Future Generation Computer Systems 68 (2017) 289–303

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

DEXIN: A fast content-based multi-attribute event matching algorithm using dynamic exclusive and inclusive methods

Wenhao Fan *, Yuan'an Liu, Bihua Tang

School of Electronic Engineering, and Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing, China

HIGHLIGHTS

- DEXIN, an event matching algorithm for publish/subscribe systems, is proposed.
- DEXIN supports point-based and limited/unlimited range-based constraints.
- DEXIN executes a dynamic pipelined event matching process with two different methods.
- DEXIN uses a low-cost near-optimal algorithm to minimize the event matching cost.
- The superiority of DEXIN is demonstrated by extensive evaluations and comparisons.

ARTICLE INFO

Article history:
Received 10 February 2016
Received in revised form
16 October 2016
Accepted 20 October 2016
Available online 26 October 2016

Keywords:
Event matching
Publish/subscribe
Data dissemination
Combinatorial optimization

ABSTRACT

Event matching acts as a key role in publish/subscribe services, which takes charge of finding all subscriptions that match an event from a subscription set. In large content-based multi-attribute publish/subscribe systems, the performance of event matching is severely challenged due to the increased system scale and search complexity. Most existing event matching algorithms cannot effectively sustain in such a scenario. In this paper, we propose DEXIN (Dynamic EXclusive and INclusive), a fast content-based multi-attribute event matching algorithm using dynamic exclusive and inclusive methods. DEXIN supports highly efficient event matching for subscriptions with limited range-based, unlimited range-based and point-based constraints. DEXIN prepares a pipelined event matching process optimized for each incoming event. The exclusive and inclusive methods, which have different matching costs, are dynamically chosen for event matching over a single attribute. Thus, the search space shrinks very fast since non-matched subscriptions can be filtered out by DEXIN at very early stages in the pipeline. The experiment results demonstrate that DEXIN is superior to several state-of-the-art reference algorithms. Especially, the leading advantage of DEXIN is more significant on matching time and stability for large number of subscriptions and attributes in multiple scenarios.

Publish/subscribe is an interaction mechanism with full decoupling of time, space and synchronization between information producers and consumers [1], which is a common pattern of end-to-end communication services in information systems. Publish/subscribe services are widely applied to various areas, such as mobile push notification [2], IoT service discovery [3], location-based service [4], content-based routing [5], network intrusion detection [6], online advertising [7], high frequency trading [8], genomic feature measurement [9], etc.

In a publish/subscribe system, subscribers are able to submit new subscriptions (called subscribing) or remove their submitted subscriptions (called unsubscribing) to declare or cancel their preferences to some certain classes of events; publishers can send events to the system (called publishing). When the system has received an event from a publisher, it searches all submitted subscriptions that match the event, then disseminates the event to the subscribers who those subscriptions belong to. The process of searching matched subscriptions for an event is called event matching. From the perspective of information flow, event matching locates at the change point from event input to output, hence, it plays a key role that directly affects the system's performance.

Content-based multi-attribute publish/subscribe brings in two new features: 1. event matching is driven by contents rather than the addresses (such as IP addresses) of events; 2. an event is described by multiple attributes, and each attribute represents

* Corresponding author.
E-mail address: whfan@bupt.edu.cn (W. Fan).

http://dx.doi.org/10.1016/j.future.2016.10.020
0167-739X/

one aspect of the event's characteristics. Correspondingly, a subscription includes a subscriber's preference over each attribute. The content-based multi-attribute event matching checks each attribute of the event, and finds out all subscriptions which match the event over all attributes.

The exponential increase of data scales, the rapid expansion of network topologies, and the continuous growth of users' demands, are all pushing publish/subscribe systems forward to coping with dense event inputs, and processing event matching in massive subscriptions and attributes. The load of publish/subscribe system always remains at a high level in large-scale applications. In content-based multi-attribute publish/subscribe systems, event matching becomes the bottleneck since the search space expands and the search complexity increases, therefore, improving the system's event matching performance is vital. Additionally, as increasing requirements on real-time performance impend, optimizing event matching performance is also one of the key issues to meet users' demands and improve service experience.

In recent years, many event matching algorithms and their variants are proposed, which aim at promoting the performance of the content-based multi-attribute event matching. Some algorithms [10–19] reduce unnecessary searching in the process of event matching, which process the single-attribute matching over each attribute and the integration of partial results over all attributes via specific technologies, then obtain the final matching result; some algorithms [20–23] exclude redundant subscriptions in the system by exploiting the relationships of subscriptions, in order to shrink the search space before the process of event matching. Nevertheless, the performance of most algorithms declines with the increase of system scale, which results in the inefficiency of system in high load situations. Therefore, investigating new event matching algorithms to improve the efficiency of event matching, is of both theoretical and practical significance for performance improvements of large-scale publish/subscribe systems.

In this paper, we propose DEXIN (Dynamic EXclusive and INclusive), a fast content-based multi-attribute event matching algorithm for large-scale publish/subscribe systems. Firstly, when processing single-attribute matching, DEXIN uses an exclusive method or an inclusive method dynamically. The former is utilized to exclude non-matched subscriptions from a partial result set; the latter selects matched subscriptions from a partial result set directly. The two methods have different matching costs over the same attribute. Secondly, when processing integration of partial results, the single-attribute matching over each attribute is deployed to a serial pipeline, where the partial result of a next attribute is integrated from the partial result of its previous attribute. DEXIN conducts the single-attribute matching over each attribute sequentially according to the sequences and methods adopted for the attributes in the pipeline.

For an event, DEXIN evaluates the matching rate of the event over each attribute based on the attribute value of the event and the constraints in subscriptions, calculates the matching costs of both exclusive method and inclusive method for each attribute, then determines the sequence and method adopted for each attribute in the pipeline by executing a near-optimal algorithm, which efficiently solves the optimization problem derived by the event matching cost model of DEXIN with small computation cost. Thus, the search space shrinks step by step through DEXIN filtering out non-matched subscriptions or selecting matched subscriptions in each single-attribute matching, then the final matching result can be obtained at the end of the pipeline. This mechanism makes DEXIN able to efficiently support the scenarios with massive subscriptions and attributes, and achieve excellent event matching performance.

Besides providing high real-time event matching, DEXIN also supports fast subscribing/unsubscribing operations, hence, DEXIN is more applicable to high dynamic scenarios. DEXIN implements exact event matching for content-based multi-attribute publish/subscribe systems, and it supports both range-based and point-based content descriptions.

In experiments, we choose 9 criteria to evaluate the performance and the maintenance costs of DEXIN, which include the number of subscriptions, the number of attributes, the width of range-based constraints, the distribution of constraints, the cardinality of attribute values, the proportion of point-based constraints, the matching rate of subscriptions, the subscribing/unsubscribing time and the memory consumption. Comprehensive comparisons are carried out between DEXIN and several state-of-the-art reference algorithms (TAMA [11], H-Tree [12], BE-Tree [13], OpIndex [14] and REIN [15]). Additionally, the near-optimal algorithm is also analyzed to evaluate its impact on the performance of DEXIN. The performance of DEXIN when using either the inclusive method or the exclusive method is compared as well. The experiment results demonstrate that DEXIN has superior performance compared with the reference algorithms. Especially, the leading advantage of DEXIN is more significant on event matching time and stability for large number of subscriptions and attributes in multiple scenarios, while guaranteeing short subscribing/unsubscribing time and low memory consumption.

In this paper, our main contributions are:

(1) With considerations of dynamics in content-based multi-attribute event matching, the proposed DEXIN uses two different methods to handle single-attribute matching, and conducts the single-attribute matching over each attribute sequentially in a serial pipeline according to the sequences and methods adopted for attributes. The matching result is integrated in the process of event matching, which shrinks search space step by step, so that the event matching performance is promoted efficiently.
(2) The event matching cost model of DEXIN is analyzed and established, which takes the sequences and methods adopted for attributes as its parameters. A near-optimal algorithm is also designed to efficiently solve the optimization problem derived from the cost model with small computation cost.
(3) DEXIN is evaluated in different scenarios with multiple parameters, and it is compared comprehensively with several state-of-the-art reference algorithms. Moreover, the near-optimal algorithm is also analyzed and compared in detail.

The rest of our paper is organized as follows. Section 1 discusses the related works on content-based multi-attribute event matching technologies. Section 2 defines several relevant terms used in DEXIN. Section 3 presents the design and analysis of DEXIN. Section 4 shows the experiment results, and evaluates the performance of DEXIN through comparing DEXIN with the reference algorithms. Section 5 concludes our work.

1. Related works

In research of content-based multi-attribute publish/subscribe technologies, event matching algorithm is a hot issue. In recent years, multiple representative algorithms have been proposed, which can be generally divided into two categories: A. reducing unnecessary searching; B. reducing redundant subscriptions.

1.1. Reducing unnecessary searching

The mechanisms of existing event matching algorithms in this area are mainly composed of two parts: the methods for single-attribute matching and the mechanisms for integration of partial results. Literatures [10–19] have proposed multiple event matching algorithms.

TAMA [11] sets up a layered index structure for each attribute, and bisects the range of each attribute into multiple cells layer by layer from the top to bottom. TAMA places a subscription into corresponding cells based on its constraints. During the event matching, for each attribute, TAMA obtains all subscriptions from relevant cells as the partial result, and then calculates the matching result from the partial results over all attributes by the counting algorithm. However, the layered index structure of TAMA can only support approximate matching, and extra storage costs are generated since subscriptions are stored in multiple cells. TAMA uses the counting algorithm to integrate partial results, which leads to non-matched subscriptions being only filtered out at final stage, so that TAMA cannot reduce unnecessary searching efficiently. Additionally, the deviation of TAMA's event matching for different events is high, which leads to a long matching time for some events.

H-Tree [12] builds up a treelike index structure, which partitions the range of each attribute into several partially overlapped cells, and each cell of a previous attribute is associated with its next attribute. A subscription is mapped into a single or multiple cells according to the center locations and widths of its constraints. Thus, H-Tree gathers similar subscriptions into groups and carries out event matching in related groups. Although, the performance of H-Tree is subject to the width of constraint. A subscription with wide-range constraints is split into multiple subscriptions with narrow-range constraints. This drawback not only increases the amount of searching, but also results in exponential increase of storage with the number of subscriptions. Moreover, the deviation of H-Tree's event matching for different events is very high, the event matching performance is deteriorated very severely for some events.

BE-Tree [13] uses a dynamic treelike structure to organize subscriptions, which are categorized by p-directories with specified attributes and by c-directories with multiple bisected intervals. Finally, the event matching is carried out in multiple l-nodes that are the rest ones in a pruning process. However, in order to maintain the node capacity, the scale of BE-Tree may become very huge for a large number of subscriptions, so the length of a search path will extend and the number of search paths in a event matching process will increase rapidly. Also, there are no optimizations designed for the single-attribute matchings in l-nodes. An adjustment mechanism is proposed to dynamic extend the scale of a subtree when the capacity overflows, and to change the arrangements of the nodes in BE-Tree in order to keep the event matching performance under variance of subscriptions. The extra costs brought by the above mechanism will have negative influences on subscribing/unsubscribing operations.

OpIndex [14] designs a two-level structure, which consists of a subscription partitioning level and a predicate partitioning level. The two levels act as an attribute filter and an operator classifier, respectively. Two hash maps are used to optimize the range scans on predicate lists. However, although subscriptions are pruned by attributes and operators, there are no optimizations for the single-attribute matchings. The counting algorithm is used to integrated the final matching result, which leads to low efficiency of event matching performance since non-matched subscriptions cannot be filtered out in time.

REIN [15] utilizes the rectangle intersection method, which builds up two bucket lists for each attribute to store the low values and high values of the constraints in subscriptions, respectively. During the event matching, REIN uses a bit set to store the matching result. For each attribute, REIN excludes non-matched subscriptions by traversing the bucket lists and disables them in the bit set. Finally, the enabled subscriptions left in the bit set are the matching result. REIN uses the same bit set to record the partial result over each attribute, but the search cost for each attribute is still directly related to the number of subscriptions in the system. Therefore, unnecessary searching still exists. Additionally, the bit set is hard to be used in a practical system since the matched subscriptions need to be collected by traversing the set once again.

Our DEXIN belongs to the category of reducing unnecessary searching. The above 5 algorithms all carry out single-attribute matching only with a single method, which neglects the fact that different effects of single-attribute matching over the same attribute can be provided by different methods. DEXIN uses two methods which have different matching costs to conduct single-attribute matching, thus the cost of single-attribute matching over an attribute can decrease by selecting the proper method with lower matching cost. Distinguished with TAMA and OpIndex which use the counting algorithm to generate the matching result, in the process of event matching, DEXIN carries out the single-attribute matching over each attribute sequentially, where the partial result of a next attribute is searched from the partial result of its previous attribute, hence, the non-matched subscriptions of current single-attribute matching are filtered out immediately and are not brought into the next single-attribute matching. Distinguished with H-Tree and TAMA who have quite high deviations of event matching performance, for each incoming event, DEXIN optimizes the sequences and methods adopted for attributes, so it can provide very stable performance for various events. Distinguished with the complex structure of BE-Tree which needs to be maintained by several adjustment mechanisms, in DEXIN, the exclusive method is implemented by arrays and RB-trees, and the inclusive method does not rely on any extra data structures, hence DEXIN needs no extra maintenance costs due to its simple index structure.

1.2. Reducing redundant subscriptions

The search space of event matching also shrinks via reducing redundant subscriptions in the system, which can further promote the performance of event matching. Algorithms in [20–23] build up different data structures, and wipe off redundant subscriptions through fusing similar subscriptions according to their relationships. The above process is also called preprocessing of subscriptions. [20,21] organize subscriptions via treelike structures, check the relationships of constraints in subscriptions by exploiting the Hilbert space-filling curve, and merge the subscriptions in inclusion relationships. Hence, the multi-dimensional search space of event matching is reduced to one-dimensional space. Beretta [22] uses a normalization method to split and subsume subscriptions, which enables efficient parametric and structural updates of subscriptions. [23] uses a similarity-based filter clustering to reduce overall event traffic and performs self-tuning summary precision selection to optimize throughput. Preprocessing of subscriptions is previous to the process of event matching, so DEXIN is compatible with these algorithms, and through preprocessing of subscriptions, the event matching performance of DEXIN can be further improved.

2. Relevant terms

An attribute is a depiction of an object at one aspect of characteristics, so when an object shows multiple different characteristics, it can be described by multiple attributes. An attribute value is the quantification of an attribute, which can be expressed numerically as an integer or decimal. In content-based multi-attribute publish/subscribe systems, attributes are used to describe events and subscriptions, and attribute values are adopted to quantify the attributes in events. In this paper, the attribute set that consists of all attributes is defined as $\mathbf{A} = \{a_1, a_2, \ldots, a_M\}$,

where a_m is an element in $\mathbf{A}$ with index m, that is $a_m \in \mathbf{A}$. v_m is the attribute value of a_m.

In publish/subscribe systems, clients who publish events are called publishers. An event is a conceptual encapsulation of information, which is published by a publisher at a certain time, location and area. An event is used to describe an observation of an object, which can be also called message, notification or publication according to application scenarios. In content-based multi-attribute publish/subscribe systems, an event is defined as a conjunction of multiple attributes and their attribute values. In this paper, for simplicity, an incoming event with M attributes is denoted by $\mathbf{E} = \{v_1, v_2, \ldots, v_M\}$.

In publish/subscribe systems, clients who submit subscriptions are called subscribers, which can also unsubscribe their previously submitted subscriptions. A subscription is a preference of a subscriber to a certain class of events. In content-based multi-attribute publish/subscribe systems, a subscription is constructed by a set of quantified constraints, which is defined as $\mathbf{S}_n = \{c_1^{(n)}, c_2^{(n)}, \ldots, c_M^{(n)}\}$. The constraints are in one-to-one correspondence with the attributes in $\mathbf{A}$, for example, $c_m^{(n)}$ is the constraint of $\mathbf{S}_n$ over a_m. We represent all subscriptions in the system as a set. The subscription set consists of N subscriptions is defined as $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_N\}$.

3. The design and analysis of DEXIN

The design and analysis of DEXIN are given in this section. Firstly, the overview of DEXIN is presented to describe the basic idea of the algorithm; secondly, the conversion from constraints to intervals used by DEXIN is explained; thirdly, we introduce the exclusive and inclusive methods which are utilized to carry out single-attribute matching, the index structures of the two methods are explained as well; fourthly, the integration mechanism of DEXIN is described, then we build up the event matching cost model of DEXIN, and explain how DEXIN determines the sequence and method adopted for each attribute so that the matching cost decreases; finally, the cost model is analyzed in detail, and we propose a near-optimal algorithm to solve the optimization problem derived from the cost model.

3.1. Overview of DEXIN

The innovations of DEXIN are inspired by the dynamics in the event matching process. Here, we give a brief introduction of DEXIN as follows.

Two methods are designed to carry out the single-attribute matching: the exclusive method and the inclusive method. The former excludes the non-matched subscriptions from the current partial result using a data structure combined by arrays and RB-trees, and the latter examines each subscription inside the current partial result via the structure of the partial result itself.

DEXIN constructs a serial pipeline, which is an ordered sequence consisting of the single-attribute matchings over all attributes. DEXIN conducts the single-attribute matching over each attribute sequentially. The partial result which is the output from a previous single-attribute matching is taken as the input of its next single-attribute matching.

The event matching cost model of DEXIN is built up, and a near-optimal algorithm—Low-Matching-Rate-First-Fit Algorithm is designed based on several observations. For each incoming event, the algorithm evaluates the matching rate of the event over each attribute, and it decides the sequence and method adopted for each attribute in the pipeline. The algorithm can provide near-optimal solutions only with a small time complexity.

For each incoming event, DEXIN first runs the near-optimal algorithm, and decides sequences of the single-attribute matchings in the pipeline and allocates the method from the two methods for each single-attribute matching. Then, DEXIN carries out the single-attribute matchings in the pipeline sequentially. Finally, the matching result is obtained at the end of the pipeline.

3.2. Conversion from constraints to intervals

DEXIN supports both range-based constraints and point-based constraints. A range-based constraint conveys that the preference of a subscriber includes a sub-range in the range of the corresponding attribute. Assuming a, b are two values in the attribute range, and $a \leq b$, the range-based constraints can be divided into 8 forms: $>a$, $\geq a$, $<a$, $\leq a$, $>a$ and $<b$, $>a$ and $\leq b$, $\geq a$ and $<b$, $\geq a$ and $\leq b$. A point-based constraint indicates that the preference of a subscriber includes a point in the range of the corresponding attribute, which contains 2 forms: $=a$ and $\neq a$.

DEXIN converts both range-based and point-based constraints to intervals. An interval can be partitioned into 4 types based on its boundaries: left-open and right-open (a, b), left-open and right-closed $(a, b]$, left-closed and right-open $[a, b)$, left-closed and right-closed $[a, b]$. For a constraint $c_m^{(n)}$, we bind the features of interval to it, so we define its low value as $c_m^{(n)}.lowValue$, the boundary type of its low value as $c_m^{(n)}.lowValueType$, its high value as $c_m^{(n)}.highValue$, and the boundary type of its high value as $c_m^{(n)}.highValueType$. The value of boundary type is chosen from OPEN and CLOSED.

A range-based constraint $c_m^{(n)}$ with a limited range ($a <$ and $<b$, $>a$ and $\leq b$, $\geq a$ and $<b$, $\geq a$ and $\leq b$) can be easily expressed as an interval. For a range-based constraint $c_m^{(n)}$ with a unlimited range ($>a$, $\geq a$, $<a$, $\leq a$), its low value or high value does not exist, so we mark the value and value type as $c_m^{(n)}.lowValue =$ Null, $c_m^{(n)}.lowValueType =$ Null or $c_m^{(n)}.highValue =$ Null, $c_m^{(n)}.highValueType =$ Null. If $c_m^{(n)}$ is a point-based constraint with the form like $= a$, we assign the point's value to $c_m^{(n)}.lowValue = c_m^{(n)}.highValue$, and set $c_m^{(n)}.lowValueType = c_m^{(n)}.highValueType =$ CLOSED fixedly. If $c_m^{(n)}$ is a point-based constraint with the form like $\neq a$, we use two ranges ($<a$ and $>a$) to express it, and the definitions of the two corresponding intervals are the same as the range-based constraints with unlimited ranges.

For a_m, if v_m falls into the interval of $c_m^{(n)}$, that is, $\mathbf{S}_n$ satisfies v_m of $\mathbf{E}$, then we say $\mathbf{S}_n$ partially matches $\mathbf{E}$ over a_m. The set that contains all partially matched subscriptions is called partial result set, which is denoted by $\mathbf{S}^{(m)} = \{\mathbf{S}_n | \mathbf{S}_n \in \mathbf{S} \wedge (\mathbf{S}_n \textit{ partially matches } \mathbf{E} \textit{ over } a_m)\}$, and is generally referenced as $\mathbf{S}$.

If every constraint in $\mathbf{S}_n$ satisfies its corresponding attribute value in $\mathbf{E}$, namely, $\forall m(a_m \in \mathbf{A} \rightarrow c_m^{(n)} \textit{ satisfies } v_m)$ holds, then we say $\mathbf{S}_n$ matches $\mathbf{E}$. The set that contains all matched subscriptions is called result set, which is denoted by $\widehat{\mathbf{S}} = \{\mathbf{S}_n | \mathbf{S}_n \in \mathbf{S} \wedge (\mathbf{S}_n \textit{ matches } \mathbf{E})\}$.

3.3. Methods for single-attribute matching

3.3.1. Exclusive method

For an attribute $a_m \in \mathbf{A}$, if the low value of a constraint is greater than v_m, or the high value of the constraint is lower than v_m, it means v_m does not fall into the interval of the constraint, so the corresponding subscription do not satisfy v_m. The basic principle of the exclusive method is to find out all non-matched subscriptions in this way, then exclude them from $\widehat{\mathbf{S}}$, so the rest of the subscriptions in $\widehat{\mathbf{S}}$ whose constraints on a_m all satisfy v_m.

As is shown in Fig. 1, for each attribute, the index structure of DEXIN employs a data structure consisting of two arrays. Each array divides the range of the attribute into multiple slots, and each slot represents a sub-range of the attribute.

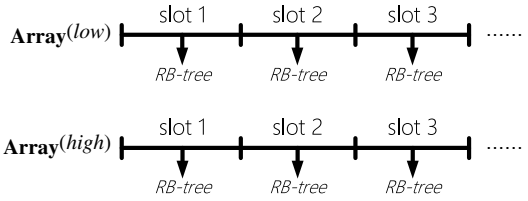

Fig. 1. The index structure of the exclusive method for each attribute.

For a_m, one array $\mathbf{Array}^{(low)}$ includes $|\mathbf{Array}^{(low)}|$ slots, and they are used to store the references of all subscriptions in ascending order of the low values of their constraints on a_m. Subscriptions whose constraint's low value on a_m falls into the range of a slot will be referred to the slot's data structure.

The other array $\mathbf{Array}^{(high)}$ has $|\mathbf{Array}^{(high)}|$ slots. These slots store the references of subscriptions and the ascending sorting is based on the constraint' high values. If the range of a slot contains the constraint's high value of a subscription on a_m, then the slot's data structure will refer the subscription.

The slots are the elements in $\mathbf{Array}^{(low)}$ and $\mathbf{Array}^{(high)}$. Inside a slot, the references of subscriptions are sorted through a RB-tree [24] in ascending order.

Actually, the "Array – RB-tree" structure implements a two-layer indexing mechanism in order to boost the speed for finding out a designated location for a subscription. The searching for a specific slot only has $O(1)$ time complexity because arrays support direct random access and the slot that an attribute value belongs to can be located via a modulo check. RB-tree can provide stable performance of searching, inserting and deleting operations, which all have $O(\log u)$ time complexity, where u is the number of subscriptions stored in the slot.

When a subscription $\mathbf{S}_n$ is subscribed or unsubscribed, all attributes are traversed. For each attribute, the exclusive method finds the two slots that the low value and high value of $\mathbf{S}_n$'s constraint correspond to in the two arrays firstly, and then inserts the reference of the subscription into or removes it from the RB-trees of two slots. Note that, when subscribing, if the low value or high value of a constraint is assigned as Null, the reference of the subscription will not be inserted into the corresponding array ($\mathbf{Array}^{(low)}$ or $\mathbf{Array}^{(high)}$), and the deletion operation in the array is omitted when unsubscribing.

We have the following rules for the exclusive method to determine whether the constraint of a subscription satisfies the attribute value of an event over an attribute:

Rule 1. *Given an attribute a_m, the attribute value v_m of an event $\mathbf{E}$ and the constraint $c_m^{(n)}$ of a subscription $\mathbf{S}_n$, we have:* (1) *if $c_m^{(n)}.lowValueType$ = OPEN, then $c_m^{(n)}$ does not satisfy v_m when $c_m^{(n)}.lowValue \geq v_m$;* (2) *if $c_m^{(n)}.lowValueType$ = CLOSED, then $c_m^{(n)}$ does not satisfy v_m when $c_m^{(n)}.lowValue > v_m$;* (3) *if $c_m^{(n)}.highValueType$ = OPEN, then $c_m^{(n)}$ does not satisfy v_m when $c_m^{(n)}.highValue \leq v_m$;* (4) *if $c_m^{(n)}.highValueType$ = CLOSED, then $c_m^{(n)}$ does not satisfy v_m when $c_m^{(n)}.highValue < v_m$.*

We can further obtain Rule 2 from Rule 1, which is adopted to determine whether a subscription matches an event.

Rule 2. *If at least one of the* 4 *conditions in Rule 1 holds, then $\mathbf{S}_n$ does not match $\mathbf{E}$.*

The design of the exclusive method is based on Rules 1 and 2. Given an attribute a_m, an event $\mathbf{E}$ and the previous partial result set $\widetilde{\mathbf{S}}$, the process of single-attribute matching of the exclusive method is composed of two parts, which are described in Algorithm 1:

Algorithm 1 The Exclusive Method

INPUT: attribute a_m, attribute value v_m of event $\mathbf{E}$ on a_m, partial result set $\widetilde{\mathbf{S}}$ of the previous single-attribute matching.
OUTPUT: partial result set $\widetilde{\mathbf{S}}$ over attribute a_m.
01. Handle $\mathbf{Array}^{(low)}$:
02. Find the slot α that v_m falls into, then find $\mathbf{S}_{t_1}$ in the RB-tree of α, which is the first subscription whose constraint on a_m satisfies $c_m^{(t_1)}.lowValue \geq v_m$;
03. $i \leftarrow 0$;
04. **while** $c_m^{(t_1+i)}.lowValue = v_m$ **do**
05. **if** $c_m^{(t_1+i)}.lowValueType$ = OPEN **then**
06. Remove $\mathbf{S}_{t_1+i}$ from $\widetilde{\mathbf{S}}$;
07. **end if**
08. $i \leftarrow i + 1$;
09. **end while**
10. **for** each $\mathbf{S}_t$ from $\mathbf{S}_{t_1+i}$ to the last subscription in α **do**
11. Remove $\mathbf{S}_t$ from $\widetilde{\mathbf{S}}$;
12. **end for**
13. **for** each slot from the next slot of α to the last slot in $\mathbf{Array}^{(low)}$ **do**
14. **for** each $\mathbf{S}_t$ in the slot **do**
15. Remove $\mathbf{S}_t$ from $\widetilde{\mathbf{S}}$;
16. **end for**
17. **end for**
18. Handle $\mathbf{Array}^{(high)}$:
19. Find the slot β that v_m falls into, them find $\mathbf{S}_{t_2}$ in the RB-tree of β, which is the last subscription whose constraint on a_m satisfies $c_m^{(t_2)}.highValue \leq v_m$;
20. **for** each slot from the first slot in $\mathbf{Array}^{(high)}$ to β **do**
21. **for** each $\mathbf{S}_t$ in the slot **do**
22. Remove $\mathbf{S}_t$ from $\widetilde{\mathbf{S}}$;
23. **end for**
24. **end for**
25. $i \leftarrow 0$;
26. **while** $c_m^{(t_2+i)}.highValue = v_m$ **do**
27. **if** $c_m^{(t_2+i)}.highValueType$ = OPEN **then**
28. Remove $\mathbf{S}_{t_2+i}$ from $\widetilde{\mathbf{S}}$;
29. **end if**
30. $i \leftarrow i - 1$;
31. **end while**
32. **for** each $\mathbf{S}_t$ from $\mathbf{S}_{t_2+i}$ to the first subscription in β **do**
33. Remove $\mathbf{S}_t$ from $\widetilde{\mathbf{S}}$;
34. **end for**

(1) Handle $\mathbf{Array}^{(low)}$: find the slot α that v_m falls into, and locate the first subscription $\mathbf{S}_{t_1}$ in α that the low value of its constraint is greater than or equal to v_m (line 2). Firstly, starting from $\mathbf{S}_{t_1}$, each subscription in α is traversed. The method checks the subscriptions around the starting of traversing. If the low value of a constraint is equal to v_m, then its boundary type is further examined. The subscription is deleted from $\widetilde{\mathbf{S}}$ if the low value type of its constraint is equal to OPEN, otherwise, the subscription is kept (line 3–9). Other subscriptions in the traverse are all deleted (line 10–12). Secondly, all subscriptions in the slots behind α are traversed, and they are deleted from $\widetilde{\mathbf{S}}$ (line 13–17).

(2) Handle $\mathbf{Array}^{(high)}$: find the slot β that v_m falls into, and locate the first subscription $\mathbf{S}_{t_2}$ in β that the high value of its constraint is less than or equal to v_m (line 19). Firstly, all subscriptions in the slots before β are traversed, and they are deleted from $\widetilde{\mathbf{S}}$ (line 20–24). Secondly, starting from $\mathbf{S}_{t_2}$, each subscription in β is traversed reversely. The method checks the subscriptions around the starting of traversing. If the high value of a constraint is equal to v_m, then its boundary type is further examined. The subscription is deleted from $\widetilde{\mathbf{S}}$ if the high value type of its constraint is equal to OPEN, otherwise, the subscription is kept (line 25–31). Other subscriptions in the traverse are all deleted (line 32–34).

When the part (1) and part (2) complete, the subscriptions that remain in $\widetilde{\mathbf{S}}$ are the partial result set over a_m, which can be taken into the next single-attribute matching.

3.3.2. Inclusive method

For an attribute $a_m \in \mathbf{A}$, the basic principle of the inclusive method is to traverse each subscription in $\widetilde{\mathbf{S}}$, and check whether

Algorithm 2 The Inclusive Method

INPUT: attribute a_m, attribute value v_m of event **E** on a_m, partial result set $\widetilde{\mathbf{S}}$ of the previous single-attribute matching.
OUTPUT: partial result set $\widetilde{\mathbf{S}}$ over attribute a_m.
01. **for** each subscription $\mathbf{S}_t$ in $\widetilde{\mathbf{S}}$ **do**
02. **if not** (($c_m^{(t)}.lowValue < v_m$ **and** $c_m^{(t)}.highValue > v_m$) **or** ($c_m^{(n)}.lowValue < v_m$ **and** $c_m^{(n)}.highValue =$ Null) **or** ($c_m^{(n)}.lowValue =$ Null **and** $c_m^{(n)}.highValue > v_m$) **or** ($c_m^{(t)}.lowValue = v_m$ **and** $c_m^{(t)}.lowValueType =$ CLOSED) **or** ($c_m^{(t)}.highValue = v_m$ **and** $c_m^{(t)}.highValueType =$ CLOSED)) **then**
03. Remove $\mathbf{S}_t$ from $\widetilde{\mathbf{S}}$;
04. **end if**
05. **end for**

its constraint on a_m satisfies v_m. If not, then the method excludes the subscription from $\widetilde{\mathbf{S}}$. Thus, after traversing, the rest of the subscriptions in $\widetilde{\mathbf{S}}$ whose constraint on a_m all satisfy v_m.

The inclusive method is conducted by utilizing the structure of $\widetilde{\mathbf{S}}$ itself, so it needs no extra index structures.

In the process of single-attribute matching, based on Rule 3, the inclusive method determines whether the constraint in a subscription on an attribute satisfies the corresponding attribute value of an event.

Rule 3. *Given an attribute a_m, the attribute value v_m of an event* **E** *and the constraint $c_m^{(n)}$ of a subscription $\mathbf{S}_n$, we have:* (1) *if $c_m^{(n)}.lowValue < v_m$ and $c_m^{(n)}.highValue > v_m$, then $c_m^{(n)}$ satisfies v_m;* (2) *if $c_m^{(n)}.lowValue < v_m$ and $c_m^{(n)}.highValue =$* Null, *then $c_m^{(n)}$ satisfies v_m;* (3) *if $c_m^{(n)}.lowValue =$* Null *and $c_m^{(n)}.highValue > v_m$, then $c_m^{(n)}$ satisfies v_m;* (4) *if $c_m^{(n)}.lowValue = v_m$ and $c_m^{(n)}.lowValueType =$* CLOSED, *then $c_m^{(n)}$ satisfies v_m;* (5) *if $c_m^{(n)}.highValue = v_m$ and $c_m^{(n)}.highValueType =$* CLOSED, *then $c_m^{(n)}$ satisfies v_m.*

We can further obtain Rule 4 from Rule 3. Rule 4 is adopted to determine whether a subscription matches an event.

Rule 4. *If at least one of the* 5 *conditions in Rule 3 holds, then $\mathbf{S}_n$ partially matches* **E** *over a_m, otherwise, $\mathbf{S}_n$ does not match* **E**.

The design of the inclusive method is based on Rules 3 and 4. As is described in Algorithm 2, given an attribute a_m, an event **E** and the previous partial result set $\widetilde{\mathbf{S}}$, the process of single-attribute matching of the inclusive method is quite straight forward. The method traverses each subscription in $\widetilde{\mathbf{S}}$ (line 1), and checks whether any one of the 5 conditions in Rule 3 holds (line 2). If the constraint of subscription on a_m does not satisfy v_m, then it is excluded from $\widetilde{\mathbf{S}}$, otherwise, the subscription is kept (line 3). When the method completes, the subscriptions remain in $\widetilde{\mathbf{S}}$ are the partial result set over a_m, which can be taken into the next single-attribute matching.

3.4. Event matching mechanism

The dynamic characteristics in content-based multi-attribute event matching are fully considered by DEXIN. DEXIN abstracts the process of event matching as the process of a serial pipeline, where each single-attribute matching is carried out sequentially. A cost model is proposed to measure the event matching cost of DEXIN, by which the sequences and methods adopted for attributes are determined to decrease the cost.

3.4.1. Pipeline consisting of single-attribute matchings

The event matching process of DEXIN can be compared to a kind of serial pipeline which is composed of multiple machining units. The input and output of each machining unit are connected sequentially. The single-attribute matching over an attribute corresponds to a machining unit in the pipeline, which has two machining methods: the exclusive and inclusive methods. Before a workpiece being processed in the pipeline, the sequences and methods of each machining unit in the pipeline are determined so as to reduce the cost of machining the workpiece in the whole process. A subscription set is a workpiece that is processed in the pipeline, which moves forward and is machined at each machining unit—executing single-attribute matching. The machining unit uses the exclusive method to exclude non-matched subscriptions and the inclusive method to select matched subscriptions from the subscription set. After a workpiece being machined by a previous machining unit, it is transmitted to its next machining unit and then gets processed. In this way, the partial result of single-attribute matching over each attribute is retained and integrated sequentially. The subscription set which is the output from the last machining unit in the pipeline is the result set of event matching.

As is shown in Algorithm 3, the event matching algorithm of DEXIN can be described as follows: Given an event **E** and the subscription set **S** of the system, aiming at decreasing event matching cost, DEXIN first determines the sequence and method (the exclusive or inclusive method) adopted for each attributes by a near-optimal algorithm, which is explained later (line 1). The sequence set is denoted by vector $\mathbf{Q} = \langle q_1, q_2, \ldots, q_M \rangle$, where $q_k \in \mathbf{Q}$ is the ID of the kth attribute in the pipeline; the method set is denoted by vector $\mathbf{D} = \langle d_1, d_2, \ldots, d_M \rangle$, where $d_k \in \mathbf{D}$ represents the method of single-attribute matching for the kth attribute in the pipeline. The exclusive method is adopted if $d_k = 0$, whereas the inclusive method is adopted if $d_k = 1$. In the process of event matching, DEXIN conducts single-attribute matching for each attribute in the pipeline based on **Q** and **D**. Starting from **S**, the output (partial result set $\widetilde{\mathbf{S}}$) of a previous single-attribute matching is taken as the input of its next single-attribute matching. When the last single-attribute matching in the pipeline completes, its output is the result set $\widehat{\mathbf{S}}$ of event matching (line 2–10).

Algorithm 3 Event Matching Algorithm

INPUT: event **E**, subscription set **S** of the system.
OUTPUT: result set $\widehat{\mathbf{S}}$.
01. Determine the sequence vector **Q** of all single-attribute matchings, and the method vector **D** of all attributes.
02. $\widetilde{\mathbf{S}} \leftarrow \mathbf{S}$;
03. **for** $k \leftarrow 1 : M$ **do**
04. **if** $d_k = 0$ **then** // use the exclusive method
05. Run **Algorithm 1** with inputs v_{q_k} and $\widetilde{\mathbf{S}}$;
06. **else** // $d_k = 1$, use the inclusive method
07. Run **Algorithm 2** with inputs v_{q_k} and $\widetilde{\mathbf{S}}$;
08. **end if**
09. **end for**
10. $\widehat{\mathbf{S}} \leftarrow \widetilde{\mathbf{S}}$.

$\widetilde{\mathbf{S}}$ and $\widehat{\mathbf{S}}$ have the same data structure, which is combined by a hash map and a linked list. Both of the exclusive and inclusive methods need to remove non-matched subscriptions from $\widetilde{\mathbf{S}}$, while the hash map has $O(1)$ time complexity for directly locating a specified subscription and deleting it. The inclusive method needs to traverse each subscription in $\widetilde{\mathbf{S}}$, while the linked list supports the traversing with $O(1)$ time complexity.

3.4.2. Cost model

The main work of the exclusive method for an attribute is to traverse the subscriptions in corresponding slots of the two arrays. Hence, the matching cost of the exclusive method mainly lies in the traversing cost. For an attribute and an event, the locations and quantities of subscriptions stored in the two arrays are affected by the distribution of subscriptions on the attribute, and the starting point of traversing in the two arrays is determined by the attribute value of the event on the attribute. Therefore, the traversing cost of the exclusive method varies with respect to different events and distributions of subscriptions.

The main work of the inclusive method for an attribute is to traverse the subscriptions from the partial result of previous

single-attribute matching. Thus, the matching cost of the inclusive method is mainly based on the traversing cost as well. For an attribute and an event, the partial result of the attribute is affected by the attribute value of the event and the distribution of subscriptions on the attribute. For this reason, the traversing cost of the inclusive method is diverse too.

When DEXIN determines the sequences and methods adopted for attributes, it needs to calculate the matching cost of each single-attribute matching. The matching cost of an attribute is approximated as the number of subscriptions in traversing. For the exclusive method, DEXIN records the number of subscriptions stored in each slot, and manages its change caused by subscribing and unsubscribing operations; for the inclusive method, DEXIN records the number of subscriptions stored in the partial result of previous single-attribute matching. Consequently, the calculation of the number of subscriptions can be completed in constant time ($O(1)$ time complexity).

For an attribute a_m, given the attribute value v_m of an event $\mathbf{E}$, and the partial result set $\widetilde{\mathbf{S}}$ of previous single-attribute matching, if the exclusive method is employed for the single-attribute matching over a_m, DEXIN can directly locate the starting-of-traversing slots called $\mathbf{Slot}_{\alpha}^{(low)}$ and $\mathbf{Slot}_{\beta}^{(high)}$ in $\mathbf{Array}^{(low)}$ and $\mathbf{Array}^{(high)}$, then the matching cost can be approximated as $\sum_{r=\alpha}^{|\mathbf{Array}^{(low)}|}|\mathbf{Slot}_r^{(low)}| + \sum_{r=1}^{\beta}|\mathbf{Slot}_r^{(high)}|$, namely, the number of subscriptions from all slots in the traversing of both arrays; if the inclusive method is employed for the single-attribute matching over a_m, then the matching cost is approximated as $|\widetilde{\mathbf{S}}|$, namely, the number of subscriptions in $\widetilde{\mathbf{S}}$. For simplicity, we normalize the matching costs of the two methods by $|\mathbf{S}|$, so the matching cost of the exclusive method can be expressed as

$$\mu_m^{(ex)} = \left(\sum_{r=\alpha}^{|\mathbf{Array}^{(low)}|}|\mathbf{Slot}_r^{(low)}| + \sum_{r=1}^{\beta}|\mathbf{Slot}_r^{(high)}|\right) \Big/ |\mathbf{S}|. \tag{1}$$

The matching cost of the inclusive method can be expressed as

$$\mu_m^{(in)} = |\widetilde{\mathbf{S}}|/|\mathbf{S}|. \tag{2}$$

Given an event $\mathbf{E}$ and the subscription set $\mathbf{S}$ of the system, the matching rate ρ_m of $\mathbf{E}$ over an attribute a_m is the proportion of subscriptions whose constraints on a_m satisfy v_m in $\mathbf{S}$, whereas, the non-matching rate δ_m of $\mathbf{E}$ over a_m is the proportion of subscriptions whose constraints on a_m do not satisfy v_m in $\mathbf{S}$. ρ_m and δ_m can be obtained approximately from the index structure of the exclusive method:

$$\rho_m = 1 - \delta_m = 1 - \frac{\sum_{r=\alpha}^{|\mathbf{Array}^{(low)}|}|\mathbf{Slot}_r^{(low)}| + \sum_{r=1}^{\beta}|\mathbf{Slot}_r^{(high)}|}{|\mathbf{S}|} \tag{3}$$

where r_1 and r_2 are the sequence number of the corresponding slots in the two arrays, respectively.

Given an event $\mathbf{E}$ and the subscription set $\mathbf{S}$ of the system, we assume the sequence vector $\mathbf{Q}$ and method vector $\mathbf{D}$ are obtained previously. For an attribute a_{q_k}, which is the kth attribute in the pipeline, the matching costs of the exclusive and inclusive methods can be further expressed via δ_{q_k} and ρ_{q_k}, respectively. As for the exclusive method, we have

$$\mu_{q_k}^{(ex)} = \delta_{q_k} = 1 - \rho_{q_k}. \tag{4}$$

The inclusive method needs to traverse the subscriptions in the partial result of its previous single-attribute matching, hence, the matching cost of the inclusive method relies on the previous partial results. We assume the distributions of subscriptions on different attributes are independent of each other. In the process of event matching, the input subscription set of the 1st single-attribute matching in the pipeline has $|\widetilde{\mathbf{S}}| = |\mathbf{S}|$, and its partial result set has $|\mathbf{S}^{(q_1)}| = \rho_{q_1}|\mathbf{S}|$. The partial result set of the 2nd single-attribute matching has $|\mathbf{S}^{(q_2)}| = \rho_{q_2}|\mathbf{S}^{(q_1)}| = \rho_{q_1}\rho_{q_2}|\mathbf{S}|, \ldots$, so forth, hence, the partial result set of the $(k-1)$th single-attribute matching has $|\mathbf{S}^{(q_{k-1})}| = \rho_{q_{k-1}}|\mathbf{S}^{(q_{k-2})}| = \rho_{q_1}\rho_{q_2}\ldots\rho_{q_{k-1}}|\mathbf{S}|$. Accordingly, for the kth single-attribute matching, the input subscription set is the partial result set of the $(k-1)$th single-attribute matching, so $|\widetilde{\mathbf{S}}| = |\mathbf{S}^{(q_{k-1})}| = (\prod_{l=q_1}^{q_{k-1}}\rho_l)|\mathbf{S}|$. Therefore, we have

$$\mu_{q_k}^{(in)} = |\widetilde{\mathbf{S}}|/|\mathbf{S}| = \begin{cases} 1, & k = 1 \\ \prod_{l=q_1}^{q_{k-1}}\rho_l, & k > 1 \end{cases}. \tag{5}$$

From Formula (4), it can be seen that $\mu_{q_k}^{(ex)}$ is irrelevant to $|\widetilde{\mathbf{S}}|$, so $\mu_{q_k}^{(ex)}$ does not rely on the partial results of any single-attribute matchings. For the reason above, it can be inferred that the matching cost of the exclusive method over an attribute is independent of the attribute's sequence in the pipeline.

In contrast, from Formula (5), $\mu_{q_k}^{(in)}$ is composed by the previous matching rates from the 1st single-attribute matching to the $k-1$th single-attribute matching in the pipeline. Thus, the matching cost of the inclusive method is dependent on the attribute's sequence in the pipeline.

Besides, the matching rate has opposite impact on the matching costs of the two methods. According to Formula (4) and (5), the matching cost of the exclusive method decreases as the matching rate grows, but the matching cost of the inclusive method increases as the matching rate grows. Thus, the exclusive method is more suitable to the single-attribute matchings with high matching rates, whereas, the inclusive method is more suitable to the single-attribute matchings with low matching rates.

Based on above analysis, the event matching cost of DEXIN is the sum of matching costs of all single-attribute matchings in the pipeline, which can be expressed as $\mu = \sum_{k=1}^{M}\mu_{q_k}$. The vector $\mathbf{D}$ uses 0 or 1 to mark the exclusive method or inclusive method, by which we can partition the event matching cost into two parts: the sum of matching costs via the exclusive method, and the sum of matching costs via the inclusive method. They are denoted by $\sum_{k=1}^{M}((1-d_k)\mu_{q_k}^{(ex)})$ and $\sum_{k=1}^{M}(d_k\mu_{q_k}^{(in)})$, respectively. Therefore, the cost model of DEXIN can be formulated as

$$\mu = \sum_{k=1}^{M}\mu_{q_k} = \sum_{k=1}^{M}\left((1-d_k)\mu_{q_k}^{(ex)} + d_k\mu_{q_k}^{(in)}\right). \tag{6}$$

3.5. Near-optimal algorithms

In order to promote the performance of event matching, the aim of DEXIN is to determine the vector $\mathbf{Q}$ and $\mathbf{D}$ for an event and the subscriptions in the system, through which the minimization of event matching cost can be approached. According to Formula (6), we have

$$\min_{\mathbf{Q},\mathbf{D}}\mu = \min_{\mathbf{Q},\mathbf{D}}\left(\sum_{k=1}^{M}\left((1-d_k)\mu_{q_k}^{(ex)} + d_k\mu_{q_k}^{(in)}\right)\right). \tag{7}$$

Formula (7) belongs to a combinatorial optimization problem [25]. Actually, it is very complicated to solve directly. Enormous expenditures will be paid to find out the optimal solution $\mathbf{Q}^{(*)}$ and $\mathbf{D}^{(*)}$ because a large number of feasible solutions need to be traversed and checked. In order to reduce the computation cost, we observe several features that appear in the optimal solution by analyzing Formula (7).

Observation 1. *In the optimal solution, the first single-attribute matching always employs the exclusive method.*

Proof of Observation 1. We assume the first single-attribute matching is carried out over an attribute a_x, the matching rate of a_x is ρ_x, and the non-matching rate of a_x is δ_x. Whether the exclusive or inclusive method is adopted, the partial result from a_x always has $|\widetilde{\mathbf{S}}| = \rho_x|\mathbf{S}|$, so it has no affects on the rest single-attribute matchings. Supposing the sum of matching costs from the second single-attribute matching to the M th is $\widetilde{\mu}$, then the event matching cost is $\delta_x + \widetilde{\mu}$ if the exclusive method is employed for a_x, whereas the event matching cost is $|\widetilde{\mathbf{S}}|/|\mathbf{S}| + \widetilde{\mu} = 1 + \widetilde{\mu}$ if the inclusive method is employed for a_x. Obviously, $\delta_x + \widetilde{\mu} \leq 1 + \widetilde{\mu}$ holds due to $0 \leq \delta_x \leq 1$.

Observation 2. *If both of the exclusive and inclusive methods are applied in the optimal solution, the sequence of attributes in the pipeline is divided into a pre-part and a post-part. The exclusive method is employed for all attributes in the pre-part, and the inclusive method is employed for all attributes in the post-part.*

Proof of Observation 2. We assume the sum of matching costs via the exclusive method is $\mu^{(ex)}$, the product of matching rates of the attributes which employ the exclusive method is φ, and the number of subscriptions which employ the inclusive method is X, the IDs of these attributes are denoted by the sequence $\pi_1, \pi_2, \ldots, \pi_X$ ordered by their locations in the pipeline.

Thus, the event matching cost $\mu = \mu^{(ex)} + \varphi(1 + \rho_{\pi_1}(1 + \rho_{\pi_2}(\ldots(1 + \rho_{\pi_{X-1}}))))$.

Now there is a new attribute with ID π_y which is chosen to employ the exclusive method. If it is inserted into a location after π_x in the sequence, which means the sequence is bisected by this attribute, then the event matching cost $\mu^{(\prime)} = \mu^{(ex)} + \delta_{\pi_y} + \varphi(1 + \rho_{\pi_1}(1+\rho_{\pi_2}(\ldots(1+\rho_{\pi_x}))))+\varphi\rho_{\pi_y}\rho_{\pi_1}\rho_{\pi_2}\ldots\rho_{\pi_x}(1+\rho_{\pi_{x+1}}(\ldots(1+\rho_{\pi_{X-1}})))$. If it is inserted before π_1, then the event matching cost $\mu^{(\prime\prime)} = \mu^{(ex)} + \delta_{\pi_y} + \varphi\rho_{\pi_y}(1 + \rho_{\pi_1}(1 + \rho_{\pi_2}(\ldots(1 + \rho_{\pi_{X-1}}))))$.

Obviously, we have $\mu^{(\prime\prime)} < \mu^{(\prime)}$, so the attributes which employ the exclusive method should always locates before the attributes which employ the inclusive method. Therefore, the Observation 2 is proved.

Observation 3. *If both of the exclusive and inclusive methods are applied in the optimal solution, in the pipeline, the attributes for which the inclusive method is employed are in ascending order of matching rate.*

Proof of Observation 3. We assume the number of subscriptions which employ the inclusive method is X, the IDs of attributes are denoted by the sequence $\pi_1, \pi_2, \ldots, \pi_X$ in ascending order of matching rate, the sum of matching costs via the exclusive method is $\mu^{(ex)}$, and the product of matching rates of the attributes which employ the exclusive method is φ.

(1) When $X = 1$, the Observation 3 always holds.
(2) Supposing Observation 3 holds when $X = k > 1$, the current sequence is $\pi_1, \pi_2, \ldots, \pi_k$, where $\rho_{\pi_1} \leq \rho_{\pi_2} \leq \cdots \leq \rho_{\pi_k}$. Now there is a new attribute with ID z which is chosen to employ the inclusive method, and which satisfy $\ldots \leq \rho_{\pi_x} \leq \rho_z \leq \rho_{\pi_{x+1}} \leq \ldots$.

If the attribute is inserted after π_{x-1} in the sequence, then the event matching cost $\mu^{(x-1)} = \mu^{(ex)} + \varphi(1 + \rho_{\pi_1}(\ldots(1 + \rho_{\pi_{x-1}}(1 + \rho_{\pi_z}(1 + \rho_{\pi_x}(\ldots(1 + \rho_{\pi_{k-1}})))))))$.

If the attribute is inserted after π_x in the sequence, then the event matching cost $\mu^{(x)} = \mu^{(ex)} + \varphi(1 + \rho_{\pi_1}(\ldots(1 + \rho_{\pi_x}(1 + \rho_{\pi_z}(1 + \rho_{\pi_{x+1}}(\ldots(1 + \rho_{\pi_{k-1}})))))))$.

If the attribute is inserted after π_{x+1} in the sequence, then the event matching cost $\mu^{(x+1)} = \mu^{(ex)} + \varphi(1 + \rho_{\pi_1}(\ldots(1 + \rho_{\pi_{x+1}}(1 + \rho_{\pi_z}(\ldots(1 + \rho_{\pi_{k-1}})))))))$.

Algorithm 4 Low-Matching-Rate-First-Fit Algorithm

INPUT: event **E**, attribute set **A**, subscription set **S** of the system.
OUTPUT: matching order vector **Q**, matching method vector **D**.
01. According to Formula (3), compute ρ_m of each a_m in **A**, then sort **A** in ascending order of matching rate;
02. Choose the first attribute a_m in **A**, then let $q_1 = a_m$ and $d_1 = 0$;
03. $k \leftarrow 2$;
04. $\epsilon \leftarrow 1$;
05. **while** $k \leq |\mathbf{A}|$ **do**
06. Choose the kth attribute a_m in **A**;
07. $\epsilon \leftarrow \epsilon \cdot \rho_{q_{k-1}}$; // accumulating matching rate
08. **if** $\epsilon > \delta_m$ **then**
09. Let $q_k = a_m$ and $d_k = 0$;
10. **else**
11. **jump to** line 15;
12. **end if**
13. $k \leftarrow k + 1$;
14. **end while**
15. **while** $k \leq |\mathbf{A}|$ **do**
16. Choose the kth attribute a_m in **A**;
17. Let $q_k = a_m$ and $d_k = 1$;
18. $k \leftarrow k + 1$;
19. **end while**

Obviously, we have $\mu^{(x)} \leq \mu^{(x-1)}$ and $\mu^{(x)} \leq \mu^{(x+1)}$ since $\rho_{\pi_x} \leq \rho_z \leq \rho_{\pi_{x+1}}$. Moreover, $\mu^{(x-1)} \leq \mu^{(x-2)} \leq \cdots \leq \mu^{(0)}$ and $\mu^{(x+1)} \leq \mu^{(x+2)} \leq \cdots \leq \mu^{(k)}$. It can be seen that the x th location is optimal for the attribute, hence, the Observation 3 holds when $X = k + 1$.

Therefore, the Observation 3 is proved via the mathematical induction above.

Based on above observations, we propose a near-optimal algorithm—Low-Matching-Rate-First-Fit Algorithm, which belongs to the greedy algorithm. Although, the solution from the algorithm is not optimal, it has very small computation cost, and the solution can be obtained in a $O(M \log M + M)$ time.

The Low-Matching-Rate-First-Fit Algorithm handles the attributes with low matching rate in priority. As is described in Algorithm 4, the algorithm first sorts the attributes in ascending order of matching rate. Then, according to Observation 1, the first in the sorted attributes is chosen as the first attribute in the pipeline, and employs the exclusive method. Subsequently, the algorithm chooses the next attribute from the sorted attributes, and calculates its accumulated matching rate, which represents the normalized number of subscriptions that the inclusive method needs to traverse and check over the attribute, namely, the matching cost of the inclusive method over the attribute. If the accumulated matching rate is greater than the matching rate, the attribute is chosen as the next attribute in the pipeline, and employs the exclusive method. Then algorithm chooses the next attribute from the sorted attributes and handles so forth. If the accumulated matching rate is less than or equal to the matching rate, the attribute and the rest attributes in the sorted attributes all employ the inclusive method according to Observation 2. Based on Observation 3, each of them is chosen sequentially as the next attribute in the pipeline since they are sequenced in ascending order of matching rate.

4. Experiment results and performance evaluation

In this section, the performance of DEXIN is evaluated comprehensively in different scenarios. The event matching time of DEXIN is measured with different criteria, and it is compared with 5 state-of-the-art reference algorithms (TAMA [11], H-Tree [12], BE-Tree [13], OpIndex [14] and REIN [15]), where H-Tree, BE-Tree, OpIndex, REIN and DEXIN belong to exact event matching algorithms, whereas, TAMA is an approximate one, which may output false positive results. The maintenance costs of DEXIN and the reference algorithms are tested as well, which include the subscribing/unsubscribing time and memory consumption. Additionally, the near-optimal algorithm is also analyzed in detail,

and it is also compared with the optimal algorithm. We also compare the performance of DEXIN itself, and DEXIN when using either the inclusive method or the exclusive method. All experiments are carried out 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. All code in experiments is written in Java language. Parallelism is not used in experiments.

In our experiments, the value domains of all attributes are normalized to [0, 1]. In order to prepare subscription dataset and event dataset for the performance evaluations with different purposes, the attribute values of constraints and events are generated from the value domains of attributes with varying cardinalities, which are denoted by η. The proportion of point-based and range-based constraints in a subscription is also configurable, which is defined as λ. The width of range-based constraints is given by θ, that is, the low values of range-based constraints are randomly generated from $[0, 1 - \theta]$, and the high values locate in $[\theta, 1]$ correspondingly. Aiming at generating subscriptions with a fixed matching rate ξ, we employ a method which is similar to the method used in [13]. Most of our experiments are conducted with subscriptions and events generated in Uniform distribution, additionally, we also pick Zipf distribution to evaluate the performance of algorithms under skewed distributions. The settings of above parameters are summarized in Table 1.

Generally speaking, for DEXIN, the traversing in $\mathbf{Array}^{(low)}$ and $\mathbf{Array}^{(high)}$ will slightly benefit with the increase of the number of slots, τ. It is because the more slots are, the fewer subscriptions are stored in a slot, so DEXIN could take shorter time to locate the starting location of traversing. However, the memory consumption will grow with the increase of τ since every slot maintains a RB-tree. Therefore, we pick $\tau = 1500$ in all experiments.

4.1. Event matching time

The event matching time is the most important metric for evaluating the performance of event matching algorithms. Experiments are conducted to compare the matching time of DEXIN with that of other 5 algorithms in the scenarios of 7 different criteria, which are (1) **the number of subscriptions**, N, (2) **the number of attributes**, M, (3) **the width of range-based constraints**, θ, (4) **the distribution of constraints**, Uniform and Zipf, (5) **the cardinality of attribute values**, η, (6) **the proportion of point-based constraints**, λ, and (7) **the matching rate of subscriptions**, ξ. The results of event matching time are obtained from multiple repeated experiments. In each experiment, we randomly input 1000 events. The parameters are configured based on TABLE Table 1 unless stated clearly.

4.1.1. Different number of subscriptions

As is shown in Fig. 2, we measure the event matching time of the 5 algorithms with different numbers of subscriptions. In the experiments, N increases from 0.25 million to 1.75 million, while the other settings are $M = 10, \eta = 10^4, \lambda = 10\%$, and θ is randomly picked from [0.2, 0.8]. Overall, for the 6 algorithms, their event matching time all increases with N. In Fig. 2, it can be seen that DEXIN is the best over the other 5 algorithms, and its event matching time is the minimum at each tick of N on x-axis. In the results, the average event matching time of DEXIN is 20.18%, 64.28%, 68.36%, 26.4%, 38.49% of that of TAMA, H-Tree, BE-Tree, OpIndex and REIN, respectively. Especially, for the worst algorithm TAMA, DEXIN approaches a 82.13% reduction in the experiment with $N = 1$ million, and for the next best algorithm BE-Tree, DEXIN exhibits a 63.01% better event matching time in the experiment with $N = 0.75$ million.

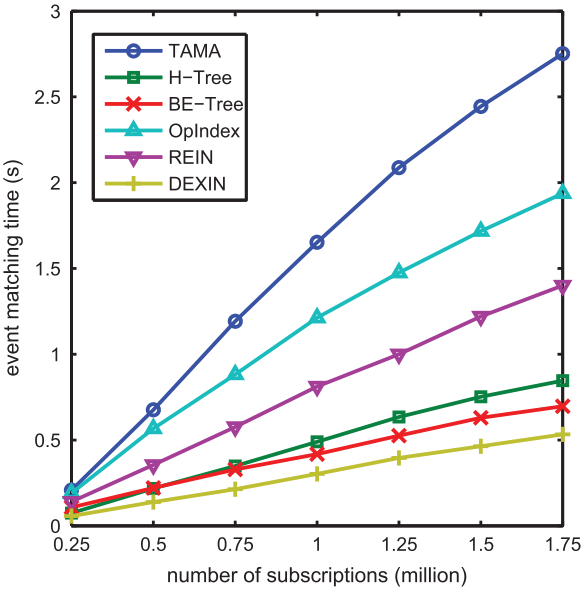

Fig. 2. Event matching time with different numbers of subscriptions.

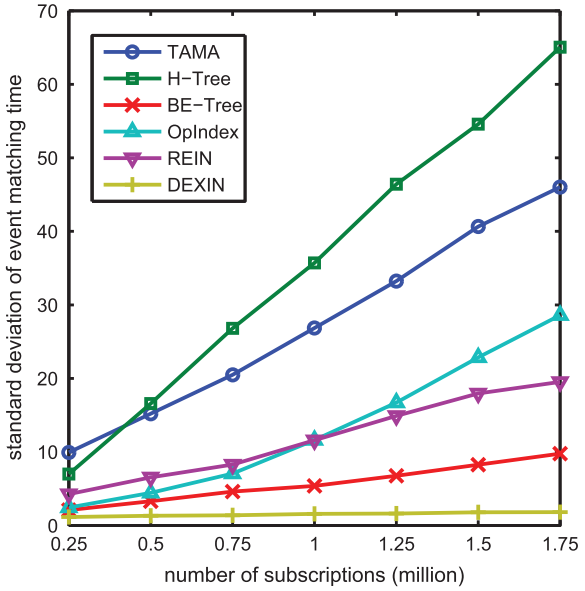

Fig. 3. Standard deviation of event matching time with different numbers of subscriptions.

Among the 6 algorithms, DEXIN's sensitivity to N is the lowest since the slope of its curve in Fig. 2 is the lowest, which is a superiority of DEXIN. It can be explained that DEXIN makes the search space for each incoming event shrink efficiently, so it further decreases the event matching cost and speed up the event matching process.

The stability of event matching time is another important concern. For the 6 algorithms, their standard deviations of event matching time at every tick of N on x-axis are also measured in above experiments. The results are shown in Fig. 3. Overall, the standard deviation of each algorithm increases with N, because the uncertainty of event matching rises as N grows. It can be observed that DEXIN always has the lowest standard deviations at all ticks of N from 0.25 million to 1.75 million. The mean of the standard deviations of TAMA, H-Tree, BE-Tree, OpIndex, REIN and DEXIN are 27.49, 36.02, 5.74, 13.39, 11.87 and 1.52, respectively. DEXIN is the most stable among them, and its curve is nearly flat, which means the shortest and the longest event matching time are quite near, so

Table 1
The parameters used in experiments and their settings.

Name	Meaning	Value
N	The number of subscriptions	[0.25M, 1.75M]
M	The number of attributes	[2, 22]
η	The cardinality of attribute values	$[10^2, 10^6]$
λ	Proportion of point-based constraints in a subscription	[10%, 90%]
θ	The width of range-based constraints	[0.1, 0.9]
ξ	The matching rate of subscriptions	[0.1%, 20%]
τ	The number of slots in the arrays of the exclusive method in DEXIN	1500

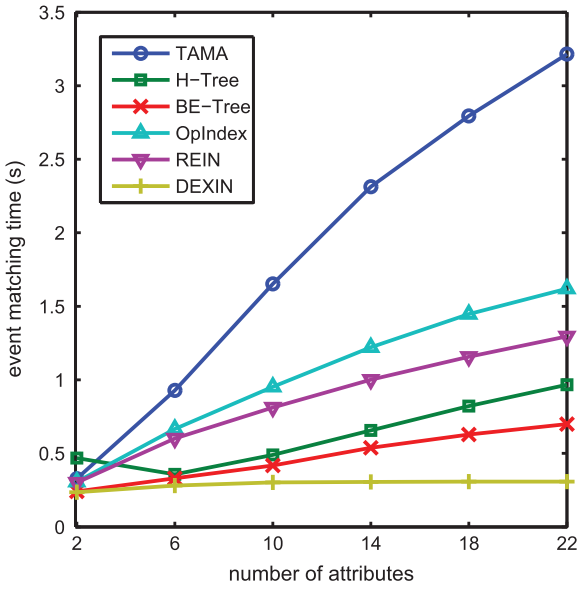

Fig. 4. Event matching time with different numbers of attributes.

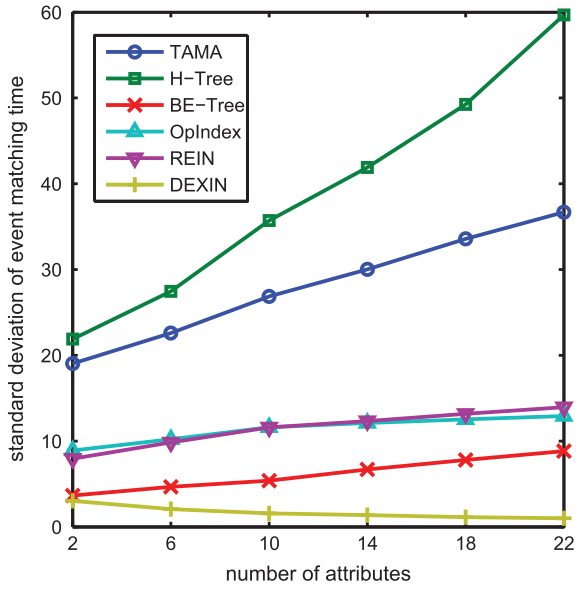

Fig. 5. Standard deviation of event matching time with different numbers of attributes.

DEXIN can keep high stability of event matching performance with different N.

4.1.2. Different number of attributes

The number of attributes in subscriptions and events also affects the performance of event matching. The more M is, the larger the search space of event matching will be. Thus, as is shown in Fig. 4, experiments are carried out to measure the event matching time of the 5 algorithms with different M from 2 to 22, where the other settings are $N = 10^6$, $\eta = 10^4$, $\lambda = 10\%$, and θ is random chosen from [0.2, 0.8]. Actually, the number of matched subscriptions deceases with the increase of M. In general, the event matching time of TAMA, BE-Tree, OpIndex and REIN keeps growing with M since their event matching mechanisms are lack of joint processing and optimization for single-attribute matchings, which results in event matching costs simply accumulating when M increases. The event matching time of H-Tree decreases when $M < 6$, and then increases when $M > 6$, because the dropping of matching rate dominates in the former, whereas, in the latter, the expansion of search space is the leading factor. In contrary to other 5 algorithms, the event matching time of DEXIN rises slightly before $M = 6$, and then keeps nearly invariant. The search space of DEXIN shrinks after each single-attribute matching in the serial pipeline, so extra matching costs brought by the increase of attributes can be reduced to a large extent. The last single-attribute matchings in the pipeline just need to traverse and check a few subscriptions via the inclusive method. This feature is a great advantage of DEXIN, which means DEXIN can support multiple attributes without severely suffering from performance degradation.

Additionally, for the 6 algorithms, their standard deviations of event matching time for the above experiments are shown in Fig. 5. The average standard deviations of TAMA, H-Tree, BE-Tree, OpIndex, REIN and DEXIN are 28.13, 39.32, 6.17, 11.38, 11.47 and 1.7, respectively, which proves DEXIN is still the most stable even for variable M. It can be seen that the curve of DEXIN decreases slowly with the increase of M, because as M grows, there are more opportunities for DEXIN to optimize the sequences and methods for single-attribute matchings. The curves of other 5 algorithms all grow as M increases since the uncertainty of event matching for each subscription rises as M increases.

4.1.3. Different widths of range-based constraints

The width of range-based constraints determines the constraints selectivities of subscriptions. The probabilities of being matched are low for subscriptions with narrow constraints, but are high for those with wide constraints. The performance of event matching is affected by the changes of θ. We measure the event matching time of the 6 algorithms with the increase of θ from 0.1 to 0.9, and the other settings are $N = 10^6$, $M = 10$, $\eta = 10^4$, $\lambda = 10\%$. The results are shown in Fig. 6, where it can be seen that the event matching time of BE-Tree and OpIndex grows slowly, because the event matching of which mainly relies on the number of subscriptions, so it is insensitive to θ. The event matching time of TAMA rapidly increases with θ although it performs well when $\theta \leq 0.1$. Narrow constraints are stored in the layered structure of TAMA with only a few duplicates, which alleviates the pressure of event matching because there is a small scale of subscriptions that need to be checked. When $\theta \leq 0.3$, the event matching time of H-Tree is the lowest, then it increases exponentially. A subscription is splits into multiple duplicates when the width of its constraints exceeds the capacity of cells in H-Tree, which enlarges the search space to a great extent. The event matching time of REIN

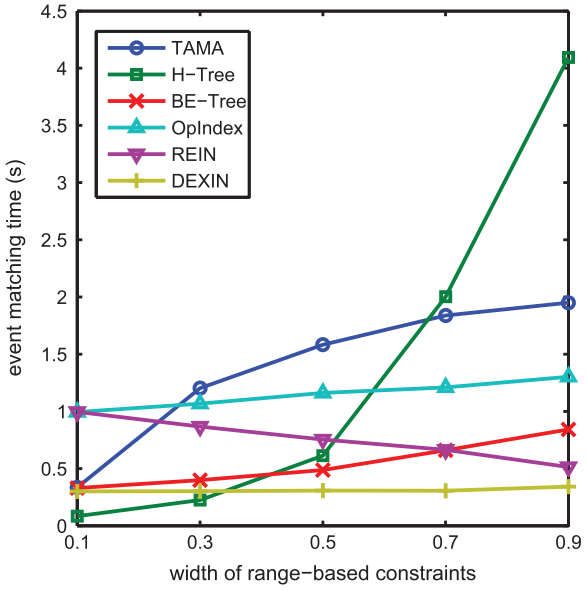

Fig. 6. Event matching time with different width of range-based constraints.

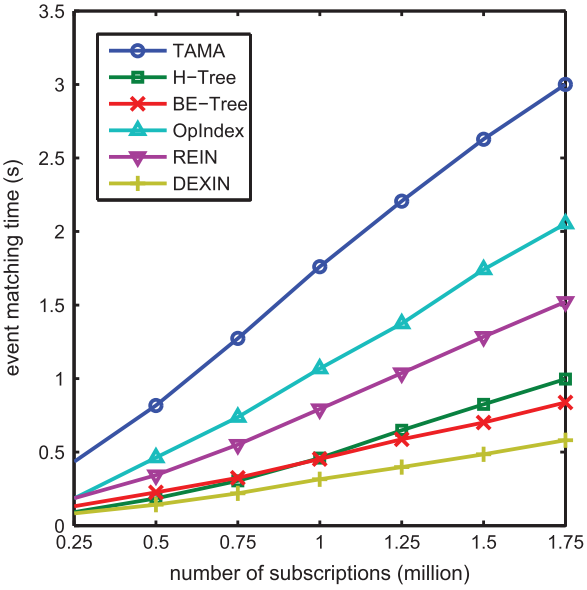

Fig. 7. Event matching time with different numbers of subscriptions under Zipf distribution.

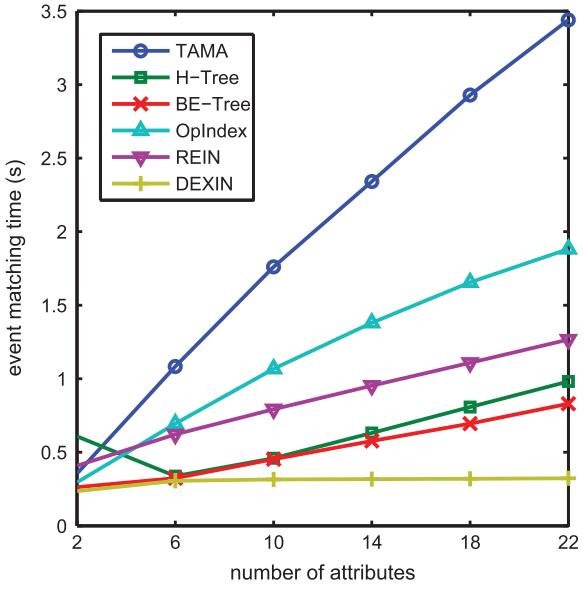

Fig. 8. Event matching time with different numbers of attributes under Zipf distribution.

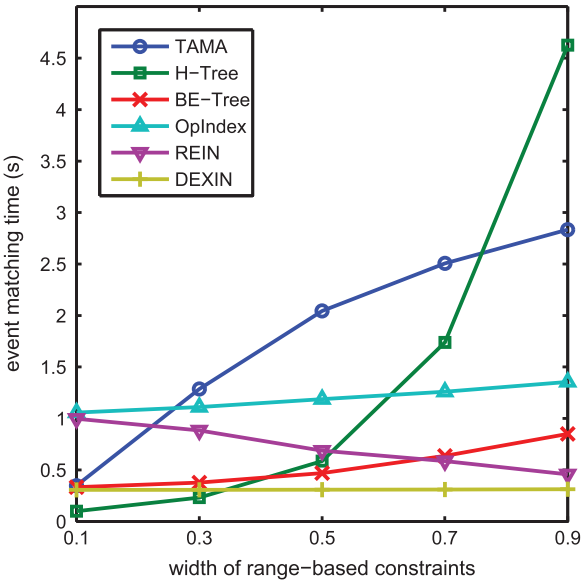

Fig. 9. Event matching time with different widths of constraints under Zipf distribution.

decreases with the increase of θ. REIN carries out event matching through excluding non-matched subscriptions, and there is only a small amount of non-matched subscriptions when constraints in subscriptions are wide, thus the matching cost drops as θ expands.

Differently, the event matching time of DEXIN keeps nearly unchanged, because when θ is low, attributes with low matching rates are chosen preferentially in the pipeline which makes partial results small enough to be quickly processed by the inclusive method, and when θ is high, the matching cost is reduced by the exclusive method since the amount of non-matched subscriptions is small. Therefore, DEXIN fits for the subscriptions with variable widths of constraints quite well, which is very common in most publish/subscribe systems.

Moreover, the average standard deviations of event matching time of TAMA, H-Tree, BE-Tree, OpIndex, REIN and DEXIN are 23.25, 35.64, 8.13, 12.02, 7.33 and 1.09, respectively, where DEXIN is still the best.

4.1.4. *Different distributions of constraints*

The distribution of constraints is relevant to the locations in the index structures where subscriptions are stored. Thus, the design of index structure determines the tolerability of an algorithm to different distributions. We measure the event matching time of DEXIN under Zipf distribution. Experiments are conducted with different numbers of subscriptions, different numbers of attributes and different widths of range-based constraints. The results are shown in Figs. 7–9, respectively. Overall, compared with the results under Uniform distribution which are shown in Figs. 2 and 4 and Fig. 6, the event matching time of all algorithms under Zipf distribution all grows to some extent. The skewness of Zipf distribution makes a large number of subscriptions stored in a local area of the index structure, the event matching may take some extra time in this area so that the event matching time is prolonged.

Among all 6 algorithm, it can be found that DEXIN has the lowest impact on distributions of subscriptions, which suggest the high stability of DEXIN again. There are mainly two reasons, on

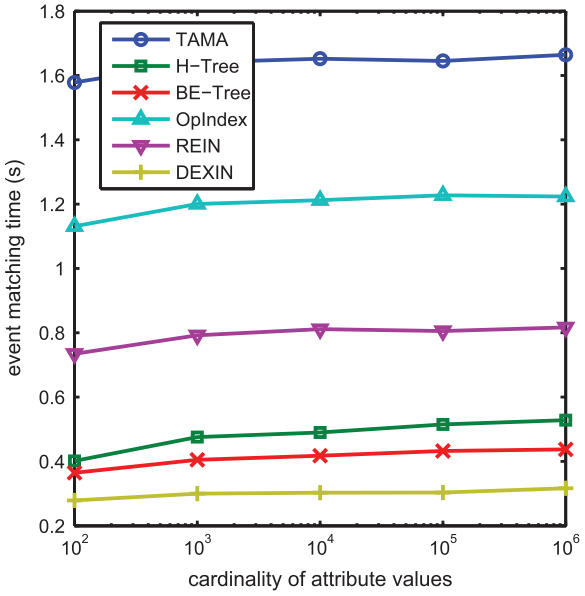

Fig. 10. Event matching time with different cardinalities of attribute values.

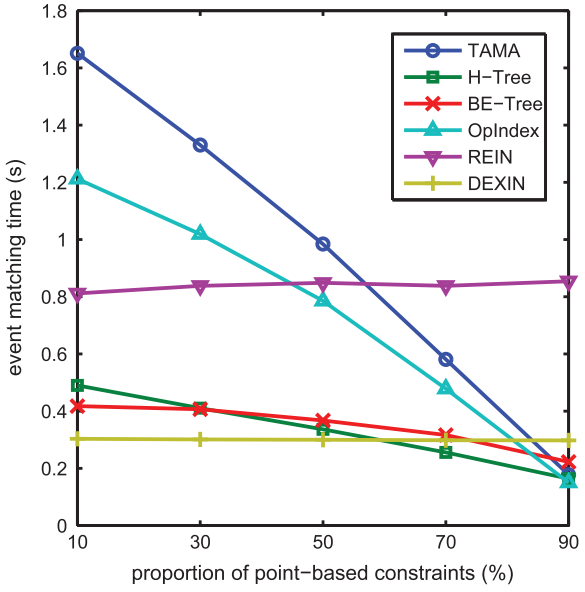

Fig. 11. Event matching time with proportions of point-based constraints.

the one hand, the effects of different distributions are alleviated by allocating the number of slots large enough ($\tau = 1500$); on the other hand, the time for searching a specified subscription is logarithmic because subscriptions are stored in RB-trees. Thus, traversing and checking are only conducted in a single slot, which only contains a small amount of subscriptions. The search in a slot has logarithmic time complexity, which can provide high search efficiency even when a large number of subscriptions stored.

4.1.5. Different cardinalities of attribute values

The cardinalities of attribute values affect the attribute value assignments of subscriptions and events. As η grows, there are more diversities on values of point-based constraints, low values and high values of range-based constraints, and values of event on each attribute, thus, extra requirements are needed to organize new values, which poses a challenge on the index structures of event matching algorithms. Fortunately, all 6 algorithms can provide full supports for variable η.

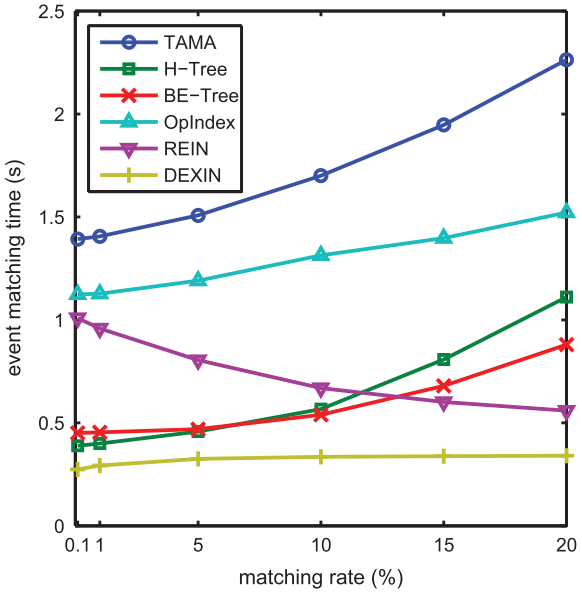

Fig. 12. Event matching time with different matching rates of subscriptions.

η could also have impacts on matching rate to some extent. Experiments are conducted to evaluate the event matching time of all 6 algorithms with the increase of η from 10^2 to 10^6. The other settings are $N = 10^6$, $M = 10$, $\lambda = 10\%$, θ is randomly picked from [0.2, 0.8]. The results are shown in Fig. 10. It can be found the event matching time of all 6 algorithms increases very slowly when $\eta < 10^3$, then keeps stable. There are some slight rises and falls of the curves due to the slight variance of event matching rate caused by η. Overall, the performance of DEXIN is the best, and it outperforms other algorithms by more than 26.88%.

4.1.6. Different proportions of point-based constraints

For some algorithms, the event matching time may vary at different ratios of point-based constraints and range-based constraints. Experiments are carried out to investigate the event matching time of all 6 algorithms with different proportions of point-based constraints from 10% to 90%, where the other settings are $N = 10^6$, $M = 10$, $\eta = 10^4$, θ is random generated from [0.2, 0.8]. As shown in Fig. 11, the event matching time of TAMA, H-Tree, BE-Tree and OpIndex all decreases as λ grows. For TAMA and H-Tree, it is because point-based constraints do not need to be stored into multiple cells, so the number of search paths decreases as λ grows. For BE-Tree, the main reason lies in that point-based constraints have no intervals so that they can be associated with the smallest bucket in a c-directory, thus, the number of search paths decreases as well. For OpIndex, the hash map mechanism is employed to filter out non-matched point-based constraints directly, which can reduce the search cost directly. The curves of the event matching time of REIN and DEXIN keep nearly flat, since there are no differences between the event matching for point-based and range-based constraints. However, the performance of DEXIN is still the best when $\lambda \leq 60\%$, and actually, there are very few point-based constraints appearing in practical systems.

4.1.7. Different matching rates of subscriptions

We also analyze the impact of different matching rates of subscription on the event matching time of all 6 algorithms. The subscriptions with a designated ξ are generated as follows: Initially, we generate a certain number of events. For each event, a number of new subscriptions, which all match the event, are created based on the attribute values of the event. For each subscription bound to the event, firstly, we randomly choose an

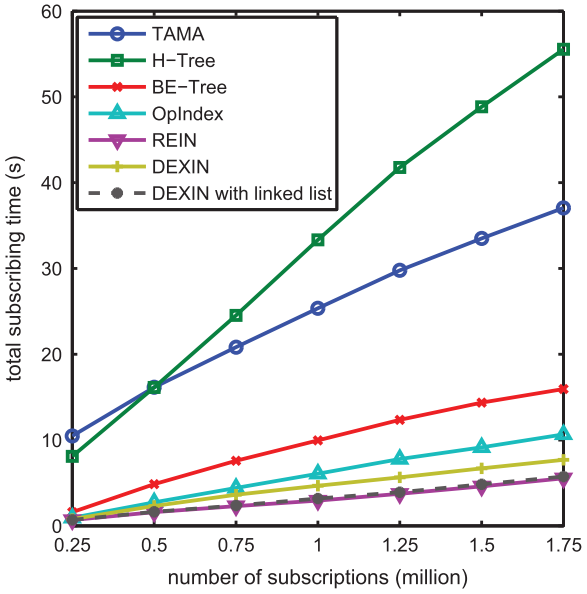

Fig. 13. Total subscribing time with different numbers of subscriptions.

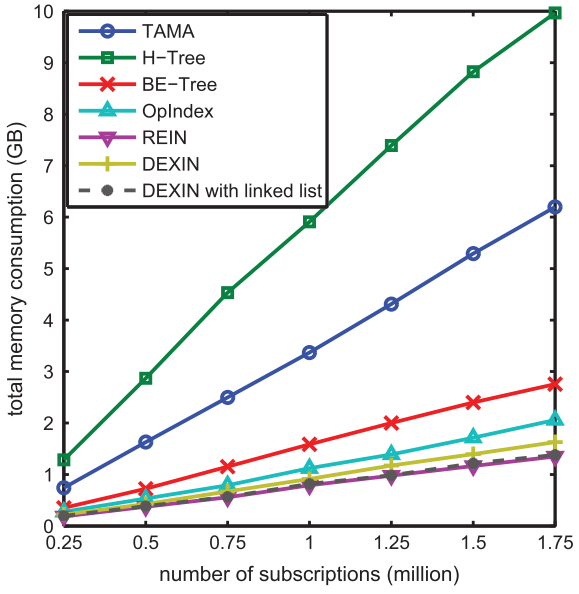

Fig. 14. Memory consumption with different numbers of subscriptions.

attribute, and produce a range-based or point-based constraint that only cover the attribute value on that attribute, then we produce the constraints for other attributes, and each constraint must cover the attribute value on its corresponding attribute at least. Thus, we can set different matching rates by varying the number of subscriptions for each event that is generated initially. The results are shown in Fig. 12, where ξ grows from 0.1% to 20%, and other settings are $N = 10^6$, $M = 10$, $\eta = 10^4$, $\lambda = 10\%$, θ is randomly picked from [0.2, 0.8]. As ξ grows from 0.1% to 20%, the event matching time of TAMA, H-Tree, BE-Tree and OpIndex all increases, due to the rise of the number of partially matched subscriptions. Whereas, the event matching time of REIN decreases because the main cost of REIN is to exclude non-matched subscriptions. The performance of DEXIN is quite stable and it wins the next best algorithm - BE-Tree by 42.46% on average. The reason is that the optimization is done for each incoming event, and the cost of DEXIN is related to both non-matched and matched subscriptions, which are handled by the exclusive and inclusive methods, respectively..

4.2. Maintenance costs

4.2.1. Subscribing and unsubscribing time

In order to evaluate the maintenance costs of DEXIN, the performance of the 6 algorithms is measured in experiments by recording the total time used for subscribing different numbers of subscriptions from 0.25 million to 1.75 million. Theoretically, the time complexity of DEXIN for subscribing or unsubscribing a subscription is $O(2M \log u)$, where u is the number of subscriptions stored in the slot that the subscription corresponds to. The experiment results are shown in Fig. 13, where $M = 10, \eta = 10^4, \lambda = 10\%$, and θ is randomly picked from [0.2, 0.8]. From the figure, it can be observed that DEXIN is better than TAMA, H-Tree, BE-Tree and OpIndex, but is slightly worse than REIN. The main reason is that the structure of RB-tree costs extra time to ensure the ascending order of subscriptions in each slot. As is described in Fig. 13, this disadvantage can be fixed by replacing RB-tree with linked list, and the slight increase of event matching time brought by which can be alleviated through increasing the number of slots properly. The performance of unsubscribing is similar to that of subscribing, so we do not provide the results due to page limitation of this paper. Overall, the performance of subscribing and unsubscribing of DEXIN is very competitive to the 5 reference algorithms.

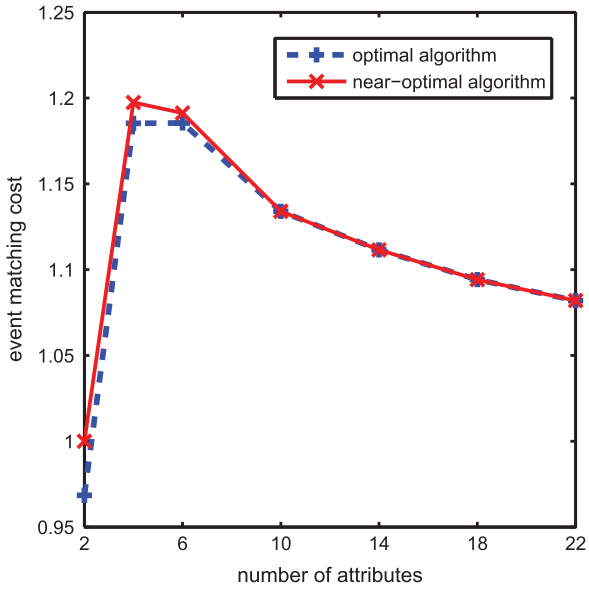

Fig. 15. Event matching cost via optimal and near-optimal algorithm.

4.2.2. Memory consumption

The memory consumption is another metric for evaluating the maintenance costs, which of the 6 algorithms is analyzed in experiments with different numbers of subscriptions from 0.25 million to 1.75 million. Theoretically, the space complexity of DEXIN for storing N subscriptions with M attributes is $O(2NM)$, because on each attribute, a subscription is stored in both arrays in the index structure of the exclusive method, besides which, there are no other duplicates of the subscription. The experiment results are shown in Fig. 14, where $M = 10$, $\eta = 10^4$, $\lambda = 10\%$, and θ is randomly picked from [0.2, 0.8]. It can be seen that the memory consumption of DEXIN is low, which is only slightly worse than that of REIN due to the extra memory costs caused by RB-trees. Using linked list instead of RB-tree can fix this disadvantage as well, the performance of which is illustrated in Fig. 14. The feature of low memory consumption of DEXIN increases the resource utilization, and makes the system able to maintain larger amount of subscriptions with limited memory.

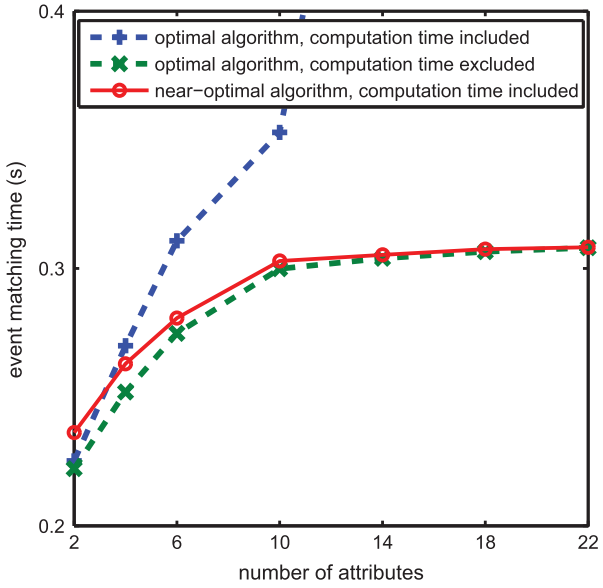

Fig. 16. Event matching time via optimal and near-optimal algorithm.

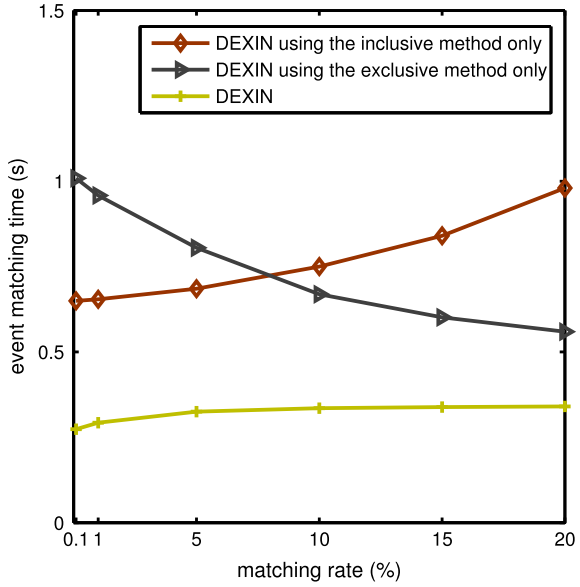

Fig. 17. Event matching time when either inclusive or exclusive method is used by DEXIN.

4.3. Analysis of the near-optimal algorithm

In the pipeline of DEXIN, the sequences and methods adopted for attributes are determined by the near-optimal algorithm, which directly affects the performance of event matching. Hence, we analyze the near-optimal algorithm, and further compare it with the optimal algorithm, which figures out the optimal solution through traversing and checking all feasible solutions that satisfy the Observations 1–3, and its time complexity is an exponential function of the number of attributes M. Experiments are carried out to measure the matching costs and event matching time via the two algorithms. The matching costs with different M from 2 to 22 are illustrated in Fig. 15, where $N = 10^6$, $\eta = 10^4$, $\lambda = 10\%$, and θ is random chosen from [0.2, 0.8]. Uniform distribution and Zipf distribution are randomly used for generating the constraints on different attributes. The above settings ensure a high variety of the matching rates of constraints among attributes. It can be found that the gap between the two curves decreases with the increase of M. The matching costs via the two algorithms are almost the same after $M = 10$. Moreover, we evaluate the average matching time by the two algorithms with different M from 2 to 22. The experiment results are shown in Fig. 16. If the computation time for obtaining the solution is excluded, we can observe that the event matching time via the near-optimal algorithm is very close to that via the optimal algorithm, and the two curves almost coincide after $M = 10$; if the computation time for obtaining the solution is included, we can see that the event matching time via the optimal algorithm is higher than that via the near-optimal algorithm, and the gap between them grows exponentially as the number of attributes rises. Therefore, although the minimum matching costs can be obtained via the optimal algorithm, the computation time brought by which rapidly worsens the holistic event matching time. On the contrary, the solution via the near-optimal algorithm is quite close to the optimal, meanwhile, it only takes a relatively negligible computation time.

4.4. Analysis of DEXIN when using either the exclusive or inclusive method

Experiments are carried out to compare the performance of DEXIN itself, and DEXIN when only using the inclusive method or the exclusive method. The experiment results are shown in Fig. 17, where ξ grows from 0.1% to 20% (The subscriptions with a designated ξ are generated using the same method in Section 4.1.7), and other settings are $N = 10^6$, $M = 10$, $\eta = 10^4$, $\lambda = 10\%$, θ is randomly picked from [0.2, 0.8]. It can be found that the inclusive and exclusive methods perform oppositely with different ξ.

If DEXIN only uses the exclusive method for all single-attribute matchings, the total event matching cost will be

$$\mu = \sum_{k=1}^{M} \mu_{q_k}^{(ex)} = \sum_{k=1}^{M} (1 - \rho_{q_k}) \tag{8}$$

where μ enlarges when the event matching rate over an attribute reduces. Thus, the event matching time via the inclusive method increases as the event matching rate grows.

If DEXIN only uses the inclusive method for all single-attribute matchings, the total event matching cost will be

$$\mu = \sum_{k=1}^{M} \mu_{q_k}^{(in)} = \sum_{k=2}^{M} \prod_{l=q_1}^{q_{k-1}} \rho_l \tag{9}$$

where μ enlarges when the event matching rate over an attribute rises. So the event matching time via the exclusive method decreases as the event matching rate grows.

For a general scenario, the matching rates over all attributes are different, and they vary as the incoming events and subscriptions change. The event matching performance of DEXIN will be severely influenced if only one of the inclusive and exclusive methods are adopted, because the event matching costs of the two methods are in strong correlation with the event matching rates, and their performance will be fluctuated by the changes of the event matching rates.

In DEXIN, the pipelined event matching process is optimized for every incoming event. DEXIN optimally picks one of the two methods for each single-attribute matching. As shown in the previous experiments results, the performance of DEXIN is superior and quite stable for the scenarios with different parameters.

5. Conclusion

DEXIN is proposed in this paper, which is a fast content-based multi-attribute event matching algorithm using dynamic exclusive and inclusive methods. In DEXIN, optimization is done for each incoming event via a suboptimal algorithm, the process of event matching is conducted in a serial pipeline, where the single-attribute matchings are dynamically sequenced, and the two method are dynamically employed for every single-attribute matching. In this way, unnecessary search costs are reduced, thus, the holistic event matching speed is promoted efficiently. The performance of DEXIN is comprehensively evaluated in several aspects, and is compared with that of TAMA, H-Tree, BE-Tree, OpIndex and REIN. The experiment results demonstrate that DEXIN is superior to these reference algorithms. The event matching time of DEXIN not only leads over that of the reference algorithms in most cases, but also has excellent stability in different scenarios. Meanwhile, the subscribing/unsubscribing time and the memory consumption of DEXIN are maintained at a lower level.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (Grant No. 61502050), YangFan Innovative & Entrepreneurial Research Team Project of Guangdong Province, Civil Aerospace Science and Technology Project, Fundamental Research Funds for the Central Universities, and Director Foundation of Beijing Key Laboratory of Work Safety Intelligent Monitoring.

References

[1] P.T. Eugster, P.A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. (CSUR) 35 (2) (2003) 114–131.
[2] G. Muhl, A. Ulbrich, K. Herrman, Disseminating information to mobile clients using publish-subscribe, IEEE Internet Comput. 8 (3) (2004) 46–53.
[3] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, D. Savio, Interacting with the soa-based Internet of things: Discovery, query, selection, and on-demand provisioning of web services, IEEE Trans. Serv. Comput. 3 (3) (2010) 223–235.
[4] X. Chen, Y. Chen, F. Rao, An efficient spatial publish/subscribe system for intelligent location-based services, in: Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, ACM, 2003, pp. 1–6.
[5] F. Cao, J.P. Singh, Efficient event routing in content-based publish-subscribe service networks, in: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, Vol. 2, IEEE, 2004, pp. 929–940.
[6] C. Krügel, T. Toth, C. Kerer, Decentralized event correlation for intrusion detection, in: Information Security and CryptologyICISC 2001, Springer, 2002, pp. 114–131.
[7] M. Fontoura, S. Sadanandan, J. Shanmugasundaram, S. Vassilvitski, E. Vee, S. Venkatesan, J. Zien, Efficiently evaluating complex boolean expressions, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, ACM, 2010, pp. 3–14.
[8] K. Jayaram, C. Jayalath, P. Eugster, Parametric subscriptions for content-based publish/subscribe networks, in: Middleware 2010, Springer, 2010, pp. 128–147.
[9] R.M. Layer, K. Skadron, G. Robins, I.M. Hall, A.R. Quinlan, Binary interval search: a scalable algorithm for counting interval intersections, Bioinformatics 29 (1) (2013) 1–7.
[10] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, Design and evaluation of a wide-area event notification service, ACM Trans. Comput. Syst. (TOCS) 19 (3) (2001) 332–383.
[11] Y. Zhao, J. Wu, Towards approximate event processing in a large-scale content-based network, in: 2011 31st International Conference on Distributed Computing Systems, (ICDCS), IEEE, 2011, pp. 790–799.
[12] S. Qian, J. Cao, Y. Zhu, M. Li, J. Wang, H-tree: An efficient index structure for event matching in content-based publish/subscribe systems, IEEE Trans. Parallel Distrib. Syst. PP (99) (2014) 1–11. http://dx.doi.org/10.1109/TPDS.2014.2323262.
[13] M. Sadoghi, H.-A. Jacobsen, Analysis and optimization for boolean expression indexing, ACM Trans. Database Syst. (TODS) 38 (2) (2013) 8.
[14] D. Zhang, C.-Y. Chan, K.-L. Tan, An efficient publish/subscribe index for e-commerce databases, Proc. VLDB Endow. 7 (8) (2014) 613–624.
[15] S. Qian, J. Cao, Y. Zhu, M. Li, Rein: A fast event matching approach for content-based publish/subscribe systems, in: INFOCOM, 2014 Proceedings IEEE, IEEE, 2014, pp. 2058–2066.
[16] F. Fabret, H.A. Jacobsen, F. Llirbat, J. Pereira, K.A. Ross, D. Shasha, Filtering algorithms and implementation for very fast publish/subscribe systems, in: ACM SIGMOD Record, Vol. 30, ACM, 2001, pp. 115–126.
[17] T.W. Yan, H. García-Molina, Index structures for selective dissemination of information under the boolean model, ACM Trans. Database Syst. (TODS) 19 (2) (1994) 332–364.
[18] Z. Jerzak, C. Fetzer, Bloom filter based routing for content-based publish/subscribe, in: Proceedings of the Second International Conference on Distributed Event-Based Systems, ACM, 2008, pp. 71–81.
[19] S.E. Whang, H. Garcia-Molina, C. Brower, J. Shanmugasundaram, S. Vassilvitskii, E. Vee, R. Yerneni, Indexing boolean expressions, Proc. VLDB Endow. 2 (1) (2009) 37–48.
[20] H. Jafarpour, S. Mehrotra, N. Venkatasubramanian, M. Montanari, Mics: an efficient content space representation model for publish/subscribe systems, in: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, ACM, 2009, pp. 7–12.
[21] Z. Shen, S. Tirthapura, Approximate covering detection among content-based subscriptions using space filling curves, J. Parallel Distrib. Comput. 72 (12) (2012) 1591–1602.
[22] K. jayaram, W. Wang, P. Eugster, Subscription normalization for effective content-based messaging, IEEE Trans. Parallel Distrib. Syst. PP (99) (2014) 1–11. http://dx.doi.org/10.1109/TPDS.2014.2355823.
[23] Y.-M. Wang, L. Qiu, C. Verbowski, D. Achlioptas, G. Das, P. Larson, Summary-based routing for content-based event distribution networks, ACM SIGCOMM Comput. Commun. Rev. 34 (5) (2004) 59–74.
[24] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, et al., Introduction to algorithms, Vol. 2, MIT press, Cambridge, 2001.
[25] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Courier Dover Publications, 1998.

Wenhao Fan received the B.E. and Ph.D. degree from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2008 and 2013, respectively. He is currently an assistant professor at the School of Electronic Engineering in BUPT.

His main research topics include parallel computing and transmission, mobile cloud computing, information security for mobile smartphones, and software engineering for mobile internet.

Yuan'an Liu received the B.E., M.Eng., and Ph.D. degrees in electrical engineering from the University of Electronic Science and Technology, Chengdu, China, in 1984, 1989, and 1992, respectively. He is currently a Professor with BUPT, and he is the Dean of the School of Electronic Engineering, BUPT.

His main research topics include pervasive computing, wireless communications, and electromagnetic compatibility.

Prof. Liu is a Fellow of the Institution of Engineering and Technology, UK; the Vice Chairman of Electromagnetic Environment and Safety of the China Communication Standards Association; the Vice Director of the Wireless and Mobile Communication Committee, Communication Institute of China; and a Senior Member of the Electronic Institute of China.

Bihua Tang received the B.E. degree from the Sichuan University, Chengdu, China, in 1984, and M.Eng. degree from the University of Electronic Science and Technology, Chengdu, China, in 1989, respectively. She is currently a professor at the School of Electronic Engineering in Beijing University of Posts and Telecommunications, Beijing, China.

Her main research topics include mobile computing, wireless sensor networks, wireless networks and hardware engineering for sensors and mobile smartphones. She has published more than 50 papers.