Contents lists available at ScienceDirect

Computer Networks

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

EEM: An elastic event matching framework for content-based publish/subscribe systems[☆]

Yongpeng Dong, Shiyou Qian [*], Wanghua Shi, Jian Cao, Guangtao Xue

Shanghai Jiao Tong University, Dongchuan 800, Shanghai, China

ARTICLE INFO

Keywords:
Publish/subscribe
IoT
Workload
Adaptability
Adjustability

ABSTRACT

The inevitable workload fluctuations in large-scale IoT applications make it difficult to ensure the QoS of data distribution services. Usually, this problem can be solved by dynamically adjusting resource supply. However, deploying more resources not only takes time, it is also difficult to achieve cost-effective results. In this paper, we propose an elastic event matching (EEM) framework and we explore the idea of endowing matching algorithms with adaptability in performance. The strategy is to enable seamless switching between exact matching and approximate matching, achieving a trade-off between matching precision and matching speed. First, we establish a predicate skipping adjustment (PSA) mechanism which quantifies the relationship between false positives and the number of skipped predicates. In addition, we design a performance adjustment decision (PAD) algorithm according to fluctuating workloads. We implemented an EEM prototype based an Kafka, which uses an existing matching algorithm enhanced by PSA as the engine. The prototype is evaluated through a series of experiments based on both synthetic data and real-world stock traces. Experiment results show that adjusting the performance of matching algorithm at the price of a small false positive rate of less than 0.1% can shorten event latency by up to 14.34 times, which clearly demonstrates the effectiveness of the EEM framework.

1. Introduction

The Internet of Things (IoT) is the vision of a ubiquitous network of interconnected physical objects. In IoT applications, sensors are extensively deployed and expected to expand at significant rates in the coming years [1]. These objects can sense and interact with the physical environment, enabling deep real-time awareness of the world around us and new upcoming applications like smart cities, intelligent traffic systems (ITS), and connected and autonomous vehicles. The rapid development of IoT technologies has perpetually generated a vast amount of data with immense application value [2–4]. Therefore, there is a pressing need for an efficient way to achieve data distribution. For example, the congestion status of intersections can be obtained according to the data collected by sensors, and driving vehicles can subscribe to the congestion status of several specific intersections ahead [5].

The publish/subscribe (pub/sub) system is a communication paradigm that is a popular means of disseminating time-sensitive events (messages). Typically, in a pub/sub system, the broker(s) mediates the exchange of events between loosely coupled producers (publishers) and consumers (subscribers). Particularly, the content-based pub/sub system provides subscribers with fine-grained expressiveness, which fits the data distribution requirements in IoT applications well [6–8]. To realize on-demand data distribution, the content-based pub/sub system needs to perform costly event matching. When the number of subscriptions is large, event matching is the potential performance bottleneck of the whole system.

What is worse, the workloads of large-scale IoT applications usually fluctuate [9–11], which makes it challenging to ensure the quality of service (QoS) of data distribution services. Moreover, workload fluctuations can also pose potential threats to the security of publish/subscribe systems [12–14]. The above ITS scenario that disseminates intersection congestion status also faces this problem. At peak hours, the sensors need to collect massive data of vehicles passing intersections and distribute the aggregated congestion status to a large number of vehicles. In such a setting, the system workload depends on the number of vehicles, which exhibits a dynamic property. For example, Fig. 1(a) shows the number of vehicles per seconds during the rush hour (7:30AM–8:30AM) in the morning, with fluctuations of up to 121.73%.[1] In addition, stock exchanges [10] also witness workload

[☆] This document is the results of the research project funded by the National Natural Science Foundation of China (61772334).
[*] Corresponding author.
E-mail address: qshiyou@sjtu.edu.cn (S. Qian).
[1] The data comes from https://dl.acm.org/doi/10.1145/2908557.

https://doi.org/10.1016/j.comnet.2023.109837
Received 31 October 2022; Received in revised form 4 April 2023; Accepted 18 May 2023
Available online 23 May 2023

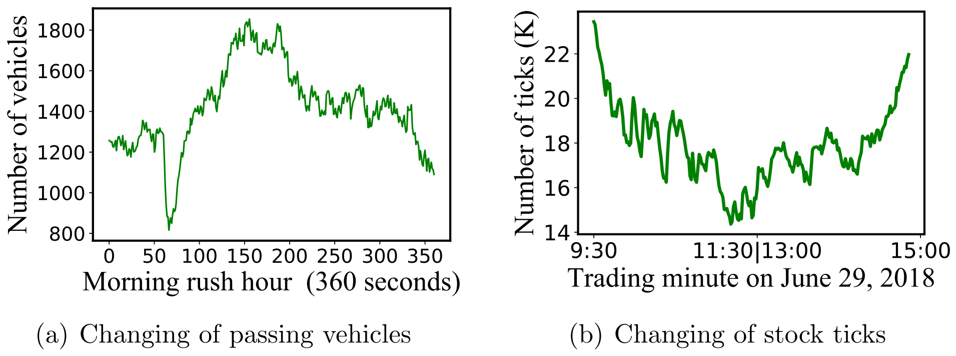

Fig. 1. Fluctuation of workload.

fluctuations over time. To ensure that latency requirements are met, stock exchanges transmit real-time ticks, while subscribers register their interest in ticks related to their investment strategies, such as when the price of a particular stock goes above or below a specified threshold. In such a scenario, the rate at which ticks are generated is dependent on the activities of the stock exchange and displays a dynamic property, where the rate of ticks are produced can change at different times during the day. Fig. 1(b) shows the rate at which ticks are produced minute-by-minute on June 29, 2018, with fluctuations up to 63.2% during the trading periods (9:30–11:30 and 13:00–15:00). In smart grid applications [11], hundreds of thousands of events per second are emitted from millions of consumers, with relevant load changes varying by up to a factor of seven over time.

A natural solution to this problem is to adopt a provisioning technique to adjust the number of brokers to adapt to churn workloads [10, 15,16]. However, this solution has two limitations. First, accurately predicting workloads is crucial for the proper provisioning of brokers, which can be challenging in large-scale systems. Consequently, it is inevitable to lead to over-provisioning or under-provisioning problems. Second, as discussed in the work [16], adjusting the number of brokers takes tens of seconds, which may not meet the strict timeliness requirements in some scenarios.

The root cause of performance bottleneck is the time-consuming event matching operation performed by brokers. Intuitively, one question is, can the performance of matching algorithms be adjusted to adapt to workload fluctuations to a certain degree and gain the critical preparation time for broker provisioning? In general, almost every matching algorithm has its own parameters that can be configured to optimize its performance. For example, the discretization lever in Tama can be configured to trade-off between matching speed and matching precision [17]. In addition, the cache size in Gem is adjustable to balance between matching speed and storage cost [18]. Nevertheless, most existing matching algorithms lack the adaptability to dynamically respond to changing workloads.

In this paper, we explore the idea of endowing the matching algorithm with this adaptability, trying to address workload fluctuations from the perspective of matching algorithm. To realize this idea, the matching algorithm should first have the power of adjustability. This means that the algorithm can adjust its performance dynamically and quantitatively, which is the prerequisite to achieving adaptability. Furthermore, a decision-making algorithm is needed to determine the adjustment scheme according to the changing workload. The combination of these two aspects provides a new method to solve the problem of workload fluctuations.

Therefore, we propose an elastic event matching (EEM) framework. For adjustability, we adopt the strategy of switching between exact matching and approximate matching to realize instantaneous performance adjustment. Thus, we establish a predicate-skipping adjustment (PSA) mechanism that trades off matching precision in favor of matching speed. Specifically, we deduce two models that quantify the relationship between matching speed and matching precision. In addition, we design a performance adjustment decision (PAD) algorithm which is able to recommend proper performance adjustment schemes according to changing workloads.

The PSA mechanism is integrated into an existing matching algorithm REIN [19], giving rise to a variant named WPA-REIN. We also implemented an EEM prototype based an Kafka, which uses WPA-REIN as the matching algorithm. A test bed is set up to verify the adjustment effect of EEM in terms of event transfer latency based on real-world stock tick traces. The experiment results show that adjusting the performance of WPA-REIN at the price of a small false positive rate of less than 0.1% can shorten event latency by almost 14.34 times, which clearly demonstrates the effectiveness of EEM.

The main contributions of this paper are as follows:

- We propose the idea of enhancing matching algorithms with adaptability to instantaneously respond to changing workloads.
- We propose a predicate skipping adjustment (PSA) mechanism and a performance adjustment decision (PAD) algorithm to realize this adaptability.
- We integrate PSA into an existing algorithm and conduct a series of experiments to evaluate the adjustment effectiveness of EEM.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 provides some preliminaries. Section 4 details the design of EEM. Section 5 presents the experiment results. Finally, Section 6 concludes the paper.

2. Related work

Considerable effort has been dedicated to investigating matching algorithms and workload fluctuation in the literature. In this section, we provide a succinct overview of recent research developments from two perspectives: parametric matching algorithms and techniques for managing dynamic workloads.

2.1. Parametric matching algorithms

Since the performance of matching algorithms is critical to a content-based pub/sub system, many efforts have been devoted to this research area, resulting in a large number of efficient matching algorithms, such as H-Tree [20], Be-Tree [21], Gryphon [22], OpIndex [23], Tama [17], Siena [24], K-Index [25], REIN [19], DEXIN [26], Gem [18], MO-Tree [27], HEM [28] and Mics [29]. Generally, almost every matching algorithm has its own parameters that can be configured to optimize its performance. For example, the discretization lever in Tama can be configured to trade-off between matching speed and matching precision [17]. The number of segments split on the attributes' space is an optimization parameter for OpIndex [23]. The number of indexed attributes and the number of cells divided on the attributes' value domain are two key parameters for H-Tree [20]. The cache size in Gem is adjustable to balance between matching speed and storage cost [18].

However, there are two shortcomings in relation to the configurability of the existing matching algorithms. First, the configuration of most existing matching algorithms cannot be performed dynamically. For example, when needing to adjust the matching precision of Tama [17], the underlying data structure must be reconstructed. Second, none of the existing matching algorithms has a parameter that can be used to quantitatively adjust the matching performance.

2.2. Techniques addressing fluctuating workloads

Many researchers explore churn workloads from the perspective of the overlay topology of the pub/sub systems, by either provisioning more brokers or reconstructing the overlay. The work in [30] proposes an autonomic approach which is inspired by the autonomic computing model. It uses the Bayesian learning technique to make decisions about the increase and decrease in resources to accommodate the workload from IoT services in the fog computing environment. The work presented in [31] proposes a technique that aims to optimize publish/subscribe systems through dimension-based subscription pruning. The work defines a selectivity metric that corresponds to the matchability of predicates in subscriptions and uses it to determine which subscriptions to prune based on their impact on network load. However, the decisions made using this metric are based on individual subscriptions and not on the algorithm's data structures. In addition, the work introduces other metrics, such as memory usage and system throughput, to complement the selectivity metric and address other aspects of a distributed publish/subscribe system. The throughput-based pruning technique targets improving filtering efficiency. However, it lacks the ability to adjust automatically to accommodate workload fluctuations in content-based publish/subscribe systems.

The work in [16] proposes GSEC, a general scalable and elastic pub/sub service based on the cloud computing environment, which uses a performance-aware provisioning technique to adjust the number of brokers to adapt to churn workloads. The works similar to GSEC include CAPS [15], MPaS [32] and E-STREAMHUB [10]. In [33], a distributed system called OMen is proposed to dynamically maintain overlays for topic-based pub/sub systems, which supports the churn-resistant construction of topic connected overlays. Other works utilize a load balancing strategy to deal with the workload churn problem through consistent hashing [34,35].

There are two limitations in relation to adjusting the provisioning of brokers to adapt to churn workloads. First, the proper provisioning of brokers depends on the accurate prediction of workloads, which is by nature difficult to predict in a pub/sub system since the rate at which events are produced can vary significantly over time. A typical example is the intelligent traffic system, as discussed above. Second, these kind of methods cannot meet the strict timeliness requirements of some applications. For example, as discussed in [16], adjusting the capacity of GSEC will take tens of seconds to adapt to churn workloads. Although over-provisioning adequate brokers in advance can be a simple solution, it is not cost-efficient.

Differing from these works, we address the problem of fluctuating workloads from the perspective of matching algorithms. The solution proposed in this paper is not to replace the existing provisioning techniques, but is a beneficial complement.

This work is an extension of Ada-REIN [36]. Ada-REIN implemented a coarse-grained dynamic adjustment scheme which is only effective in the case of data skew. Thus, its dynamic adjustment mechanism does not work well when the data is uniformly distributed. In addition, Ada-REIN does not consider the influence of predicate widths on the false positive rate. Under the same conditions, skipping predicates with a larger width is beneficial to reducing the false alarm rate. To overcome these shortcomings, we propose a fine-grained adjustment model which achieves a significant performance improvement.

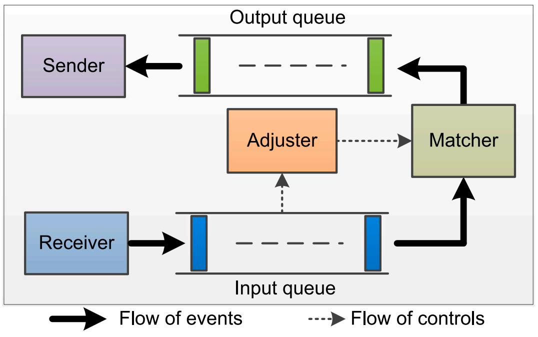

Fig. 2. The whole framework of EEM.

3. Preliminaries

We follow the common data model of events and subscriptions, which is widely used in the existing literature [17–19,29].

An **event** e is expressed as a conjunction of attribute-value pairs. The set of attributes appearing in all event types is defined as $\mathbb{A} = \{a_1, a_2, \dots, a_m\}$. Each attribute appears only once in an event.

An **interval predicate** is a condition defined on an attribute selected from $\mathbb{A}$. It is represented as a triple $\{a_i, v_1, v_2\}$, where a_i is an attribute, v_1 and v_2 are bounded by the value domain of a_i, and v_1 is not larger than v_2. The predicate width w is $v_2 - v_1$.

A **subscription** defines a user's interests in events, which is a conjunction of multiple interval predicates. Each subscription is identified by a unique subID. A subscription matches an event if all the interval predicates contained in the subscription are satisfied when they are assigned the corresponding attribute values of the event.

The **matchability** of a subscription is the matching probability that it matches events. For example, when a subscription is matched with 100 events, if it matches 20 of them, then its matchability is 20%. The matchability of a predicate is the probability that it is satisfied by assigning the corresponding attribute value in an event. Generally, the matchability of a predicate is determined by its width.

4. Design of EEM

4.1. The overall framework

A broker needs three components to collaboratively undertake event matching, namely a receiver, matcher and sender, as shown in Fig. 2. The receiver is responsible for receiving incoming events from upstream peers and injecting them into the input queue. The sender is in charge of transmitting events from the output queue to the next hops (either brokers or subscribers). The matcher runs a matching algorithm which performs selective event filtering according to the subscriptions. The matcher is the connector between the input and the output queue. Note that to obtain high throughput and low latency, there might be multiple instances running at the same time for all or part of these three components, which is the parallelization problem and is out of the scope of this paper.

Ideally, the three components should have equivalent processing capacity, so events get through the receive-match-send procedure without any delay. However, when the event arrival rate is higher than the event matching rate, events are congested in the input queue and the matcher becomes the performance bottleneck. To adapt to fluctuating workloads, a new component called **adjuster** is added to the framework, as shown in Fig. 2. It is responsible for making performance adjustment schemes for the matcher according to the changing workloads.

For the EEM framework to work, two things are essential. First, the matcher, namely the matching algorithm, should have the ability to dynamically and quantitatively adjust its performance. In addition, the adjuster should recommend proper performance adjustment schemes for the matcher according to the fluctuating workloads. In the remainder of this paper, we focus on addressing these two points by proposing a predicate skipping adjustment (PSA) mechanism and a performance adjustment decision (PAD) algorithm.

4.2. Predicate Skipping Adjustment (PSA) mechanism

In this section, we detail the design of PSA, a dynamic and quantitative performance adjustment mechanism. It adopts the predicate skipping strategy to realize a seamless switch between exact matching and approximate matching.

4.2.1. Basic idea

Achieving the adjustability of matching algorithms means meeting two requirements, namely dynamics and quantification. First, performance adjustment can be made anytime, without interrupting the normal matching operations. In practice, it is difficult to meet this requirement because matching algorithms usually rely on their underlying data structures which are time-consuming to adjust. Second, the mechanism should support quantitative performance adjustment.

In terms of matching mode, matching algorithms can be classified into two categories: exact matching and approximate matching. Some approximate algorithms are proposed to pursue a high matching performance, such as Mics [29] and Tama [17]. The metrics that are used to measure matching precision are false positives and false negatives. When an unmatching subscription is judged as a match, a false positive occurs. When a matching subscription is not picked out, a false negative occurs. For exact matching algorithms, neither false positives nor false negatives are allowed. Approximate matching is to trade off matching precision in favor of matching speed, usually allowing for a small false positive rate. When false positives exist, subscribers may receive events that are not interesting to them.

The basic idea of the PSA mechanism is to achieve seamless switching between exact matching and approximate matching to support instantaneous performance adjustment. Specifically, according to changing workloads, PSA selects a certain number of predicates to skip for the purpose of improving matching performance. For instance, Ada-REIN is a matching algorithm that incorporates the PSA mechanism into REIN. In this way, Ada-REIN can dynamically switch between exact and approximate matching modes without the need to reconstruct the underlying index structure. Ada-REIN operates by incorporating an additional input parameter F, which represents the allowable false positive rate of event matching. If F is set to 0, Ada-REIN functions as an exact matching algorithm. Alternatively, if F is non-zero, Ada-REIN operates in the approximate matching mode. Specifically, Ada-REIN computes the expected false positive rate for each attribute by taking into account its frequency, and subsequently decides which attributes to skip based on their associated false positive rates. Moreover, distinct events can be matched at different false positive rates.

Based on Ada-REIN, a finer-grained PSA mechanism is proposed in this work, as illustrated in Fig. 3, where each rectangle represents a predicate, and its width denotes the width of the interval predicate. It can be observed that Ada-REIN's adaptive approach mainly focuses on a single dimension, i.e., selecting low-frequency attributes for skipping. However, this coarse-grained adaptive mechanism inevitably limits its performance to a certain extent. In contrast, WPA-REIN considers two dimensions, namely predicates and their width. WPA-REIN gives preference to skipping the predicates with larger width, which can utilize the false positive rate more efficiently. Specifically, under the same conditions, WPA-REIN can skip as many predicates with larger width as possible, effectively using the false positive rate. Therefore, by utilizing a predicate-width granularity skipping mechanism, WPA-REIN can significantly improve both matching efficiency and system stability. To quantify this relationship, we establish two models that describe the correlation between the specified false positive rate and the corresponding improvement in matching performance.

Table 1
Description of symbols.

Symbol	Description
$\mathbb{S}$	Set of subscriptions
k_i	Size of subscription s_i
$\bar{k}$	Average of subscriptions' size
$w_{i,j}$	Width of the jth predicate contained in s_i
$\bar{w}$	Average of predicates' width
m	Dimensionality of content space
m_i	Size of event e_i
P^i_j	Probability of subscription s_i that contains j predicates
X	Skipped attributes
f_i	Appearance frequency of attribute a_i
f_X	Appearance frequency of skipped X attributes
Y	Number of skipped predicates
F	False positive rate

4.2.2. Matching precision model

In this subsection, we derive a model to quantify the relationship between the false positive rate and the number of predicates to be skipped. The symbols used in this work are described in Table 1.

Given a set of n subscriptions $\mathbb{S} = \{s_1, s_2, \dots, s_n\}$, we refer to the size of a subscription s_i as the number of predicates k_i ($k_{min} \le k_i \le k_{max}$) contained in s_i, where k_{min} and k_{max} represent the lower bound and the upper bound of subscription size. Let $w_{i,j}$ be the matchability of the jth predicate in subscription s_i. For simplicity, let $\bar{k}$ and $\bar{w}$ represent the average of subscriptions' size and predicates' width in $\mathbb{S}$, respectively. The matching algorithm we design is able to support the multiple event types whose attributes collectively form the m-dimensional content space. Then, let m_i be the number of attribute-value pairs in event e_i. Let f_i be the appearance frequency of attribute a_i for $i = 1, 2, \dots, m$ in the set of subscriptions $\mathbb{S}$.

Lemma 1. *Suppose the attributes of subscriptions are uniformly selected from the set of event attributes. When X attributes are skipped in the matching process, the probability P^i_j that subscription s_i contains j predicates defined on the X skipped attributes is upper-bounded by $\frac{C^{k_i-j}_{m-X} \cdot C^j_X}{C^{k_i}_m}$ for* $(0 \le j \le k_i)$.

Proof. The attributes from all event types form the content space. The dimension of content space is m. Selecting k_i attributes from the content space to define the predicates contained in subscription s_i is equivalent to selecting k_i balls from m ones where $m-X$ balls are white and X balls are red. When balls are uniformly selected, the selecting probability of all balls, no matter white or red, is the same at the beginning. In the first case where the selected balls are returned, the selecting probability remains constant, which is $\frac{1}{m-X}$ and $\frac{1}{X}$ for white and red balls, respectively. However, in the second case where the selected balls are not returned, the selecting probability of balls decreases with each selection, and is less than when the selected balls are returned since each attribute appears only once in a subscription, which corresponds to the second case. Therefore, P^i_j is upper-bounded by

$$P^i_j \le \frac{C^{k_i-j}_{m-X} \cdot C^j_X}{C^{k_i}_m} \quad (0 \le j \le k_i) \quad \square \tag{1}$$

Lemma 2. *Suppose the attributes of subscriptions are uniformly selected from the set of event attributes. Given the X attributes to be skipped and their appearance frequency f_X in subscriptions, the probability P^i_j that subscription s_i contains j predicates defined on the X skipped attributes is upper-bounded by $C^j_{k_i} \cdot (\frac{f_X}{\sum_{i=1}^n k_i})^j \cdot (1 - \frac{f_X}{\sum_{i=1}^n k_i})^{k_i-j}$ for* $(0 \le j \le k_i)$.

Proof. Since the frequency of the X skipped attributes is f_X, the probability of selecting a skipped attribute can be expressed by $\frac{f_X}{\sum_{i=1}^n k_i}$.

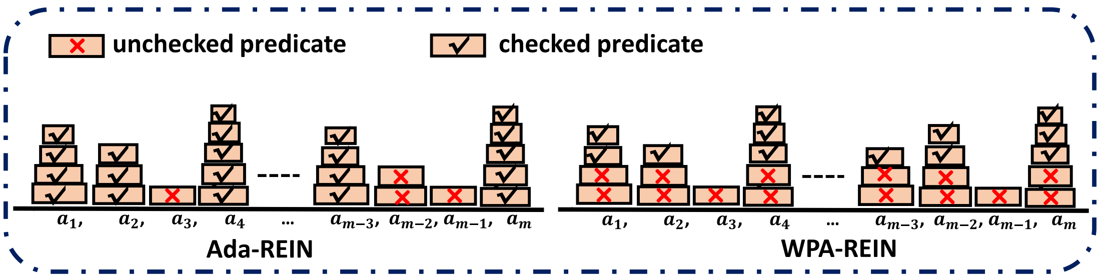
unchecked predicate
checked predicate
a_1, a_2, a_3, a_4 ... a_{m-3}, a_{m-2}, a_{m-1}, a_m
Ada-REIN
a_1, a_2, a_3, a_4 ... a_{m-3}, a_{m-2}, a_{m-1}, a_m
WPA-REIN

Fig. 3. The visualization of PSA for Ada-REIN and WPA-REIN.

Given this probability, as proved by Lemma 1, P_j^i is upper-bounded by

$$P_j^i \le C_{k_i}^j \cdot (\frac{f_X}{\sum_{i=1}^n k_i})^j \cdot (1 - \frac{f_X}{\sum_{i=1}^n k_i})^{k_i - j} \quad (0 \le j \le k_i) \quad \square \tag{2}$$

Lemma 3. *Given P_j^i, for each subscription s_i and $0 \le j \le k_i$, when X attributes are skipped in the event matching process, the expected size of subscriptions $\bar{k}'$ is $\sum_{i=0}^n \sum_{j=0}^{k_{max}} P_j^i \cdot (k_i - j)$.*

Proof. When an attribute is skipped in the event matching process, if a subscription contains a predicate defined on the attribute, the real size of the subscription is reduced. For $0 \le j \le k_i$, the probability P_j^i that subscription s_i contains j predicates defined on the X skipped attributes is given in Eq. (2) when the appearance frequency of the X skipped attributes is available. With the probability P_j^i $(0 \le j \le k_i)$, the expected size of subscriptions $\bar{k}'$ is

$$\bar{k}' = \sum_{i=0}^{n} \sum_{j=0}^{k_{max}} P_j^i \cdot (k_i - j) \quad \square \tag{3}$$

Lemma 4. *Given $\bar{k}$ and $\bar{k}'$, the expected false positive rate F is $\bar{w}^{\bar{k}} \cdot (1 - \bar{w}^{\bar{k}-\bar{k}'})$ when X attributes are skipped.*

Proof. There are n subscriptions, each of which contains different number predicates and all the interval predicates have different widths. Since the matchability of each subscription decreases monotonically with the increase in subscription size, skipping X attributes leads to an increase in the matchability of the subscriptions that contain the skipped predicates, which is $\bar{w}^{\bar{k}'}$. When the attributes are skipped, all predicates defined on these attributes are not checked. Thus, the average size of subscriptions is shortened. The expectation of false positive rate F arising from skipping Y attributes is

$$F = \bar{w}^{\bar{k}} \cdot (1 - \bar{w}^{\bar{k}-\bar{k}'}) \quad \square \tag{4}$$

Theorem 1. *Given a false positive rate F, the number of predicates that can be skipped in the event matching process is $Y = n(\log_w(F+1))$.*

Proof. Substituting F into Eq. (5) gets the expected subscription size $\bar{k}$ for the given false positive rate F.

$$\bar{k}' = \log_{\bar{w}}(F + \bar{w}^{\bar{k}}) \tag{5}$$

On average, $\bar{k} - \bar{k}'$ predicates are skipped in each subscription. The expected number of predicates that can be skipped without offending F for n subscriptions is

$$Y = n(\bar{k}' - \log_{\bar{w}}(F + \bar{w}^{\bar{k}'})) \quad \square \tag{6}$$

Algorithm 1 shows the pseudo code of selecting skipped predicates. As the number of allowable skipped predicates is determined by the false positive rate, skipping predicates with a larger width can more effectively utilize the false positive rate. Therefore, we classify the predicates into L classes based on their predicate width. In this context, L represents the number of classes used for this categorization (line 1).

Algorithm 1 Selecting skipped predicates

```
Require: Subscriptions set $S$ and the false positive $F$
Ensure: The skipped predicate cells
 1: Classify the predicates defined on each attribute into $L$ classes
    based on the width of predicates;
 2: Use Eq. (6) to calculate the total number of skipped predicates $Y$;
 3: Initialize $skip[]$ to all False;
 4: for each class in $L$ do
 5:     Calculate attributes frequencies array $freq$;
 6:     Sort $freq$ in non-decreasing order ;
 7:     Select the attributes that can be fully skipped at the attribute-
        width granularity without exceeding the total number of
        skipped predicates $Y$;
 8:     Divide the predicates of each attribute that is not fully skipped
        into $B$ cells;
 9:     Sort the cells in non-decreasing order;
10:     Select the smallest cell until the number of skipped predicates
        is near to $Y$;
11:     $skip[selectedPredicate] = True$;
```

Given the false positive rate F, the total number of skipped predicates Y can be computed according to Eq. (6) (line 2). Furthermore, according to the total number of skipped predicates, the occurrence frequency of each attribute $freq$ and B cells of predicates for each attribute, we use the greedy strategy to select the predicates to be skipped starting from the maximum width layer and mark them (lines 4–11). Furthermore, each cell corresponds to a bucket that is based on the attribute domain and is utilized to store predicates into which predicate values fall.

Through in-depth analysis, we know that the time complexity of Algorithm 1 mainly consists of three parts: (i) The operation of categorizing kn predicates contained in n subscriptions of size k is $O(kn)$. (ii) The cost of sorting m attributes divided into L categories by width is $O(Lm \log m)$. (iii) The cost of sorting the mB cells in each of the L categories is $O(LmB \log mB)$. Therefore, the time complexity of Algorithm 1 is $O(kn + Lm \log m + LmB \log mB)$

4.3. Performance Adjustment Decision (PAD) algorithm

Instead of making the adjustment scheme according to the congestion degree of events at brokers, we specify an upper bound on the false positive rate F_{max} by analyzing the cost efficiency between false positives and performance improvement, for the following reasons. First, the position of our proposed solution is clear, as a complement to the broker provisioning techniques to deal with workload churn to a certain extent. Second, when the performance of the matching algorithm needs to be improved, network resources contention is also intense. Allowing a larger false positive rate will increase network contention.

Generating an adjustment scheme is in essence a control problem and we are inspired by the spirit of PID [37] to design PAD, as shown in Algorithm 2. The rationale of the PAD algorithm is to achieve a

Algorithm 2 PAD Algorithm

```
Require: window size $W$
1: $F \leftarrow 0,\ F_{last} \leftarrow 0,\ t \leftarrow 0,\ t_{last} \leftarrow 0$;
2: for the incoming each $W$ events to the broker do
3:     $t \leftarrow$ the average latency of $W$ events
4:     if $t \geq \max\{t_{upperThreshold}, t_{last}\}$ then
5:         $F \leftarrow \min\{F_{last} + \Delta_F, F_{max}\}$
6:     else $t < \min\{t_{lowerThreshold}, t_{last}\}$
7:         $F \leftarrow \max\{F_{last} - \Delta_F, 0\}$
8:     Match the $W$ events with $F$
9:     $F_{last} \leftarrow F,\ t_{last} \leftarrow t$
```

trade-off between matching speed and matching precision by adjusting the false positive rate F in a self-adaptive way. In Algorithm 2, we use the sliding window method to calculate the congestion degree of each window. We use W to represent the size of the window, and the constant Δ_F to denote the amount of change in the false positive. When the broker receives W events (line 2), it first calculates how long these events have been waiting in the input queue (line 3), which reflects the congestion degree of events at the broker. If the latency t is larger than the maximum of the normal one $t\_upperThreshold$ and the last one t_{last} (line 4), it can be inferred that events are congested at the broker at this moment. In this case, F is set to $\min\{F_{last} + \Delta_F, F_{max}\}$, which can further improve the matching performance (line 5) without exceeding F_{max}. Similarly, when event congestion is alleviated, F is gradually decreased to achieve high matching precision (lines 6–7). As can be seen, the PAD algorithm adaptively converge to the case where exactly no congestion occurs at the broker, and the false positive rate is minimized. The time complexity of PAD is dependent on the window size, denoted by W. In each window, the latency of events is measured. Specifically, to determine a false positive rate for the each window, the time complexity of PAD is $O(W)$.

5. Experiments

5.1. Evaluation of PSA

To enable the mechanism of PSA, the adjustability of existing matching algorithm REIN was enhanced using PSA to create a variant called WPA-REIN. REIN [19] is a backward matching algorithm that quickly searches for unmatching subscriptions to indirectly identify the matching ones. It offers two advantages that meet PSA's requirements: First, REIN's index structure is predicate-oriented, which aligns well with the spirit of the adjustment mechanism. Second, predicates defined on the same attribute are indexed together in REIN, independently of other attributes. This independence makes it possible to quantify the relationship between the matching time and matching precision, thereby satisfying the quantification design requirement.

When implementing WPA-REIN, we made two main modifications based on REIN. First, we modify the data structure of REIN by introducing multiple layers based on the width of predicates. Second, based on the matching procedure of REIN, we decompose its matching time into four parts: predicate comparison time, bucket traversal time, bucket switch time and bitset mark time. The purpose of the four components is to determine the percentage of performance improvement that can be achieved by skipping Y predicates. Consequently, it becomes evident that skipping the predicate primarily reduces the time spent on predicate comparison, bucket traversal, and bucket switching. Based on these metrics and the given false positive rate F, we use Eq. (6) to compute the number of skipped predicates in WPA-REIN. The detailed calculation model and the source code of WAP-REIN are available.[2]

[2] https://github.com/citsjtu2020/EEM.git

Table 2
Parameters used in experiments.

Note	Description	Value
n	Number of subscriptions	1 M
m	Dimensionality of content space	500
α	Parameter of attributes' Zipf distribution	0.8
k	Subscription size	5
w	Matchability of interval predicates	0.5
B	Number of buckets in REIN	1000
L	Number of index layers in WPA-REIN	5

Table 3
Effectiveness verification results of PSA.

F	R	Ada-REIN			WPA-REIN		
		Y_1	F_1	R_1	Y_2	F_2	R_2
0.1%	35.2%	2131	0.09%	2.97%	4173	0.08%	37.20%
0.3%	36.12%	7404	0.29%	6.25%	12536	0.26%	38.93%
0.5%	37.73%	16681	0.49%	9.96%	20912	0.40%	44.95%
1%	39.14%	21218	0.97%	14.77%	41924	0.65%	47.20%
3%	43.38%	82395	2.98%	21.07%	127064	1.57%	50.01%
5%	47.24%	139333	4.97%	24.19%	214074	2.50%	55.17%

5.1.1. Experiment setup

WPA-REIN is implemented in C++ language and compiled by g++ 9.3.0 with -O3 optimization enabled on an Ubuntu 20.04 system. All the experiments are conducted on an AMD R9 5900X CPU with a base clock speed of 3.7 GHz. The capacity of main memory is 64 GB.

For brevity, the value domain of attributes is normalized on [0, 1.0] from which the predicate values and event values are uniformly generated with precision 10^{-6}. The parameters used in the experiments are listed in Table 2. 1000 events are matched in each experiment and each experiment is repeated 10 times. The matching time is used to evaluate the performance of the matching algorithms.

5.1.2. Experiment results

In the experiments, WPA-REIN with different expected false positive rates is compared to REIN and Ada-REIN. For each expected false positive rate, the number of skipped predicates is counted, which are used to compute the expected rate of performance improvement. In addition, the number of matching subscriptions of WPA-REIN and Ada-REIN is counted in the experiment, which is compared with that of REIN to measure the real false positive rate. The matching time of REIN is used as the baseline to measure the performance improvement of WPA-REIN and Ada-REIN. The results are listed in Table 3, where $n = 1M$, $k = 5$, $m = 500$ and $w = 0.5$. As Ada-REIN does not filter predicates in the case of uniform distribution, we have set $\alpha = 2$. In the table, the first and second columns represent the expected false positive rate and expected rate of performance improvement, respectively. The third to fifth columns show the results for Ada-REIN, including the number of skipped predicates (Y_1), the measured false positive rate (F_1), and the measured rate of performance improvement (R_1). Similarly, the sixth to eighth columns represent the results for WPA-REIN, including the number of skipped predicates (Y_2), the measured false positive rate (F_2), and the measured rate of performance improvement (R_2).

As shown in Table 3, the measured false positive rate and performance improvement are very close to the expected ones, which clearly demonstrates the effectiveness of the PSA mechanism. Specifically, the measured false positive rate is a little smaller than the expected one while the measured performance improvement is not less than the expected one. The philosophy behind this is that the expected false positive rate should be an upper bound, limiting the costs; while the expected performance improvement should be a lower bound, ensuring the gains.

The performance improvement does not linearly grow with the false positive rate. This phenomenon is explainable. At the beginning, even with a small false positive rate, a certain number of interval predicates

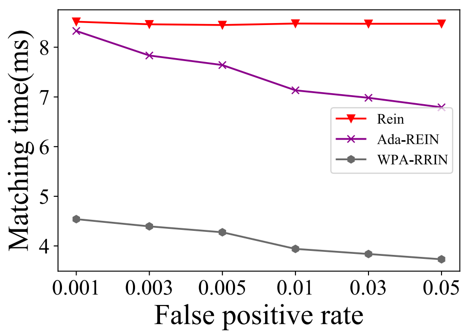

Fig. 4. Performance comparison between REIN, Ada-REIN and WPA-REIN.

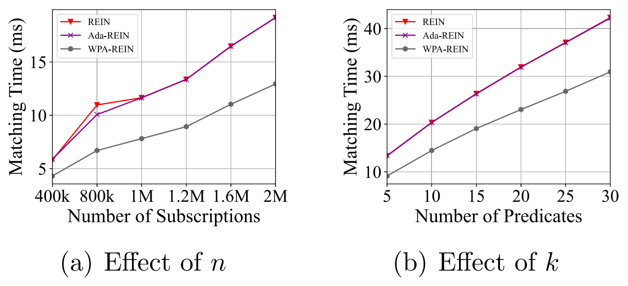

Fig. 5. Matching time with different subscription number and size.

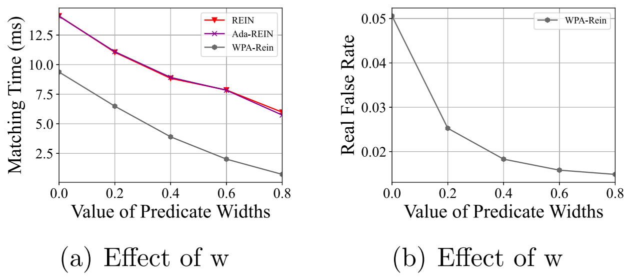

Fig. 6. Matching time with different predicate width.

with high matching probability are skipped. Thus, the obtained performance improvement is considerable. For example, when the expected false positive rate is 0.1%, the measured performance improvement is 37.20%. In this case, 4173 predicates out of 6000000 ones are skipped. However, increasing the false positive rate does not bring proportional performance improvement. For example, when the expected false positive rate is 1%, the measured performance improvement is 47.20%. Although the false positive rate increases tenfold compared with 0.1%, the performance improvement only grows by 10%. This is because that not all skipped predicates need to be marked, and these repeated marking operations account for most of the matching time, up to 92% [28].

In comparison to Ada-REIN, WPA-REIN demonstrates superior performance under the same conditions. For example, when the expected false positive rate is set at 5%, the performance improvement achieved by Ada-REIN and WPA-REIN is up to 24.19% and 55.17%, respectively, compared to REIN. Specifically, WPA-REIN skips more predicates than Ada-REIN while maintaining a lower false positive rate, which can be attributed to two main factors. Firstly, WPA employs a more fine-grained predicate skipping mechanism, compared to Ada-REIN's attribute skipping mechanism. In Ada-REIN, if the frequency of an attribute exceeds the number of allowable skipping predicates, that attribute cannot be skipped, which decreases the expected false positive efficiency. Secondly, WPA-REIN utilizes a probabilistic hierarchical index structure based on predicate width. The wider a predicate, the higher its matching probability, and predicates with larger width are given higher priority for skipping. This approach results in fewer false positives than Ada-REIN. Fig. 4 provides a more detailed representation of the performance of REIN, Ada-REIN and WPA-REIN as the expect false rate increases. REIN is an exact matching algorithm that lacks a filtering mechanism, thus the false positive rate has no impact on its matching time.

Therefore, from the viewpoint of cost effectiveness, the "Elbow Rule" can be used to determine the expected false positive rate [38]. In this experiment, WPA-REIN with a false positive rate 0.1% is more cost efficient. So, in the next experiments, we compare WPA-REIN of 1% false positive rate with REIN to evaluate the impact of different parameters. The performance of matching algorithms can be influenced by various parameters, including the number of subscriptions, subscription size, event size, and other factors. We comprehensively evaluate the performance of WPA-REIN with different settings of these parameters.

Impact of the number of subscriptions n. The number of subscriptions n is a critical parameter to measure the workload, which has a significant impact on the matching time. Fig. 5(a) shows the matching time of the three tested algorithms with different setting of n. Intuitively, all algorithms take longer to match events with more subscriptions and exhibit an upward trend. WPA-REIN performs better than their counterparts in all situations. Compared with REIN and Ada-REIN, WPA-REIN reduces the matching time by 32.0% and 32.7% respectively on average. When $m = 20$, Ada-REIN has the same performance as REIN. This is because in the low dimensionality of content space, the frequency of attribute is relatively large, which causes Ada-REIN to not skip fewer attributes. On the other hand, WPA-REIN uses a more precise skipping mechanism, resulting in significant performance improvements.

Impact of subscription size k. As shown in Fig. 5(b), subscription size k has an important effect on the performance of event matching. To measure the effect of k, we set $m = 30$ and vary k from 5 to 30. when $k = 5$, WPA-REIN makes about a 32% performance improvement over REIN and Ada-REIN, whereas the improvement only reduces to 27% when $k = 30$. The reason for this issue is the higher frequency of attributes as the subscription size increases. When WPA-REIN skips the same number of predicates, it processes fewer attributes. As a result, the performance of WPA-REIN is affected.

Impact of predicate width w. The matching probability is determined by subscription size k and predicate width w. Fig. 6(a) shows how width affects the matching time by varying w from 0.2 to 1. In this set of experiments, width w is a dynamic parameter. For instance, 0.8 represents the minimum value in the range of 0.8 to 1. k is set to 5 to ensure there are always some matches. This setting makes the predicates sparsely distributed in cells meanwhile. As the predicate width increases, the matching time of REIN, Ada-REIN and WPA-REIN sharply decrease. This improvement is attributed to the backward matching strategy utilized by all three algorithms. A larger predicate width results in a higher matching probability of a subscription. During the matching process, all three algorithms are able to process fewer unmatched subscriptions. Additionally, when compared to REIN and Ada-REIN, WPA-REIN's performance can be improved by up to 87.4% and 87.9%, respectively. This is due to WPA-REIN's more fine-grained skipping mechanism that prioritizes skipping predicates with larger widths, which also minimizes the false positive rate, as depicted in Fig. 6(b).

Impact of the dimensionality of content space m. Supporting high dimensional spaces is one of the challenges for matching algorithms. Given n and k, the higher the dimensionality, the sparser the workload in the space. Fig. 7 shows a remarkable tendency that the three algorithms behave monotonically. This is because all three backward matching algorithms need to process more attributes to mark all the

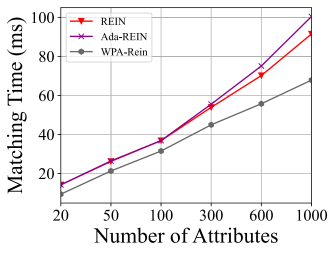

Fig. 7. Matching time with different dimensionality.

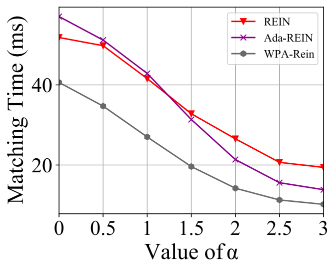

Fig. 8. Matching time with different attribute skew.

unmatched subscriptions. Furthermore, when subscriptions are uniformly distributed, the performance of Ada-REIN becomes worse and worse. Even when $m = 100$, the performance of Ada-REIN is not as good as that of REIN. Compared with REIN and Ada-REIN, on average the performance of WPA-REIN is improved by 21.7% and 24.2%, respectively.

Impact of attribute distribution α. The skewed distribution of subscriptions and events is divided into two categories. One is the value distribution and the other is attribute distribution. On account of the fact that the uneven distribution of values has little effect on the matching time, here we show the experiment results under skewed attribute distribution of both subscriptions and events in Fig. 8(a) where m is 500. When considering the key parameter α in the Zipf distribution, it is worth noting that the higher the value of α, the greater skew in the data, resulting in some attributes being more popular than others. As shown in the figure, the performance of the three algorithms is sensitive to the distribution of attributes.

For Ada-REIN with $\alpha = 3$ where the popular attributes are concentrated, more attributes belong to the infrequent category, which provides more candidates to skip without offending the expected false positive rate. Even though, compared with REIN and Ada-REIN, WPA-REIN improves the performance by 47.5% and 26.4% respectively. On the other hand, when $\alpha = 0$, that is uniform distribution, the performance of Ada-REIN is not as good as that of REIN. This is because that Ada-REIN spends time searching for low-frequency attributes and does not find any that could be filtered. On the contrary, WPA-REIN overcomes this disadvantage of Ada-REIN by utilizing its fine-grained skip mechanism. WPA-REIN outperforms REIN and Ada-REIN by up to 21.7% and 28.8%, respectively.

Impact of the number of levels L. Fig. 9(a) shows the matching time of WPA-REIN under different settings of L. We observe that L has important effects on WPA-REIN. Compared with REIN and Ada-REIN, WPA-REIN improves the performance by up to 37.1% and 37.1%, respectively on average. This is due to the hierarchical index structure based on predicate width, which changes the distribution of subscriptions in buckets. In other words, the hierarchical index structure makes subscriptions more concentrated in certain buckets and increases empty buckets. More empty buckets can be skipped before matching, which reduces the cost of traversing buckets. Fig. 9(b) shows that the real false rate dramatically drops from 3.7% to 0.21% when L increases from 1 to 50. The contributing factor for this phenomenon is the width-aware hierarchical index structure. During the process of selecting skipped predicates, priority should be given to the predicates with larger width, so as to reduce the false rate.

The experiments conducted above demonstrate the superior performance of WPA-REIN when compared to REIN and Ada-REIN. This is primarily due to the hierarchical index structure of WPA-REIN, which allows it to prioritize skipping predicates with larger widths without significantly increasing the false positive rate.

5.2. Evaluation of EEM

We implemented a prototype of EEM based on Kafka 2.13-3.2.1 and constructed a test bed to evaluate the effectiveness of the prototype. The test bed is composed of three virtual machines (VMs) running Ubuntu 18.04.1. Two VMs are used to imitate a publisher that generates events at changing rates and to simulate a certain number of subscribers who submit subscriptions to the broker, respectively. The EEM framework is run at the third VM. We measure event latency under changing workloads in terms of event generation rate. In the experiments, the prototype uses REIN, Ada-REIN and WPA-REIN as the matcher.

5.2.1. Dataset

The event data comes from a real-world stock dataset with 50 attributes after being cleaned up. The original data was collected from the Chinese stock market on June 8, 2018 from the Wind Economic Database.[3] After normalization processing, most event values are smaller than 0.5. The subscription data is stochastically generated based on this stock dataset.

5.2.2. Workload generation

We set the event generation rate in the experiments to be proportional to the actual generation rate in the dataset. In this sense, the rate at which events are produced well simulates the workload fluctuations of the real-world stock exchanges.

The experiment procedure can be divided into two stages: submitting subscriptions and publishing events. In the first stage, 1000,000 subscriptions are synthesized and submitted to the broker. Each subscription contains five predicates and the matchability of predicates is set to 0.4. The selection of subscription attributes follows a uniform distribution. In the second stage, the publisher generates events at a changing rate which is proportional to the real one in the stock dataset. Whenever the broker receives an event, it judges the congestion degree, determines the allowed false positive rate, matches the event with the subscription, and sends the events to interested subscribers. We measure the overall latency of events from generation to the moment when each interested subscriber receives the events. The publish stage lasts 30 s and the average latency of events at each second is calculated. Since the system workload is fluctuating, the average latency per second will vary with the workloads. In the experiment, the PAD algorithm is used to determine the false positive rate for each event where $F_{max} = 0.1\%$, $\Delta_F = 0.02\%$, $t_{upperThreshold} = 20$ ms, and $t_{lowerThreshold} = 15$ ms.

5.2.3. Experiment results

To fully demonstrate the performance of EEM, we conducted two sets of experiments with workload fluctuation under uniform and Zipf distribution.

Workload under uniform distribution. The changing workload under uniform distribution is shown in Fig. 10(a). The average publishing rate

[3] http://www.wind.com.cn/en/

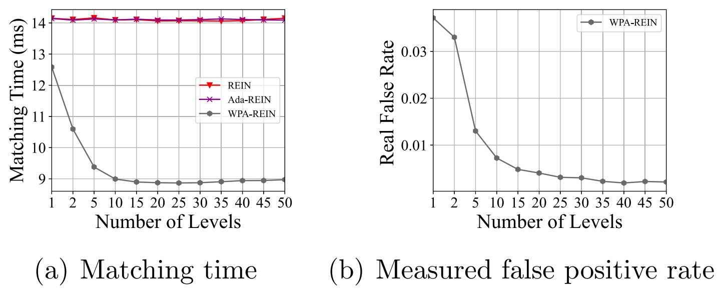

Fig. 9. Matching time and measured false positive rate with different L.

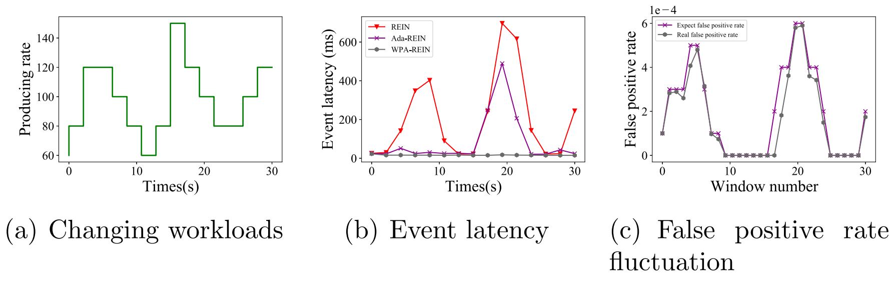

Fig. 10. Changing workloads, event latency and false positive rate under uniform distribution.

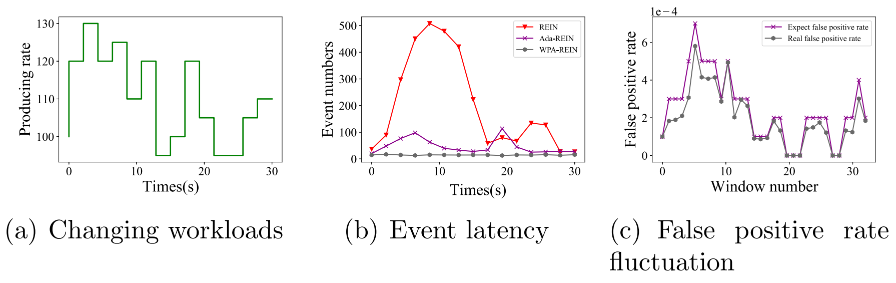

Fig. 11. Changing workloads, event latency and false positive rate under Zipf distribution.

is 97 events per second and the fluctuation degree is up to 93.10%. The attributes and their values follow uniform distribution. The event latency calculated at each second is shown in Fig. 10(b). Overall, the average event latency of WPA-REIN is shortened by 12.53 and 5.23 times compared to REIN and Ada-REIN, reducing from 204.80 ms and 85.41 ms to 22.46 ms respectively. The reason for this improvement is that when encountering a workload spike, REIN does not have the adjustability to adapt to the changing workloads, thus events are congested at the broker, having a cumulative effect on event latency. Since Ada-REIN skips predicates defined on infrequent attribute, it does not work well in the case of uniform distribution. On the contrary, WPA-REIN has the ability to alleviate event congestion, stabilizing the latency of events. Compared to REIN and Ada-REIN, the standard variance of the event latency of WPA-REIN is improved by almost 112.63 and 67.09 times respectively. Furthermore, according to the workload fluctuations, Fig. 10(c) shows that EEM can adjust the false positive rate to achieve instantaneous performance adjustment without breaking the expect false positive rate, based on workload fluctuations. When the latency is low, it can switch to an exact matching model. This result not only confirms the effectiveness of EEM but also demonstrates its remarkable ability to achieve seamless switching between approximate and exact matching. Thus, this finding provides strong evidence to support the practical value of EEM in handling diverse and dynamic workloads.

Workload under Zipf distribution. The workload under Zipf distribution is presented in Fig. 11(a). The average publishing rate is 110 events per second and the fluctuation degree is 31.82%. Attributes and their values follow Zipf distribution with the skew factor of $\alpha = 0.8$. The Zipf distribution widely exists in many fields [39]. Under the condition that the Zipf distribution is satisfied and the workload fluctuation is relatively weak, WPA-REIN makes 14.34 times and 3.33 times performance improvement over REIN and Ada-REIN. As shown in Fig. 11(b), it can be seen that Ada-REIN also has good performance improvement, because Zipf distribution makes attributes more concentrated and can skip more predicates without offending the expected false positive rate. In addition, compared with REIN and Ada-REIN, the standard variance of the event latency of WPA-REIN is improved by almost 39.66 times and 8.26 times respectively. Fig. 11(c) presents a comparison between the expected and real false positive rates, indicating that EEM operates within the expected false positive rate limit, thus providing strong evidence of its effectiveness.

6. Conclusions

We explore the idea of enhancing the adaptability of matching algorithms to deal with workload fluctuations. To do so, we propose an elastic event matching (EEM) framework which consists of a predicate skipping adjustment (PSA) mechanism and a performance adjustment decision (PAD) algorithm. Augmenting REIN with PSA gives rise to WPA-REIN. To evaluate the adjustment effectiveness of EEM, a series of experiments are conducted based on both synthetic data and real-world stock tick data. The experiment results show that, even after adjusting the performance of WPA-REIN at the price of a small false positive rate of less than 0.1%, event latency can be shortened by almost 14.34

times, which clearly demonstrates the feasibility of our exploratory idea. In the future, optimization of EEM will be pursued in two key areas. Firstly, the system's throughput will be enhanced by optimizing the costs associated with predicate comparison, bucket traversal, and bucket switching. Secondly, the performance of EEM with respect to latency perception will be improved by identifying the optimal false positive rate.

CRediT authorship contribution statement

Yongpeng Dong: Conceptualization, Methodology, Writing – original draft. **Shiyou Qian:** Supervision. **Wanghua Shi:** Software. **Jian Cao:** Writing – review & editing. **Guangtao Xue:** Writing – review & editing.

Declaration of competing interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Data availability

Data will be made available on request.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61772334).

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, IEEE Trans. Ind. Inform. 14 (11) (2018) 4724–4734.

[2] A. Lindgren, F.B. Abdesslem, B. Ahlgren, O. Schelén, A.M. Malik, Design choices for the IoT in information-centric networks, in: 2016 13th IEEE Annual Consumer Communications & Networking Conference, CCNC, 2016, pp. 882–888.

[3] M.S. Zahedinia, M. Khayyambashi, A. Bohlooli, Fog-based caching mechanism for IoT data in information centric network using prioritization, Comput. Netw. 213 (2022) 109082.

[4] T. Zaarour, A. Bhattacharya, E. Curry, OpenPubSub: Supporting large semantic content spaces in peer-to-peer publish/subscribe systems for the internet of multimedia things, IEEE Internet Things J. 9 (18) (2022) 17640–17659.

[5] M. Jaseemuddin, A. Alam, N. Gawhar, MQTT pub-sub service for connected vehicles, in: 2021 IEEE 18th International Conference on Smart Communities: Improving Quality of Life using ICT, IoT and AI, HONET, 2021, pp. 167–172.

[6] T. Salman, R. Jain, A survey of protocols and standards for internet of things, 2019, CoRR abs/1903.11549.

[7] C.E. Arruda, P.F. Moraes, N. Agoulmine, J.S.B. Martins, Enhanced pub/sub communications for massive IoT traffic with SARSA reinforcement learning, 2021, CoRR abs/2101.00687.

[8] P.F. Moraes, R.F. Reale, J.S.B. Martins, A publish/subscribe QoS-aware framework for massive IoT traffic orchestration, 2018, CoRR abs/1806.03157.

[9] A. Shraer, M. Gurevich, M. Fontoura, V. Josifovski, Top-k publish-subscribe for social annotation of news, Proc. VLDB Endow. 6 (6) (2013) 385–396.

[10] R. Barazzutti, T. Heinze, A. Martin, E. Onica, P. Felber, C. Fetzer, Z. Jerzak, M. Pasin, E. Rivière, Elastic scaling of a high-throughput content-based publish/subscribe engine, in: IEEE ICDCS, 2014, pp. 567–576.

[11] R.C. Fernandez, M. Weidlich, P.R. Pietzuch, A. Gal, Scalable stateful stream processing for smart grids, in: U. Bellur, R. Kothari (Eds.), The 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, Mumbai, India, May 26-29, 2014, ACM, 2014, pp. 276–281.

[12] M. Dahlmanns, J. Pennekamp, I.B. Fink, B. Schoolmann, K. Wehrle, M. Henze, Transparent end-to-end security for publish/subscribe communication in cyber-physical systems, in: Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, 2021, pp. 78–87.

[13] M. Padmaja, S. Shitharth, K. Prasuna, A. Chaturvedi, P.R. Kshirsagar, A. Vani, Grow of artificial intelligence to challenge security in IoT application, Wirel. Pers. Commun. (2021) 1–17.

[14] S. Selvarajan, G. Srivastava, A.O. Khadidos, A.O. Khadidos, M. Baza, A. Alshehri, J.C.-W. Lin, An artificial intelligence lightweight blockchain security model for security and privacy in IIoT systems, J. Cloud Comput. 12 (1) (2023) 1–17.

[15] X. Ma, Y. Wang, X. Pei, F. Xu, A cloud-assisted publish/subscribe service for time-critical dissemination of bulk content, Concurr. Comput.: Pract. Exper. 29 (8) (2017) 1–15.

[16] Y. Wang, X. Ma, A general scalable and elastic content-based publish/subscribe service, IEEE Trans. Parallel Distrib. Syst. 26 (8) (2015) 2100–2113.

[17] Y. Zhao, J. Wu, Towards approximate event processing in a large-scale content-based network, in: IEEE ICDCS, 2011, pp. 790–799.

[18] W. Fan, Y. Liu, B. Tang, GEM: An analytic geometrical approach to fast event matching for multi-dimensional content-based publish/subscribe services, in: IEEE INFOCOM, 2016, pp. 1–9.

[19] S. Qian, J. Cao, Y. Zhu, M. Li, REIN: A fast event matching approach for content-based publish/subscribe systems, in: IEEE INFOCOM, 2014, pp. 2058–2066.

[20] S. Qian, J. Cao, Y. Zhu, M. Li, J. Wang, H-Tree: An efficient index structurefor event matching in content-BasedPublish/Subscribe systems, IEEE Trans. Parallel Distrib. Syst. 26 (6) (2015) 1622–1632.

[21] M. Sadoghi, H.-A. Jacobsen, Be-tree: An index structure to efficiently match boolean expressions over high-dimensional discrete space, in: ACM SIGMOD, 2011, pp. 637–648.

[22] M.K. Aguilera, R.E. Strom, D.C. Sturman, M. Astley, T.D. Chandra, Matching events in a content-based subscription system, in: ACM PODC, 1999, pp. 53–61.

[23] 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.

[24] A. Carzaniga, A.L. Wolf, Forwarding in a content-based network, in: ACM SIGCOMM, 2003, pp. 163–174.

[25] 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.

[26] W. Fan, Y.A. Liu, B. Tang, DEXIN: A fast content-based multi-attribute event matching algorithm using dynamic exclusive and inclusive methods, Future Gener. Comput. Syst. 68 (2017) 289–303.

[27] T. Ding, S. Qian, J. Cao, G. Xue, Y. Zhu, J. Yu, M. Li, MO-tree: An efficient forwarding engine for spatiotemporal-aware pub/sub systems, IEEE Trans. Parallel Distrib. Syst. 32 (4) (2021) 855–866.

[28] W. Shi, S. Qian, HEM: A hardware-aware event matching algorithm for content-based pub/sub systems, in: Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part I, Springer, 2022, pp. 277–292.

[29] H. Jafarpour, S. Mehrotra, N. Venkatasubramanian, M. Montanari, MICS: An efficient content space representation model for publish/subscribe systems, in: ACM DEBS, 2009, pp. 1–12.

[30] M. Abbasi, M. Yaghoobikia, M. Rafiee, A. Jolfaei, M.R. Khosravi, Efficient resource management and workload allocation in fog–cloud computing paradigm in IoT using learning classifier systems, Comput. Commun. 153 (2020) 217–228.

[31] S. Bittner, A. Hinze, Dimension-based subscription pruning for publish/subscribe systems, in: 26th International Conference on Distributed Computing Systems Workshops, ICDCS 2006 Workshops, 4-7 July 2006, Lisboa, Portugal, IEEE Computer Society, 2006, p. 25.

[32] N. Ahmed, MPaS: A micro-services based publish/subscribe middleware system model for IoT, in: 2022 5th Conference on Cloud and Internet of Things, CIoT, 2022, pp. 220–225.

[33] C. Chen, R. Vitenberg, H.-A. Jacobsen, Omen: Overlay mending for topic-based publish/subscribe systems under churn, in: Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems, 2016, pp. 105–116.

[34] J. Gascon-Samson, F.P. Garcia, B. Kemme, J. Kienzle, Dynamoth: A scalable pub/sub middleware for latency-constrained applications in the cloud, in: IEEE ICDCS, 2015, pp. 486–496.

[35] A.K.Y. Cheung, H.A. Jacobsen, Load balancing content-based publish/subscribe systems, ACM Trans. Comput. Syst. 28 (4) (2010) 1–55.

[36] S. Qian, W. Mao, J. Cao, F.L. Mouël, M. Li, Adjusting matching algorithm to adapt to workload fluctuations in content-based publish/subscribe systems, in: IEEE INFOCOM, 2019, pp. 1936–1944.

[37] K.H. Ang, G. Chong, Y. Li, PID control system analysis, design, and technology, IEEE Trans. Control Syst. Technol. 13 (4) (2005) 559–576.

[38] R.L. Thorndike, Who belongs in the family? Psychometrika 18 (4) (1953) 267–276.

[39] Y. Yang, J. Zhu, Write skew and zipf distribution: Evidence and implications, ACM Trans. Storage 12 (4) (2016).

Yongpeng Dong is currently pursuing a Ph.D. with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include event matching for content-based publish/subscribe systems.

Shi-You Qian (Member, IEEE and CCF) received a Ph.D. in computer science from Shanghai Jiao Tong University, China, in 2015. He is currently an associate researcher with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include event matching for content-based publish/subscribe systems, resource scheduling for the hybrid cloud, and driving recommendations with vehicular networks.

Wanghua Shi is currently pursuing a master degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include event matching for content-based publish/subscribe systems.

Jian Cao (Member, IEEE and CCF) received a Ph.D. from the Nanjing University of Science and Technology, China, in 2000. He is currently a professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His main research interests include service computing, network computing, and intelligent data analytics.

Guangtao Xue (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2004. He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include vehicular ad hoc networks, wireless networks, mobile computing, and distributed computing. He is a member of the IEEE Computer Society and the IEEE Communication Society.