIN [11]), a subscription may be stored in multiple locations, thus, insertion and deletion operations are related to the writing operations at these locations, which consume more time and occupy more storage.

4.2.2 *The direct matching method*

The idea of the direct matching method is quite straight forward. For a_m, the method directly traverses the subscriptions in $\widetilde{\boldsymbol{S}}$, which is the output of the previous single-attribute matching. In process of single-attribute matching via the direct matching method, a subscription, whose constraint on a_m does not satisfy v_m, is excluded from $\widetilde{\boldsymbol{S}}$. When the process is done, the rest subscriptions remained in $\widetilde{\boldsymbol{S}}$ all

Algorithm 1 The triangle-based matching method

INPUT: v_m of $\boldsymbol{E}$ on a_m, $\widetilde{\boldsymbol{S}}$ of the previous single-attribute matching.
OUTPUT: $\widetilde{\boldsymbol{S}}$ on attribute a_m.
01. Compute the index i_v of v_m, the row index i_{r,t_1} of t_1 and the row index i_{r,t_2} of t_2 via Formula 3;
02. Compute the formulation of l_1 and l_2 via Formula 2, then locate the grids that intersect with l_1 and l_2;
03. Handle Subarea 1:
04. **for**$(i \leftarrow 0; i \leqslant i_{r,t_1}; i \leftarrow i+1)$
05. **for**$(j \leftarrow i; j \leqslant i_v - i; j \leftarrow j+1)$
06. **if** grid(i,j) intersects with l_1 and l_2 **then**
07. Check each subscription in grid(i,j) via Theorem 1 and Theorem 2, if the constraint of the subscription does not satisfy v_m, then exclude the subscription from $\widetilde{\boldsymbol{S}}$;
08. **else if** grid(i,j) intersects with l_1 **then**
09. Check each subscription in grid(i,j) via Theorem 1, if the constraint of the subscription does not satisfy v_m, then exclude the subscription from $\widetilde{\boldsymbol{S}}$;
10. **else**
11. Exclude all subscriptions from $\widetilde{\boldsymbol{S}}$;
12. **end if**
13. **end for**
14. **end for**
15. Handle Subarea 2:
16. **for**$(i \leftarrow 0; i \leqslant i_{r,t_2}; i \leftarrow i+1)$
17. **for**$(j \leftarrow i + i_v; j \leqslant 2\tau - i; j \leftarrow j+1)$
18. **if** grid(i,j) intersects with l_1 and l_2 **then**
19. **continue**; // the same work has been done at line 07
20. **else if** grid(i,j) intersects with l_2 **then**
21. Check each subscription in grid(i,j) via Theorem 2, if the constraint of the subscription does not satisfy v_m, then exclude the subscription from $\widetilde{\boldsymbol{S}}$;
22. **else**
23. Exclude all subscriptions from $\widetilde{\boldsymbol{S}}$;
24. **end if**
25. **end for**
26. **end for**

partially match $\boldsymbol{E}$ on a_m.

The direct matching method is carried out by utilizing the data structure of $\widetilde{\boldsymbol{S}}$, thus, it requires no index structures.

We have the following theorems to describe all situations that a subscription partially matches an event on an attribute.

Theorem 4. Given a_m, v_m of $\boldsymbol{E}$ and $c_m^{(n)}$ of $\boldsymbol{S}_n$, we say $c_m^{(n)}$ satisfies v_m when (1) $c_m^{(n)}$.low_value $<$ v_m and $c_m^{(n)}$.high_value $> v_m$; (2) $c_m^{(n)}$.low_value $= v_m$ and $c_m^{(n)}$.low_value_type = INCLUSIVE; (3) $c_m^{(n)}$.high_value $= v_m$ and $c_m^{(n)}$.high_value_type = INCLUSIVE.

Proof. As is formally mentioned, the single-attribute matching is transformed into interval searching. So a constraint satisfies an event when the corresponding attribute value of the event is contained in the interval of the constraint. In Theorem 4, the condition (1) describes that v_m locates in $c_m^{(n)}$, and the conditions (2) and (3) describe that v_m lies at the low and high inclusive boundary of $c_m^{(n)}$, respectively.

Theorem 5. If all of the 3 conditions in Theorem 4 do not hold, then $\boldsymbol{S}_n$ unmatches $\boldsymbol{E}$.

Proof. $\boldsymbol{S}_n$ matches $\boldsymbol{E}$ only if every constraint in $\boldsymbol{S}_n$ satisfies its corresponding attribute value in $\boldsymbol{E}$, otherwise $\boldsymbol{S}_n$ unmatches $\boldsymbol{E}$. If all of the conditions in Theorem 5 do not hold, it means $c_m^{(m)}$ does not satisfy v_m on a_m, so $\boldsymbol{S}_n$ unmatches $\boldsymbol{E}$.

The direct matching method is designed based on Theorem 5. The process of the method can be described in brief: traverse each subscription in $\widetilde{\boldsymbol{S}}$, and check if it satisfies any of conditions in Theorem 5 on a_m. If all conditions are not satisfied, then exclude the subscription from $\widetilde{\boldsymbol{S}}$. When the traversing and checking are completed, the rest subscriptions in $\widetilde{\boldsymbol{S}}$ all partially match $\boldsymbol{E}$ on a_m.

The details of the process of the direct matching method are shown in Algorithm 2.

4.3 Event matching of HEM

In this section, the event matching mechanism of HEM is introduced. We first show the process of event matching of HEM, then describe HEM's cost model, where an optimization problem is built up to minimize event matching cost of HEM. At last, the design of the fast near-optimal algorithm, which solves the optimization problem efficiently, is explained in detail.

Algorithm 2 The direct matching method

INPUT: v_m of $\boldsymbol{E}$ on a_m, $\widetilde{\boldsymbol{S}}$ of the previous single-attribute matching.
OUTPUT: $\widetilde{\boldsymbol{S}}$ on attribute a_m.
01. **for** each subscription $\boldsymbol{S}_i$ in $\widetilde{\boldsymbol{S}}$ **do**
02. **if not** (($c_m^{(i)}$.low_value $< v_m$ **and** $c_m^{(i)}$.high_value $> v_m$) **or**
($c_m^{(i)}$.low_value $= v_m$ **and** $c_m^{(i)}$.low_value_type = INCLUSIVE) **or**
($c_m^{(i)}$.high_value $= v_m$ **and** $c_m^{(i)}$.high_value_type = INCLUSIVE)) **then**
03. Exclude $\boldsymbol{S}_i$ from $\widetilde{\boldsymbol{S}}$;
04. **end if**
05. **end for**

Algorithm 3 Event matching algorithm of HEM

INPUT: $\boldsymbol{E}$, $\boldsymbol{S}$.
OUTPUT: $\widehat{\boldsymbol{S}}$.
01. Based on the fast near-optimal algorithm, determine $\boldsymbol{Q}$ and $\boldsymbol{T}$.
02. $\widetilde{\boldsymbol{S}} \leftarrow \boldsymbol{S}$;
03. **for** $k \leftarrow 1 : M$ **do**
04. **if** $t_k = 0$ **then** // use the triangle-based method
05. Carry out single-attribute matching on a_{q_k} via Algorithm 1 with inputs v_{q_k} and $\widetilde{\boldsymbol{S}}$;
06. **else** // $t_k = 1$, use the direct method
07. Carry out single-attribute matching on a_{q_k} via Algorithm 2 with inputs v_{q_k} and $\widetilde{\boldsymbol{S}}$;
08. **end if**
09. **end for**
10. $\widehat{\boldsymbol{S}} \leftarrow \widetilde{\boldsymbol{S}}$.

4.3.1 *Event matching process of HEM*

In HEM, the integration of partial results is conducted via an order queue, which is composed of the single-attribute matchings on all attributes. In the queue, the single-attribute matchings are connected end-to-end. The subscription set is taken as the input of the first single-attribute matching in the queue, and for other single-attribute matchings, the partial result output by a previous single-attribute matching is passed to the next single-attribute matching, and is taken as its input. Finally, the partial result, which is completed by the last single-attribute matching in the queue, is the matching result.

In each single-attribute matching, there are two candidate methods — the triangle-based matching method and the direct matching method, that can be employed to carry out single-attribute matching.

The dynamics in HEM lie in the sequence of each single-attribute matching in the queue, and the method employed by each single-attribute matching. HEM optimizes these dynamics to promote event matching performance. The optimization is done by a fast near-optimal algorithm, which efficiently sub-optimally solves the optimization problem derived from HEM's cost model with a small time cost. The fast near-optimal algorithm is explained in detail later.

Two vectors $\boldsymbol{Q}$ and $\boldsymbol{T}$ are adopted to express the sequences and the methods, respectively. The sequence set is denoted by $\boldsymbol{Q} = \{q_1, q_2, ..., q_M\}$, where the value of $q_k \in \boldsymbol{Q}$ is the index of the kth attribute in the queue; the method set is denoted by $\boldsymbol{T} = \{t_1, t_2, ..., t_M\}$, where $t_k \in \boldsymbol{T}$ represents the single-attribute matching method employed on the kth attribute in the queue. If $t_k = 0$, then the triangle-based matching method is employed, whereas, the direct matching method is employed if $t_k = 1$.

In the process of event matching of HEM, $\boldsymbol{Q}$ and $\boldsymbol{T}$ are determined firstly via the fast near-optimal algorithm. Then the single-attribute matchings in the ordered queue are executed sequentially based on $\boldsymbol{Q}$, the proper one from the two methods according to $\boldsymbol{T}$. The subscription set of the system is taken as the input of the first single-attribute matching, and the event matching result $\widehat{\boldsymbol{S}}$ is the output $\widetilde{\boldsymbol{S}}$ of the last single-attribute matching.

The details of event matching process of HEM are shown in Algorithm 3.

$\widehat{\boldsymbol{S}}$ and $\widetilde{\boldsymbol{S}}$ use the same data structure to store subscriptions, which is composed of a linked list and a hash set. The linked list is used to traverse subscriptions by the direct matching method, whereas, the hash set is used to exclude unmatched subscriptions by the triangle-based matching method. The time complexities of single operations of the two structures are all $O(1)$.

4.3.2 *Cost model of HEM*

In the single-attribute matching via the triangle-based matching method, the main costs are brought by traversing all subscriptions in the Subarea 1 and Subarea 2. In the single-attribute matching via the direct matching method, the main costs are brought by traversing all subscriptions in $\widetilde{\boldsymbol{S}}$. Therefore, we use the number of traversed subscriptions to estimate the cost of a method for the single-attribute matching on an attribute.

For v_m of $\boldsymbol{E}$ on a_m, the number of subscriptions in the Subarea 1 and Subarea 2, namely, the number of all unmatched subscriptions for $\boldsymbol{E}$ on a_m, is denoted by U_m. U_m is recorded in the process of subscription insertion and deletion. The number of subscriptions in $\widetilde{\boldsymbol{S}}$ is denoted by $|\widetilde{\boldsymbol{S}}|$.

We normalize U_m and $|\widetilde{\boldsymbol{S}}|$ by the total number $|S|$ of subscriptions in the system, then the matching cost of the triangle-based matching method can be formulated as $\mu_m^{(\mathrm{tr})} = U_m/|\boldsymbol{S}|$, and the matching cost of the direct matching method can be formulated as $\mu_m^{(\mathrm{dr})} = |\widetilde{\boldsymbol{S}}|/|\boldsymbol{S}|$.

The matching rate ρ_m of $\boldsymbol{E}$ on a_m can be defined as the proportion of the partially matched subscriptions in $\boldsymbol{S}$, whose constraints on a_m satisfy v_m. Conversely, the unmatching rate θ_m of $\boldsymbol{E}$ on a_m is the proportion of all unmatched subscriptions. It can be observed that θ_m is actually equal to the normalized number of unmatched subscriptions on a_m, that is

$$\theta_m = 1 - \rho_m = \mu_m^{(\mathrm{tr})}. \tag{4}$$

Based on the correspondence of matching cost and matching rate, we further investigate the event matching cost of HEM. In the single-attribute matchings in the ordered queue, if the kth single-attribute matching employs the triangle-based matching method, then its matching cost is $\mu_{q_k}^{(\mathrm{tr})} = \theta_{q_k}^{(\mathrm{tr})}$. It can be found that $\mu_{q_k}^{(\mathrm{tr})}$ is irrelevant with $|\widetilde{\boldsymbol{S}}|$, so the matching cost of a certain single-attribute matching via the triangle-based method is independent with its sequence in the ordered queue. On the contrary, if the 1st single-attribute matching adopts the direct matching method, then its matching cost $\mu_{q_1}^{(\mathrm{dr})} = |\widetilde{\boldsymbol{S}}|/|\boldsymbol{S}| = |\boldsymbol{S}|/|\boldsymbol{S}| = 1$ since $|\widetilde{\boldsymbol{S}}| = |\boldsymbol{S}|$ for the 1st single-attribute matching; if the 2nd single-attribute matching adopts the direct matching method, then its matching cost $\mu_{q_2}^{(\mathrm{dr})} = |\widetilde{\boldsymbol{S}}|/|\boldsymbol{S}| = |\boldsymbol{S}^{(q_1)}|/|\boldsymbol{S}| = \rho_{q_1}$, because its $|\widetilde{\boldsymbol{S}}|$ is the output from the 1st single-attribute matching, and can be approximated by $\rho_{q_1}|\boldsymbol{S}|$. Likewise, if the kth single-attribute matching adopts the direct matching method, then its matching cost $\mu_{q_k}^{(\mathrm{dr})} = \rho_{q_{k-1}}|\boldsymbol{S}^{(q_{k-2})}|/|\boldsymbol{S}| = \rho_{q_1}...\rho_{q_{k-1}}$. Thus, we have the conclusion that $\mu_{q_k}^{(\mathrm{dr})} = 1$ when $k = 1$, and $\mu_{q_k}^{(\mathrm{dr})} = \prod_{l=q_1}^{q_{k-1}} \rho_l$ when $k > 1$. It can be found that the matching cost of a certain single-attribute matching via the direct method is dependent with its sequence in the ordered queue, because $\mu_{q_k}^{(\mathrm{dr})}$ is computed by the $|\widetilde{\boldsymbol{S}}|$ of each single-attribute matching previous to q_k.

The event matching cost of HEM is the sum of matching costs of all single-attribute matchings in the order queue. We use $\boldsymbol{T}$ to distinguish the method that a single-attribute matching chooses from the triangle-based matching method and the direct method, then partition the cost into the cost by the triangle-based matching method and the cost by direct matching method. Based on above, we have the event matching cost of HEM,

$$\mu = \sum_{k=1}^{M} \mu_{q_k} = \sum_{k=1}^{M} \left((1 - t_k)\mu_{q_k}^{(\mathrm{tr})} + t_k \mu_{q_k}^{(\mathrm{dr})} \right). \tag{5}$$

In order to improve the event matching performance of HEM, the optimization problem is designed to determine $\boldsymbol{Q}$ and $\boldsymbol{T}$ to minimize the event matching cost of HEM, which is formulated from Formula 5

$$\min_{\boldsymbol{Q},\boldsymbol{T}} \mu = \min_{\boldsymbol{Q},\boldsymbol{T}} \left(\sum_{k=1}^{M} \left((1 - t_k)\mu_{q_k}^{(\mathrm{tr})} + t_k \mu_{q_k}^{(\mathrm{dr})} \right) \right). \tag{6}$$

4.3.3 *The fast near-optimal algorithm*

The optimization problem expressed by Formula 6 is indeed a combinatorial optimization problem. The optimal solution $\boldsymbol{Q}^{(*)}$ and $\boldsymbol{T}^{(*)}$ of the problem is very complicated to be obtained directly, since it is a time-consuming work to search, traverse and validate a large amount of feasible solutions. Thus, trying

Algorithm 4 The fast near-optimal algorithm

INPUT: $\boldsymbol{E}$, $\boldsymbol{A}$ and $\boldsymbol{S}$.
OUTPUT: $\boldsymbol{Q}$ and $\boldsymbol{T}$.
01. According to Formula (4), compute ρ_m of each a_m in $\boldsymbol{A}$, then sort $\boldsymbol{A}$ in ascending order of matching rate via an arbitrary sorting algorithm with $O(M\log M)$ time complexity;
02. Choose the first attribute a_m in $\boldsymbol{A}$, then let $q_1 = a_m$ and $t_1 = 0$;
03. $k \leftarrow 2$;
04. $\epsilon \leftarrow 1$;
05. **while** $k \leqslant |\boldsymbol{A}|$ **do**
06. Choose the kth attribute a_m in $\boldsymbol{A}$;
07. $\epsilon \leftarrow \epsilon \cdot \rho_{q_{k-1}}$; // accumulating matching rate
08. **if** $\epsilon > \theta_m$ **then**
09. Let $q_k \leftarrow a_m$ and $t_k \leftarrow 0$;
10. **else**
11. **jump to** line 15;
12. **end if**
13. $k \leftarrow k + 1$;
14. **end while**
15. **while** $k \leqslant |\boldsymbol{A}|$ **do**
16. Choose the kth attribute a_m in $\boldsymbol{A}$;
17. Let $q_k \leftarrow a_m$ and $t_k \leftarrow 1$;
18. $k \leftarrow k + 1$;
19. **end while**

to find the optimal solution is not practical for HEM, because the extra computation cost brought by the optimal algorithm can not be neglected, and it may worsen the event matching performance of HEM to a considerable extent.

In order to balance optimality of the solution and computation cost, we propose a fast near-optimal algorithm, which can obtain near-optimal solution and only has $O(M\log M + M)$ time complexity. The algorithm is based on several observations as listed below.

Observation 1. In the optimal solution, the method employed by the first single-attribute matching in the ordered queue is always the triangle-based matching method.

Observation 2. If both the triangle-based and direct matching methods are employed in the optimal solution, the sequence of attributes in the ordered queue can be partitioned into a pre-part and a post-part. The triangle-based matching method is employed on all attributes in the pre-part, whereas the direct matching method is employed on all attributes in the post-part.

Observation 3. If both of the triangle-based and direct matching methods are applied in the optimal solution, in the ordered queue, the attributes on which the direct matching method is employed are located in ascending order of matching rate.

Here, we do not provide the proofs of above observations due to the page limitation of this paper. According to above observations, the fast near-optimal algorithm is designed, which is categorised as a greedy algorithm. Firstly, $\boldsymbol{A}$ is sorted in ascending order of matching rate based on Observation 3. Then, the first attribute is selected to be put into the first location in the ordered queue, and the triangle-based matching method is employed based on Observation 1. The next attribute is chosen from the sorted $\boldsymbol{A}$, and put to the next location in the ordered queue. We use a factor ϵ to record the product of the matching rates of previously handled attributes, namely, the matching cost on current attribute via the direct matching method. If ϵ is greater than the matching rate of current attribute, then the triangle-based matching method is selected to apply on the attribute, so forth. If not, then the rest attributes all employ the direct matching method based on Observation 2. Their sequences are in ascending order of matching rate, which have been already sorted in $\boldsymbol{A}$.

The details of the fast near-optimal algorithm are shown in Algorithm 4.

5 Experiment results and performance evaluations of HEM

In this section, we evaluate the performance of HEM extensively. Experiments are conducted in several scenarios with multiple settings of parameters, where HEM is compared with its counterparts TAMA [9], H-TREE [10] and REIN [11]. In experiments, main criteria of event matching are evaluated, which

Table 1 The parameters used in experiments

Name	Meaning	Value
N	The number of subscriptions	$[1 \times 10^5, 1.7 \times 10^6]$
M	The number of attributes	$[3, 17]$
δ	The width of range-based constraints	$[0.05, 0.8]$
τ	The max number of grids on a row or a column in the arrays of the triangle-based matching method in HEM	80
η	The number of cells in H-TREE	4
λ	The number of indexed attributes in H-TREE	6
ξ	The number of discretization levels in TAMA	12
ω	The number of buckets in REIN	3000

include event matching time, subscription insertion time and subscription deletion time. Additionally, the performance of the fast near-optimal algorithm in HEM is analysed and compared with its optimal algorithm.

The programs of all experiments are executed on a Dell PowerEdge R720 server with a 2.8 GHz Intel Xeon E5-2680 CPU and 96 GB 1866 MHz RAM, which runs Ubuntu 12.04 with Linux kernel 3.8.0. The 4 algorithms are all implemented in Java language, and the programs are executed under the Java Virtual Machine Version J2SE 1.8. Parallelism is not used in experiments. Parameters used in experiments are based on Table 1 unless stated clearly. In each experiment, subscriptions with specified parameters are initialized firstly, and they are inserted to the index structures of the 4 algorithms. For each algorithm, the time interval from the insertion of the first subscription to the insertion of the last subscription is recorded, and it is called total subscription insertion time. When evaluating the event matching time, 500 events are randomly generated firstly, and then they are taken as the inputs for the 4 algorithms. For each algorithm, the time when an event is received and the time when the matching result is generated are recorded, and the time interval is the event matching time for the event. The average event matching time is computed by averaging the event matching time for the 500 events. The subscription deletion begins when the event matching for all 500 events is done. The subscriptions are deleted for the index structures of the 4 algorithm one by one. For each algorithm, the time interval from the deletion of the first subscription to the deletion of the last subscription is recorded, and it is called total subscription deletion time.

In TAMA [9], H-TREE [10], REIN [11] and our HEM, TAMA belongs to approximate event matching algorithm, whereas, other algorithms are exact event matching algorithms. The matching result via the former may contain positive fault [9], which leads to a slightly larger result than the exact result; the algorithms in the latter can obtain exact matching result. There are several parameters of these algorithms and of the scenarios, which are listed in Table 1.

In experiments, the range of all attribute values is normalized to $[0, 1]$; attribute values of events, low value and high values of constraints are generated randomly from their corresponding ranges with precision of 10^{-6}. The proportion of 4 boundary types of range-based constraints is $1 : 1 : 1 : 1$. For the width δ of constraint, the low value of a constraint is randomly generated from $[0, 1 - \delta]$, and its high value is from $[\delta, 1]$.

5.1 Event matching time

As the vital metric in the performance of event matching algorithm, event matching time is evaluated extensively by experiments in different scenarios with 3 variable parameters: number of subscriptions, number of attributes, and width of range-based constraints.

5.1.1 *Event matching with different numbers of subscriptions*

We measure the average event matching time of the 4 algorithms with different N, and the parameters used in experiments are set as $M = 10, \delta = 0.5$ fixedly. The experiment results are shown in Figure 5.

As N increases from 1×10^5 to 1.7×10^6, the average event matching time of the 4 algorithms increases, but the growth rate of HEM is the lowest among them. On average, the event matching time of HEM is 16.53%, 35.07% and 50.78% of that of TAMA, H-TREE and REIN, respectively. The above results prove the efficiency of HEM on event matching performance. Moreover, the gaps between HEM and other 3 algorithms widen as N increases. It can be seen that the deterioration of event matching time of HEM caused by the increase of N is minimum when compared with its 3 counterparts, so HEM is superior to them for large-scale systems.

The advantage of HEM on event matching time with different N is mainly brought by the hybridized methods and queue-like integration mechanism. The searching space shrinks at stages of single-attribute matchings in the process of event matching, thus, in such fine-grained way, the event matching time shortens efficiently.

5.1.2 *Event matching with different numbers of attributes*

The amount of attributes in events and the constraints of subscriptions are another two important parameters that impact event matching performance. The average event matching time by the 4 algorithms is measured in experiments with different M, where the fixed parameters are $N = 9 \times 10^5$ and $\delta = 0.5$. As shown in Figure 6, generally, the average event matching time increases with the increase of M. This is mainly because the searching space expands as attributes grow, and the searching costs increase correspondingly. We found that the event matching time of H-TREE first decreases before $M = 5$, then increases after $M = 5$, since the dropping of matching rate is the leading role in the pre-part, whereas, the expansion of searching space dominates in the post-part. Among the 4 algorithms, HEM is the best one with the lowest average event matching time, and even more, its curve is nearly flat as N grows. This fact demonstrates that HEM is not only highly efficient in large-scale systems, but also has superior adaptability with different N.

The advantage of HEM on event matching time with different N is still due to the single-attribute matching with hybrid methods and queue-like integration mechanism. Unmatched subscriptions are excluded in time at each single-attribute matching, and as the number of attributes increases, the matched subscriptions decrease actually. In this case, single-attribute matchings with low matching rates are handled in priority, so there is very small amount of work for the last single-attribute matchings in the order-queue. This boosts the event matching speed to a large extent.

5.1.3 *Event matching with different widths of range-based constraints*

The matching rate on an attribute is directly impacted by δ on the attribute. When δ is narrow, the possibility that subscriptions match an event is low, whereas, it is high if the width is wide. In order to evaluate the impact of δ on event matching time, experiments are done with different δ, where other parameters are configured as $M = 10$ and $N = 9 \times 10^5$ fixedly. As shown in Figure 7, when δ grows from 0.05 to 0.8, it can be observed that the 4 algorithms behave differently. The average event matching time of TAMA increases linearly, because subscriptions with wide constraints are stored in multiple cells in the layered index structure of TAMA, which increases searching cost. The average matching time of H-TREE grows exponentially due to the duplicates of subscriptions with wide constraints increase exponentially in the tree-like structure of H-TREE, thus, the number of search paths grows correspondingly, which worsens the event matching performance. The average event matching time of REIN decreases with the increase of δ due to the exclusive mechanism that REIN is based on, and the number of subscriptions in the traversing of the two bucket lists shrinks as δ grows, so the searching cost decreases. The average event matching time of HEM is almost unchanged as δ grows, and the main reason lies in that: when δ is low, the matching rates on all attributes decrease, single-attribute matching with low matching rate is carried out in priority, and the direct matching method is frequently adopted to speed up the event matching; when δ is high, the matching costs via the triangle-based matching method on all attributes decrease since the method excludes unmatched subscriptions, so the event matching performance is also boosted. In this hybrid way, the event matching performance of HEM with different δ is promoted holistically.

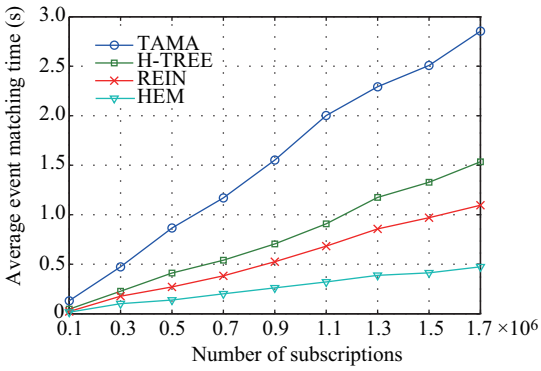

Figure 5 (Color online) The average event matching time with different numbers of subscriptions.

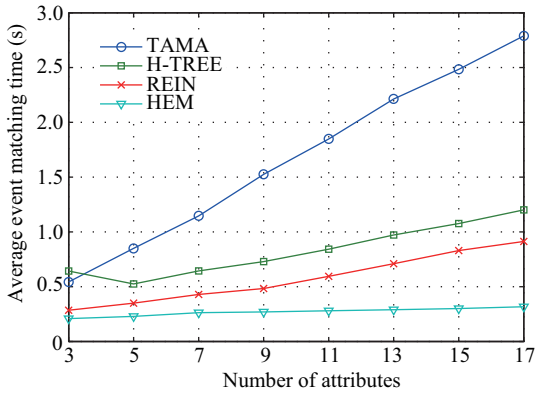

Figure 6 (Color online) The average event matching time with different numbers of attributes.

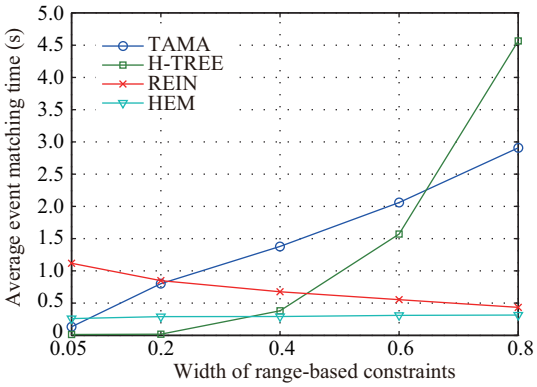

Figure 7 (Color online) The average event matching time with different widths of range-based constraints.

5.2 Subscription insertion and deletion time

As shown in Figures 8 and 9, the subscription insertion time and subscription deletion time of the 4 algorithms are measured in order to evaluate their maintenance mechanism, respectively. The total subscription insertion and deletion time are recorded in experiments with different numbers of subscriptions, where other parameters are set as $M = 10$, $\delta = 0.5$ fixedly, and y-axis is in log-scale. The results show that HEM has the lowest insertion and deletion time among the 4 algorithms due to its $O(1)$ time complexity in theoretical analysis. The average subscription insertion time per subscription is 0.0358 ms, 0.0641 ms, 0.0039 ms and 0.00067 ms via TAMA, H-TREE, REIN and HEM, respectively. The average subscription deletion time per subscription is 0.0232 ms, 0.0342 ms, 0.0054 ms and 0.0012 ms via the 4 algorithms, respectively. In the triangle-based matching method, HEM uses a two-dimensional array to express the triangle area with grids, and subscriptions are stored in their corresponding grids. Thus HEM can provide very fast subscription insertion and deletion time because searching for a specific grid can be done immediately ($O(1)$ time complexity) via the proposed hash algorithm. In contrast, subscriptions may be inserted into or deleted from multiple cells in the index structures of TAMA and H-TREE. REIN, as the best on subscription insertion and deletion among the 3 algorithms, still has to handle the low value and high value of the constraint in a subscription from REIN's two bucket lists, respectively.

5.3 Analysis of the fast near-optimal algorithm and comparisons

In HEM, the sequence of each single-attribute matching and the method employed by each single-attribute matching are determined by the fast near-optimal algorithm. The result generated by the algorithm directly impacts the event matching performance of HEM. To this end, the fast near-optimal algorithm is analyzed and further compared with the optimal algorithm, which traverses and examines all feasible

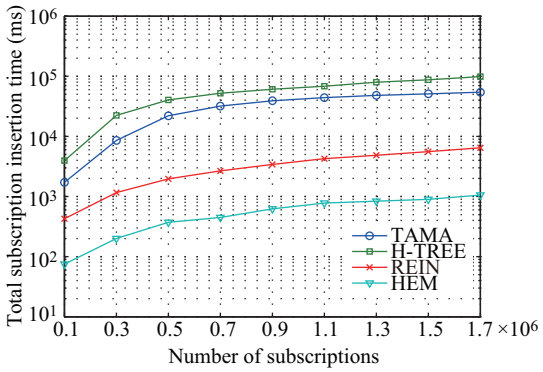

Figure 8 (Color online) The total subscription insertion time with different numbers of attributes.

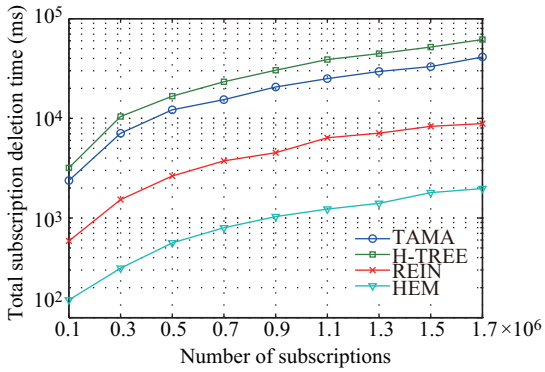

Figure 9 (Color online) The total subscription deletion time with different numbers of attributes.

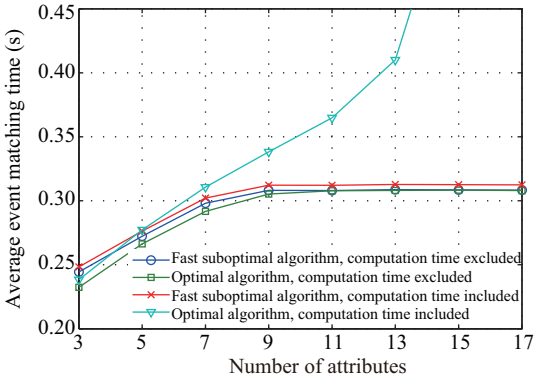

Figure 10 (Color online) The average event matching time by the fast near-optimal algorithm and the optimal algorithm with different numbers of attributes.

solutions that satisfy Observation 1–3, and then finds out the best solution from them. This is a very costly operation since the searching space grows rapidly with the increase in the number of attributes. Experiments are carried out in scenarios with different numbers of attributes from 3 to 17, and $N = 9\times10^5$, $\delta = 0.5$. The average event matching time via the fast near-optimal algorithm, via the optimal algorithm excluding its computation time, and via the optimal algorithm including its computation time is evaluated. As shown in Figure 10, it can be found that the average event matching time via the fast near-optimal algorithm is very close to that via the optimal algorithm excluding its computation time, as the two curves almost coincide after $M = 11$. The gap between the average event matching time via the fast near-optimal algorithm and the optimal algorithm including its computation time grows exponentially since the searching space of the feasible solutions for the optimal algorithm expands exponentially.

The fast near-optimal algorithm provides the near-optimal solution which is near optimal, but only needs a small time complexity ($O(M\log M+M)$). Whereas, the optimal algorithm can obtain the optimal solution, but it generates tremendous computation costs, which, conversely, worsens the holistic event matching performance of HEM severely.

6 Conclusion

In this paper, we propose a high-efficiency content-based multi-attribute event matching algorithm, called HEM (hybrid event matching), which is hybridized by two different methods — the triangle-based matching method and the direct matching method. The former is designed based on analytic geometry theories, where subscriptions are mapped to the points in a triangle area, and the matching is conducted through filtering out unmatched subscriptions with a rectangle in the area, whereas, the latter picks up the

matched subscriptions directly from a partial result set. The integration of partial results generated from each single-attribute matching is conducted in an order queue, where the output of a previous single-attribute matching is taken as the input of its next single-attribute matching, and the event matching result is obtained at the end of the queue. The sequence of each single-attribute matching and the method adopted by each-single attribute matching are determined by the fast near-optimal algorithm, which can provide near-optimal solution only with a small computation cost. In this way, the searching space shrinks at the stage of each single-attribute matching, hence, the holistic event matching performance is boosted efficiently. Experiments are carried out extensively, and multiple main criteria in event matching of HEM are evaluated and compared with several typical counterparts — TAMA, H-TREE and REIN. The experiment results show that HEM outperforms its counterparts to a large extent on event matching time, subscription insertion and subscription deletion, which proves that HEM is superior in scenarios with high dynamic large-scale systems. However, there are still several problems that we plan to investigate and solve. Firstly, optimization should be further done on the performance of subscription insertion and deletion for high-dynamic publish/subscribe environments. Secondly, a distributed deployment strategy of our event matching mechanism needs to be designed for broker-based publish/subscribe systems. Additionally, the issues for the implementation of HEM when it is applied to different practical applications systems must be considered as well.

Acknowledgements This work is supported in part by National Natural Science Foundation of China (Grant Nos. 61502050, 61170275), YangFan Innovative & Entrepreneurial Research Team Project of Guangdong Province, Civil Aerospace Science and Technology Project and Fundamental Research Funds for the Central Universities.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Eugster P T, Felber P A, Guerraoui R, et al. The many faces of publish/subscribe. ACM Comput Surv, 2003, 35: 114–131
2 Ma X K, Wang Y J, Sun W D. Feverfew: a scalable coverage-based hybrid overlay for internet-scale pub/sub networks. Sci China Inf Sci, 2014, 57: 052103
3 Muhl G, Ulbrich A, Herrman K. Disseminating information to mobile clients using publish-subscribe. IEEE Internet Comput, 2004, 8: 46–53
4 Guinard D, Trifa V, Karnouskos S, et al. Interacting with the soa-based internet of things: discovery, query, selection, and on-demand provisioning of web services. IEEE Trans Serv Comput, 2010, 3: 223–235
5 Cao F, Singh J P. Efficient event routing in content-based publish-subscribe service networks. In: 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, 2004, 2: 929–940
6 Jayaram K, Jayalath C, Eugster P. Parametric subscriptions for content-based publish/subscribe networks. In: Middleware. Berlin: Springer, 2010. 128–147
7 Krügel C, Toth T, Kerer C. Decentralized event correlation for intrusion detection. In: Information Security and Cryptology-ICISC 2001. Berlin: Springer, 2002. 114–131
8 Layer R M, Skadron K, Robins G, et al. Binary interval search: a scalable algorithm for counting interval intersections. Bioinformatics, 2013, 29: 1–7
9 Zhao Y, Wu J. Towards approximate event processing in a large-scale content-based network. In: 31st International Conference on Distributed Computing System (ICDCS), Minneapolis, 2011. 790–799
10 Qian S, Cao J, Zhu Y, et al. H-tree: an efficient index structure for event matching in content-based publish/subscribe systems. IEEE Trans Parall Distr Syst, 2014, 99: 1–11
11 Qian S, Cao J, Zhu Y, et al. Rein: a fast event matching approach for content-based publish/subscribe systems. In: Proceedings of IEEE INFOCOM, Toronto, 2014. 2058–2066
12 Jerzak Z, Fetzer C. Bloom filter based routing for content-based publish/subscribe. In: Proceedings of the 2nd International Conference on Distributed Event-based Systems, Rome, 2008. 71–81
13 Whang S E, Garcia-Molina H, Brower C, et al. Indexing boolean expressions. Proc VLDB Endowment, 2009, 2: 37–48
14 Jafarpour H, Mehrotra S, Venkatasubramanian N, et al. Mics: an efficient content space representation model for publish/subscribe systems. In: Proceedings of the 3rd ACM International Conference on Distributed Event-based Systems, Nashville, 2009. 7–12
15 Shen Z, Tirthapura S. Approximate covering detection among content-based subscriptions using space filling curves. J Parall Distrib Comput, 2012, 72: 1591–1602
16 Jayaram K, Wang W, Eugster P. Subscription normalization for effective content-based messaging. IEEE Trans Parall Distrib Syst, 2014, 99: 1–11
17 Wang Y M, Qiu L, Verbowski C, et al. Summary-based routing for content-based event distribution networks. ACM SIGCOMM Comput Commun Rev, 2004, 34: 59–74